Vertrouwelijk: Machine Learning voor het verlagen van buffervoorraden

Naam	Studentnummer	Eerste examinator	Bedrijf	Datum
Stefan A. Rang	1655299	Jos Schmeltz	EKB Houten	27-6-2018

Sjabloon: V4.0

VOORWOORD

Voor u ligt de scriptie ‘Vertrouwelijk: Machine Learning voor het verlagen van buffervoorraden’ en is het resultaat van onderzoek bij EKB Houten. Dit afstudeeronderzoek is bedoeld ter voltooiing van mijn opleiding HBO-ICT Software & Information Engineering aan de Hogeschool Utrecht.

Graag wil ik mijn docentbegeleider, Frank Verbruggen, en bedrijfsbegeleider Auke Roelofsen bedanken voor de goede communicatie, feedback en sturing van dit onderzoek.

Ook bedank ik Maarten Kok voor de ondersteuning met betrekking tot de EMI-software.

Ik wens u veel leesplezier toe.

Stefan A. Rang

Houten, 27 juni 2018

MANAGEMENTSAMENVATTING

Deze scriptie beschrijft de kwestie, het onderzoek en de implementatie van een machine learning algoritme voor het verlagen van voorraden die zich tussen verschillende productielijnen van een fabriek bevinden. Het verlagen van deze voorraden heeft impact op de omzet van de fabriek.

Een klant van EKB, Tsubaki Nakashima, bepaald de hoogte van de buffervoorraden op dit moment op basis van ervaring. Het is voor Tsubaki Nakashima onduidelijk hoe de buffervoorraden verlaagd kunnen worden zonder dat dit van invloed is op de productie van de fabriek.

Het doel van dit onderzoek is om met machine learning algoritmes in EKB Manufacturing Intelligence (EMI) de buffervoorraden te verlagen terwijl de productie gelijk blijft of stijgt. Hiervoor is de volgende onderzoeksvraag verwoord:

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

Om deze onderzoeksvraag te kunnen beantwoorden is er allereerst deskresearch gedaan naar de data die nodig zijn om de algoritmes op te kunnen laten trainen en waar deze data vandaan gehaald kunnen worden. Hieruit bleek dat de meeste data uit de database van EMI gehaald kan worden en de overige data uit de database van Tsubaki Nakashima.

Hierna is er onderzoek gedaan naar verschillende simulatie softwarepakketten en machine learning algoritmes. Uit dit onderzoek is gebleken dat de simulatiesoftware JaamSim het meest geschikt is en het SharpNEAT-algoritme het meest geschikte machine learning algoritme is om in EMI te implementeren om hiermee de buffervoorraden van Tsubaki Nakashima te verlagen.

Tot slot is er experimenteel onderzoek gedaan om te controleren of de algoritmes voldoen aan de verwachtingen van EKB. **TODO!!! (Samenvatting conclusie van Deelvraag 4)**

TODO!!! (Samenvatting aanbevelingen)

Inhoudsopgave

FIGUREN- EN TABELLENLIJST

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 2.5)
GA	Genetic algorithm (voor de definitie zie paragraaf 4.2.4)
ML	Machine learning (voor de definitie zie hoofdstuk 4.2)
NEAT	Neuro Evolution of Augmenting Topologies (voor de definitie zie paragraaf 4.2.5)
OEE	Overall Equipment Effectiveness (voor de definitie zie paragraaf 2.5.1)
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 2.4)
TOC	Theory of Constraints (zie voor de definitie hoofdstuk 4.1)

BEGRIPPENLIJST

BEGRIP	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 2.4.2)
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

Het afstudeerbedrijf, Elektro Kasten Bouwen Industriële Automatisering (EKB), is gericht op het automatiseren van industriële bedrijfsprocessen zoals het aansturen van machines. Daarnaast kan EKB haar klanten ook inzicht geven in de prestatie van fabrieken door middel van de software genaamd EKB Manufacturing Intelligence (EMI).

Dit onderzoek is gericht op het implementeren van machine learning algoritmes in EMI om de voorraden tussen fabrieksmachines te verlagen waardoor de omzet kan stijgen.
Machine learning is een tak van artificial intelligence (AI), kunstmatige intelligentie, en werd bedacht in de jaren vijftig door Alan Turing. Sindsdien zijn er velen verschillende vormen van machine learning ontstaan en worden sommigen hiervan gebruikt door huidige techgiganten als Google, Microsoft en Apple.

Voor zover duidelijk is er nog geen onderzoek gedaan en zijn er nog geen toepassingen van machine learning om buffervoorraden te verlagen. Daarnaast wordt machine learning nog niet in EMI toegepast. Wel gebruikt EKB machine learning voor automatische sortering van onder andere groente en fruit op basis van camera's. Echter is dit een andere vorm van machine learning dan de vorm die relevant is voor dit onderzoek.

EKB wil met de implementatie van machine learning in EMI zorgen dat klanten een indicatie krijgen van buffervoorraadgroottes die voordeliger zijn dan de huidige.

In hoofdstuk 2 wordt het afstudeerbedrijf EKB verder geïntroduceerd en zijn de bedrijfs- en persoonsgegevens opgenomen. Ook wordt er belangrijke informatie gegeven over EMI en Tsubaki Nakashima, een klant van EKB, die als business case voor deze afstudeeropdracht fungeert.

Hoofdstuk 3 beschrijft de kwestie en het onderzoek van deze afstudeeropdracht. Vervolgens wordt in hoofdstuk 4 de opgedane kennis over Theory of Constraints en machine learning beschreven.

Het onderzoek met haar resultaten en deelconclusies zijn terug te vinden in hoofdstuk 5 van deze scriptie. Hoofdstuk 6 bevat de realisatie van de machine learning algoritmes in EMI. De eindconclusie van dit onderzoek staat in hoofdstuk 7 en geeft een antwoord op de onderzoeksvraag.

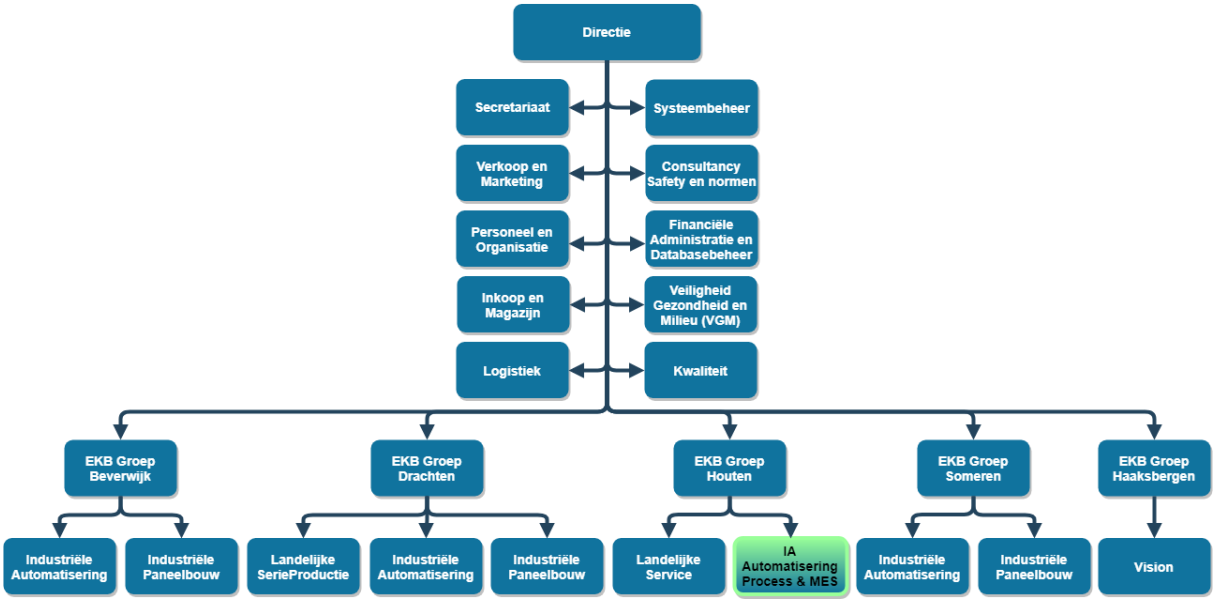
Tot slot worden de aanbevelingen gegeven en wordt de procesgang van de afstudeeropdracht geëvalueerd.

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 weergeven 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	http://www.ekb.nl/nl/home/

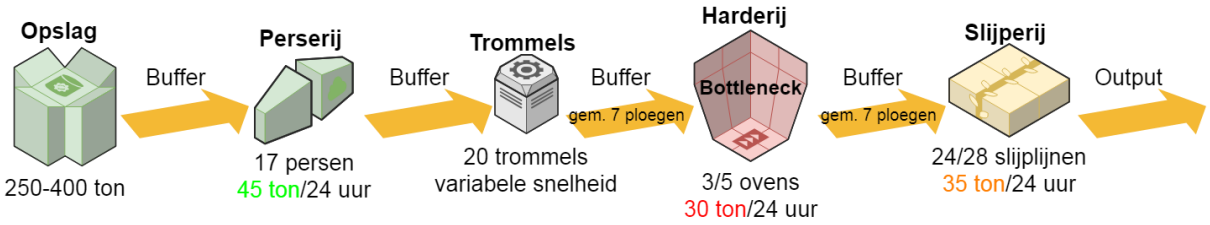
2.3 Persoonsgegevens

Tabel 2: Persoonsgegevens van betrokkenen

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

2.4 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

2.4.1 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.

2.4.2 Buffervoorraden

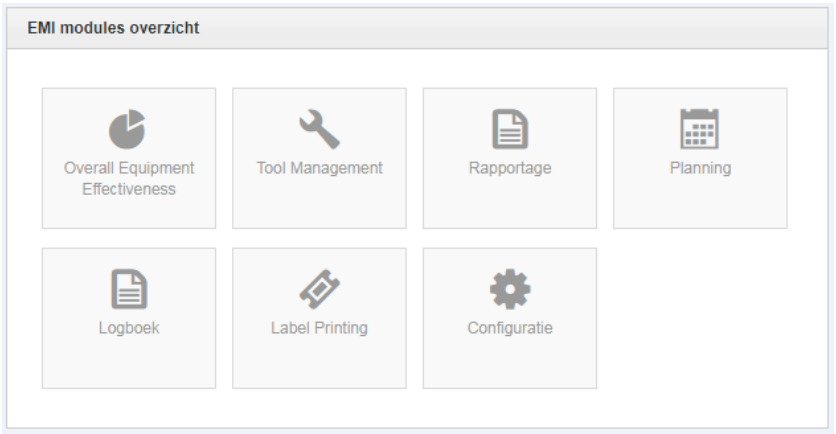
Tussen de hallen bevinden zich de buffervoorraden. Dit zijn voorraden van containers met rollen die nog verwerkt moeten worden door de volgende hal. In het geval van de buffervoorraden tussen de opslag en de perserij zijn dit echter geen containers maar staalhaspels. De buffervoorraden voor en na de 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.

2.4.3 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.

2.5 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.



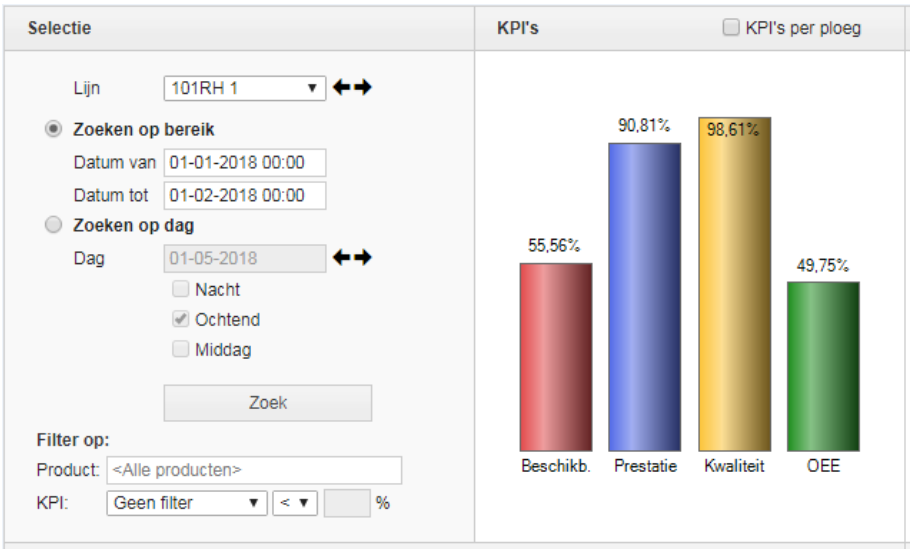
Figuur 3: EMI modules van de rollenfabriek van Tsubaki Nakashima
Noot. Rang, S. A. (2018, 3 april). EMI modules van de rollenfabriek van Tsubaki Nakashima [Foto], Geraadpleegd van http://emi-demo.ekb.nl/EMI_NNN/Default.aspx[1]

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.

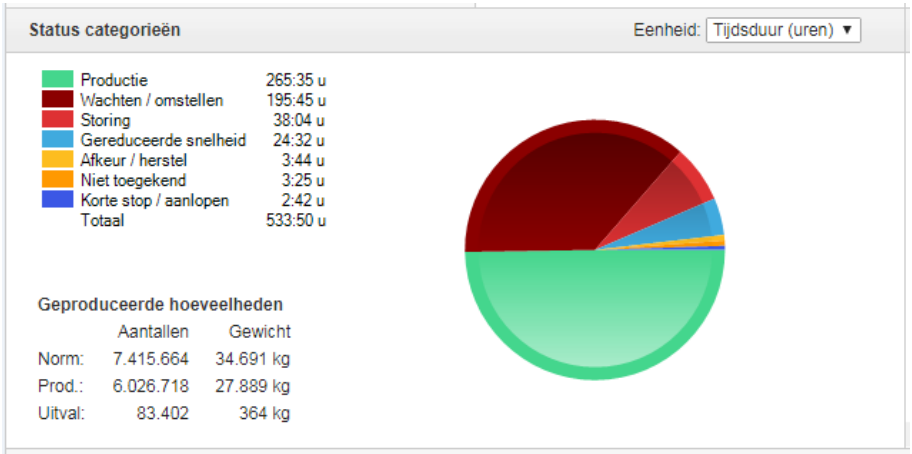
2.5.1 Overall Equipment Effectiveness

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.
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.

[1] Bron afkomstig van de EMI software (niet publiekelijk toegankelijk) van EKB.



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[2]



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[2]

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).

[2] Bron afkomstig van de EMI software (niet publiekelijk toegankelijk) van EKB.

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 worden 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 omzet 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 het aantal geproduceerde producten gelijk te houden of te verhogen terwijl de voorraden tussen de verschillende productielijnen verlaagd worden. Hierdoor verminderen de kosten per product en stijgt de omzet van de fabriek.

3.4 Hoofdvraag en deelvragen

Uitgaande van de opdrachtomschrijving 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 moeten 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

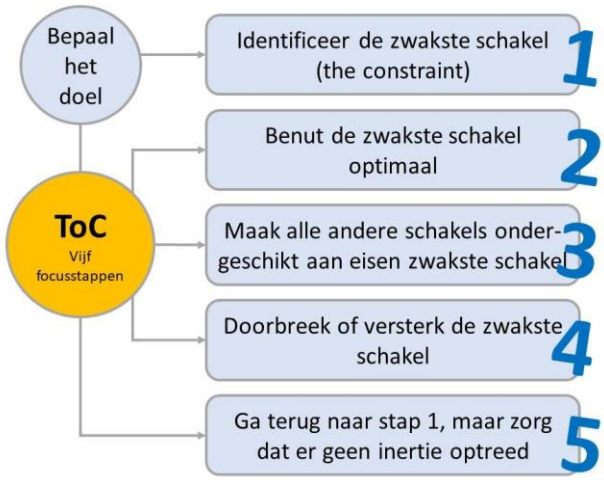
4 THEORETISCH KADER

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 Theory of Constraints

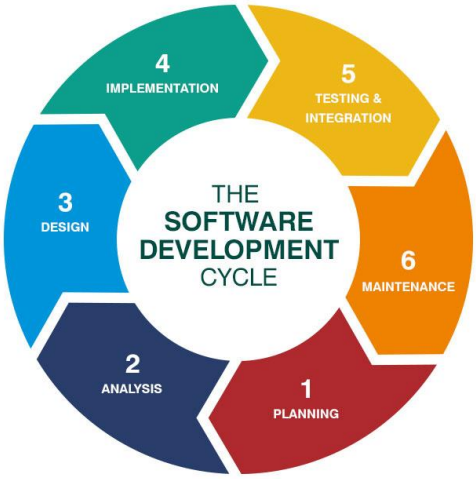
In 2000 schreef Goldratt het boek *'Het Doel'* waarin de basisprincipes van TOC op spannende wijze worden verteld. Volgens een samenvatting van Goldratt en Cox (2007) gaat dit boek over een manager die zijn fabriek wil verbeteren. De manager komt erachter dat lokale verbeteringen irrelevant zijn en dat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

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.

4.1.1 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):

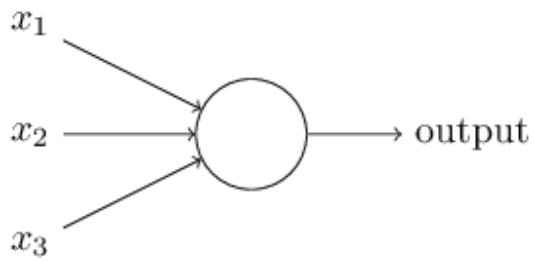
1. 'Value', het identificeren welke producten en diensten de klant van waarde vindt
2. 'Value stream', het identificeren van bedrijfsprocessen met toegevoegde waarde en het elimineren van verliezen die van invloed zijn op de doorlooptijd
3. 'Flow', het zorgen voor stroming en betere doorlooptijden. Hiervoor 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.

4.2 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.

4.2.1 De perceptron

Aan de basis van ML staan zogeheten Neural Networks. Dit zijn simpele netwerken die outputs berekenen door het vermenigvuldigingen van inputs. De meest simpele vorm hiervan is een perceptron. Volgens Nielsen (2017) kan een perceptron een ouput 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

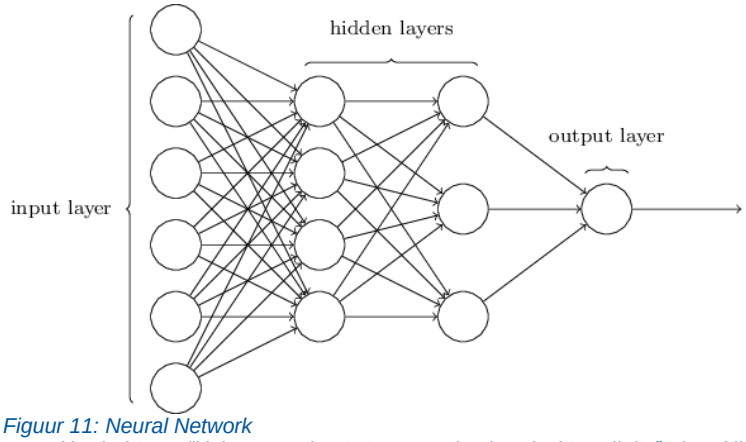
$$\text{output} = \begin{cases} 0 & if \ \sum_j w_j x_j \leq threshold \\ 1 & if \ \sum_j w_j x_j > threshold \end{cases}$$

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.

4.2.2 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

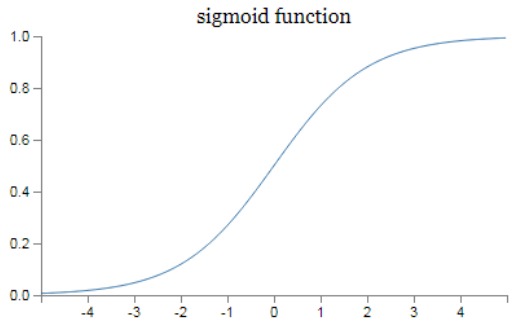
In het geval van neural networks wordt echter vaak de term '*neuron*' gebruikt in plaats van perceptron. Naast de waarde (x) en de bijbehorende weights (w_n) heeft een neuron ook een *bias*. Dit is een extra variabele die wordt opgeteld bij de output van de neuron. Hierna wordt nog een extra berekening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

4.2.3 Activation Functions

In plaats van een threshold heeft een neuron een activation function om de output te berekenen. Een veelvoorkomende activation function is de *sigmoïd* (Figuur 12). Het resultaat van de standaard sigmoïd 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.

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

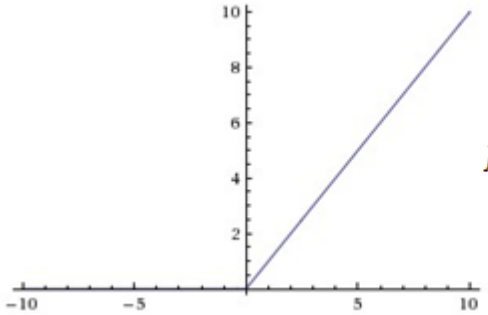
Figuur 12: Standaard sigmoïd formule



Figuur 13: Plot van de sigmoïd 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.

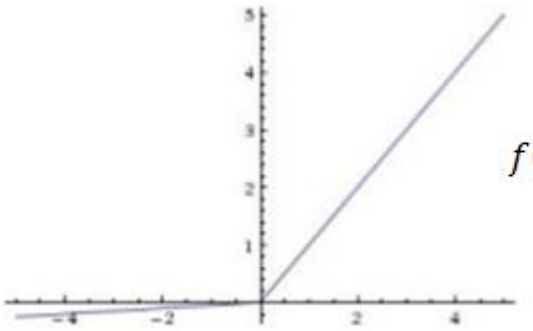
Een andere activation function die steeds vaker gebruikt wordt is de *Rectified Linear Unit (ReLU)*. Het resultaat van de ReLU formule is 0 voor alle negatieve waardes en veranderd niets aan de positieve waardes (Figuur 14).



$$f(x) = \max(0, x)$$

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-inteligence/content/relu_layer.html

In 2017 onderzocht Sharma de voor- en nadelen van de ReLU formule. Een belangrijk voordeel is dat de negatieve waardes 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 waardes. 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 veranderd 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 waardes hetzelfde als de standaard ReLU. Voor negatieve waardes is de helling niet 0 maar 0,01. Hiermee is het stervende ReLU probleem verholpen.



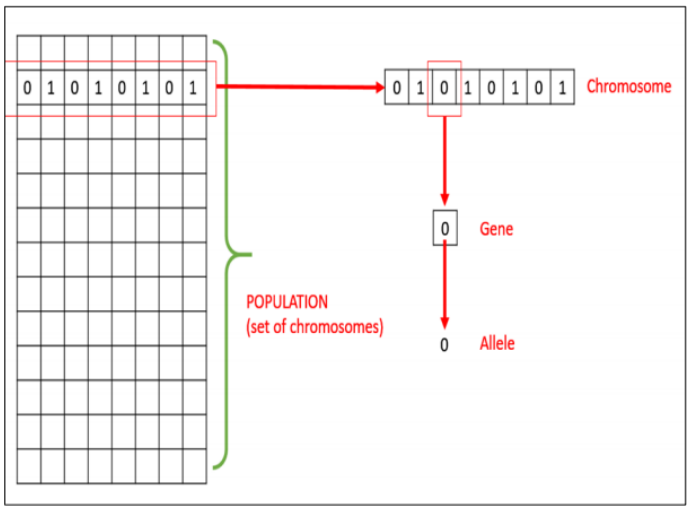
$$f(x) = \max(0.01x, x)$$

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.

4.2.4 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.

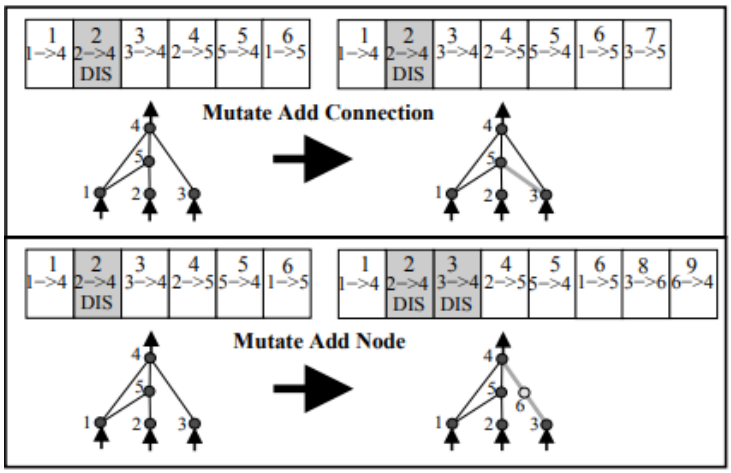


Figuur 16: Basis structuur en terminologie van GA's
Noot. Herdrukt van "Genetic Algorithms", door Tutorials Point (I) Pvt. Ltd.., (2016). Geraadpleegd van https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_tutorial.pdf

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.

4.2.5 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.



Figuur 17: Aanvullende mutatiemogelijkheden van het NEAT algoritme
Noot. Herdrukt van "Efficient Evolution of Neural Networks though Complexification", door Stanley, K., (2004). Geraadpleegd van http://nn.cs.utexas.edu/downloads/papers/stanley.phd04.pdf

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.

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 verzameling

Om te kunnen simuleren zijn er allereerst data over de rollenfabriek van TN nodig. Deze data zouden 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

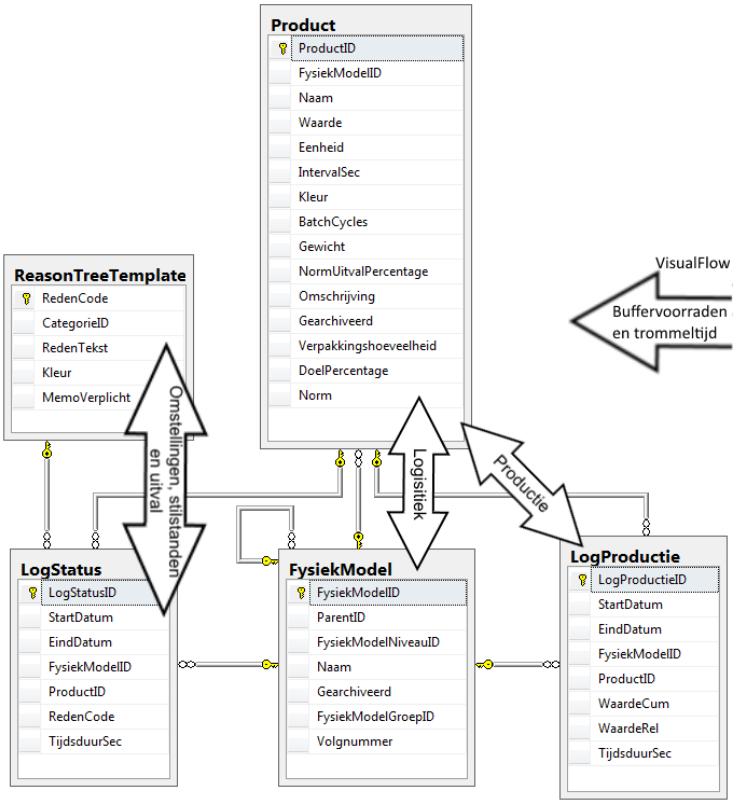
De logistiek van de fabriek is nodig om in de simulaties vast te kunnen leggen welke productielijnen er allemaal zijn en wat de relatie tussen deze productielijnen is. Welke routes kunnen producten allemaal afleggen van het magazijn tot aan de opslag aan het eind van de fabriek?
Als een productielijn 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 zijn er ook data nodig over de buffervoorraden aan het begin van de simulatie periode.

5.1.2 Interne data

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[3]

5.1.3 Uitzonderingen op interne data

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 zijn er dus externe data nodig. In het geval van TN zijn 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.

[3] Bron afkomstig van het intranet (niet publiekelijk toegankelijk) van EKB.

Ook de **trommels** van de rollenfabriek van TN staan niet geregistreerd in de EMI database. Zoals aangegeven in hoofdstuk 2.4.1 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.

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 Simulatiesoftware afweging

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

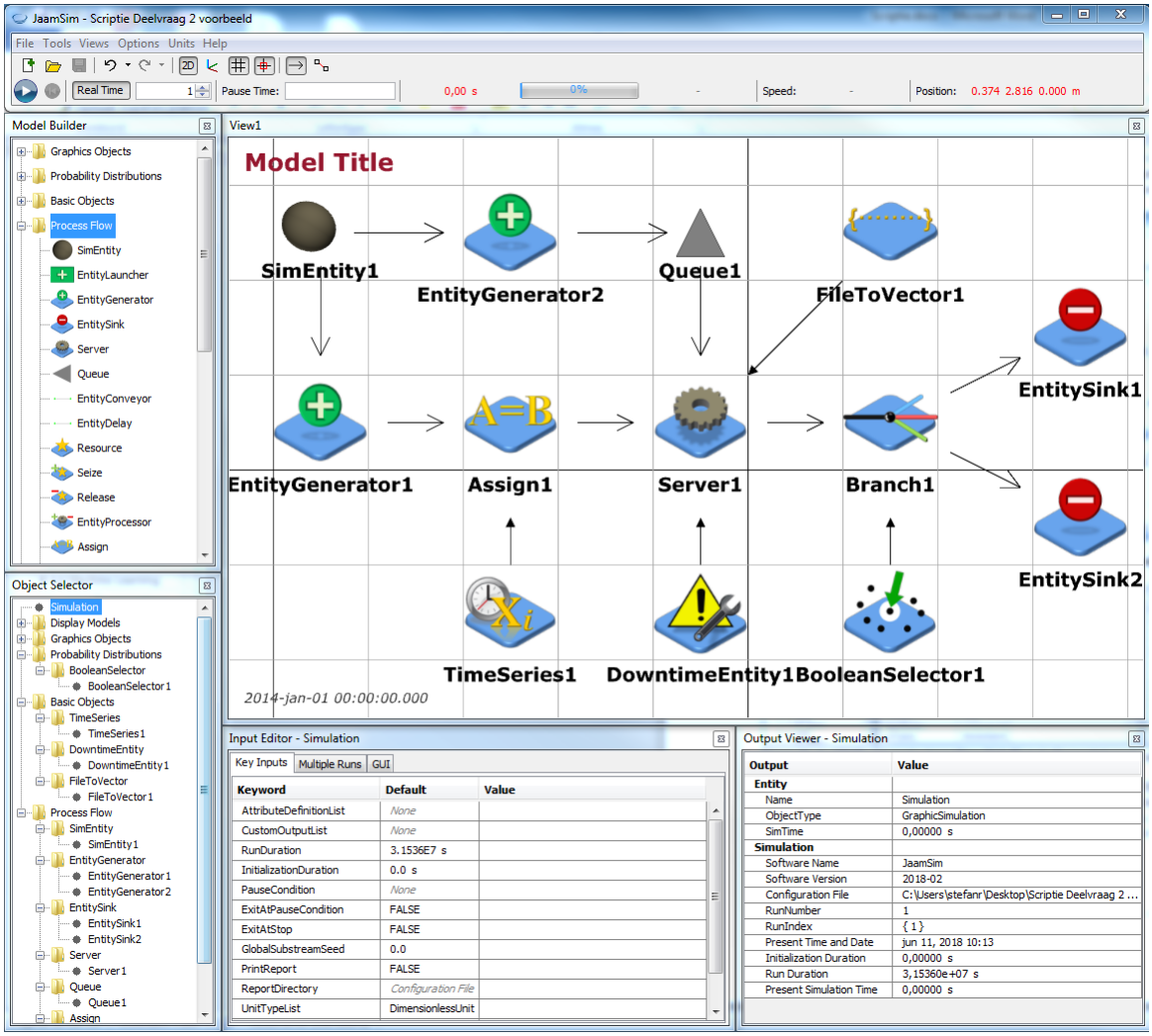
Het inladen van externe data is nodig om de data uit de EMI- en VisualFlow database te kunnen gebruiken.

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 via onder andere capterra.com. Na overleg met de bedrijfsbegeleider en de product owner van EMI is besloten om softwarepakket **JaamSim** (2018) te gebruiken. JaamSim voldoet aan bovenstaande functionele en niet-functionele eisen. In de komende paragrafen zal JaamSim en haar functionaliteiten worden besproken.

5.2.2 JaamSim simulatiesoftware

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 is 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 trommeltijden uit VisualFlow.

Het object ‘FileToVector1’ kan van een tekstbestand een vector maken met waardes. 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 waardes kunnen genereren aan de hand van een waarschijnlijkheidsverdeling. Dit is een formule van kansberekening. Deze ‘Probability Distribution’-objecten hebben de invoerwaardes minimum, gemiddelde en maximum nodig om de formule te berekenen. Ook deze waardes kunnen weer worden ingeladen met een ‘FileToVector’-object gevuld met OEE-data uit de EMI database.

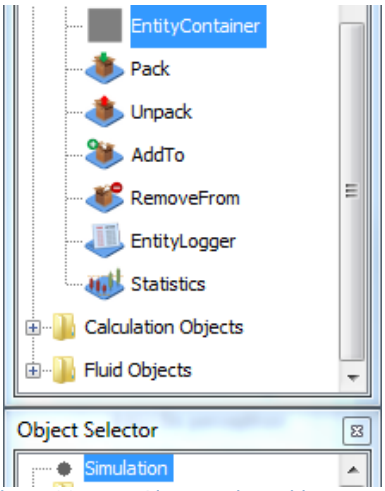
5.2.2.4 Uitval

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 persen 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 moeten 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 lastig zijn.

5.3 Machine learning algoritme afweging

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 waardes. Deze waardes 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 4 toont de resultaten van de vergelijkingen tussen een aantal GA-frameworks. De oorsprong van deze resultaten staat in Bijlage B.

Tabel 4: 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
TOTAAL	**4,5**	**4**	**3,5**	**5**	**5,5**

Het puntensysteem van Tabel 4 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 4 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 om in EMI te implementeren.

5.4 Experimenteel onderzoek

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 waardes mag de looptijd van een simulatie niet meer zijn dan: $\frac{2\ weken}{100 \cdot 100} = 120{,}96\ seconden.$

Tabel 5: 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 5 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 entities 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 entities van 1000 rollen te simuleren. De geschatte tijd om honderd generaties met een population van honderd te simuleren wordt dan: $30{,}2\ seconden \cdot 100 \cdot 100 \approx 3{,}5\ dag.$

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 trommeltijd is wel uit de data van Visual Flow berekend en is ongeveer 7 uur en 23,5 minuut.

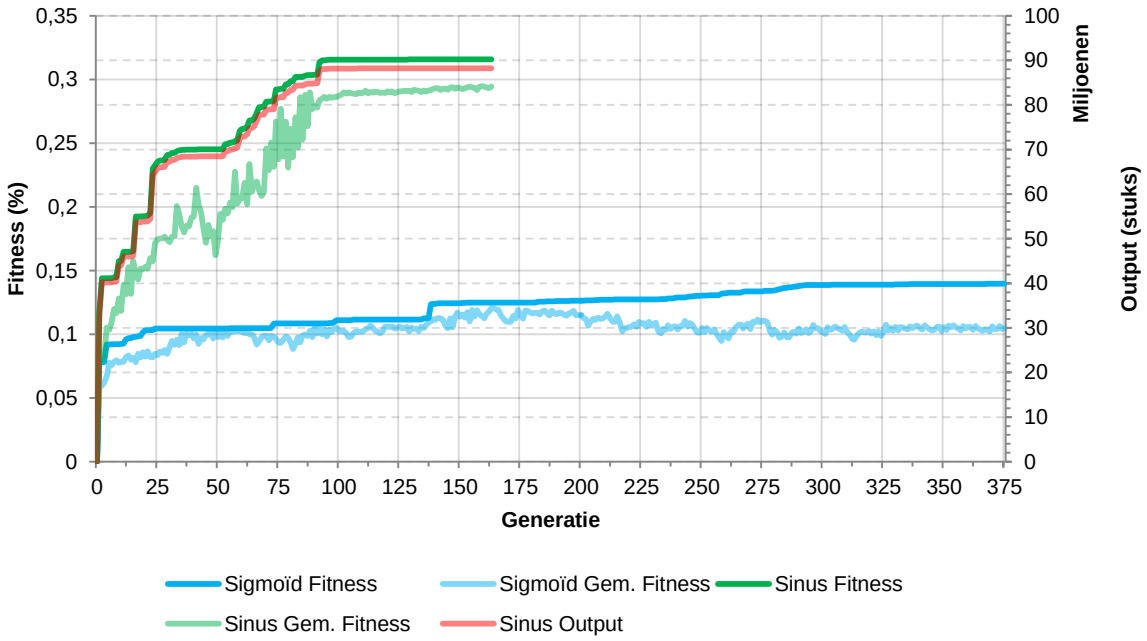
SharpNEAT bevat een aantal verschillende activation functions zoals de sigmoïd, ReLU, leaky ReLU, hyperbolische tangens en sinus. Zoals beschreven in hoofdstuk 4.2.3 wordt de ReLU activation function steeds meer gebruikt. Een nadeel is dat de ReLU niet goed om kan gaan met grote getallen die in dit geval 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: $max.buffervoorraadgrootte = \frac{max.totale\ output}{productielijnen} \cdot OEE$. 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 6 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 6: 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



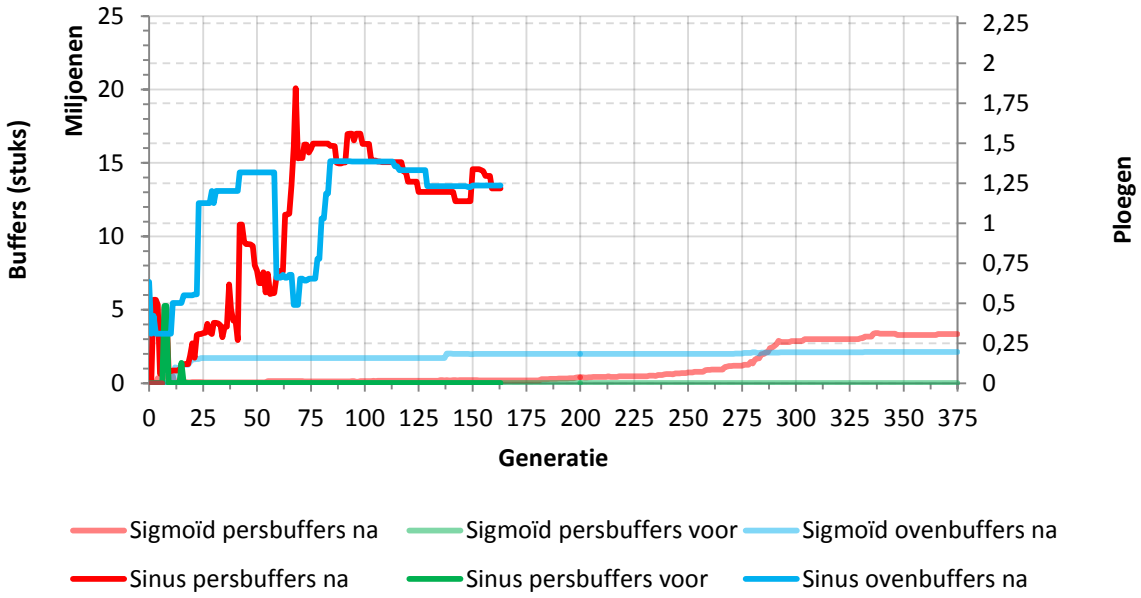
Figuur 21: Simulatie resultaten sigmoïd- en sinus activation functions

De fitness is berekend volgens de formule:
$fitness(\%) = \frac{output}{max.output} \cdot 50\% + (1 - \frac{buffers}{max.buffers}) \cdot 50\%.$
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 entities 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 fabrieksoutput van TN in februari was ongeveer 95,9 miljoen stuks. De simulatie met de sinus activation function haalde een fabrieksoutput 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 fabrieksoutput van de simulatie ongeveer 91,9% van de daadwerkelijke fabrieksoutput.

In Figuur 22 is te zien dat de bijbehorende buffervoorraadgroottes net als de fitness en fabrieksoutput 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. In generatie 57 is er een mutatie geweest die zorgde dat een van de buffergroottes van de ovens die onnodig hoog was met ongeveer 6 miljoen stuks daalde, waardoor zowel de output als de fitness steeg.



Figuur 22: Simulatie buffervoorraadgroottes sigmoïd- en sinus activation functions

De hoogst behaalde output zonder buffer-data uit VisualFlow bedraagt ongeveer 88,2 miljoen stuks. In werkelijkheid produceerde TN in februari ongeveer 95,9 miljoen stuks. De hypothese voor de output met buffer-data uit VisualFlow is: $output \approx 88.2\ mil. + 24.4\ mil.\ \cdot\ 50\% \approx 100\ mil.$. De totale buffervoorraad van 1 februari was volgens de VisualFlow database ongeveer 24,4 miljoen stuks (zie Bijlage C voor de volledige datasets). Niet alle buffervoorraden hebben de tijd om binnen een maand bij de output terecht te komen en een gedeelte zal ook uitval worden. Daarom is de weegfactor van 50% geschat. De hypothese voor deze output ligt ongeveer 4,1 miljoen stuks hoger dan de werkelijke productie van TN over februari.

5.4.3 Simulaties met VisualFlow buffer-data

TODO!!!

5.4.4 Conclusie

TODO!!!

6 EMI IMPLEMENTATIE

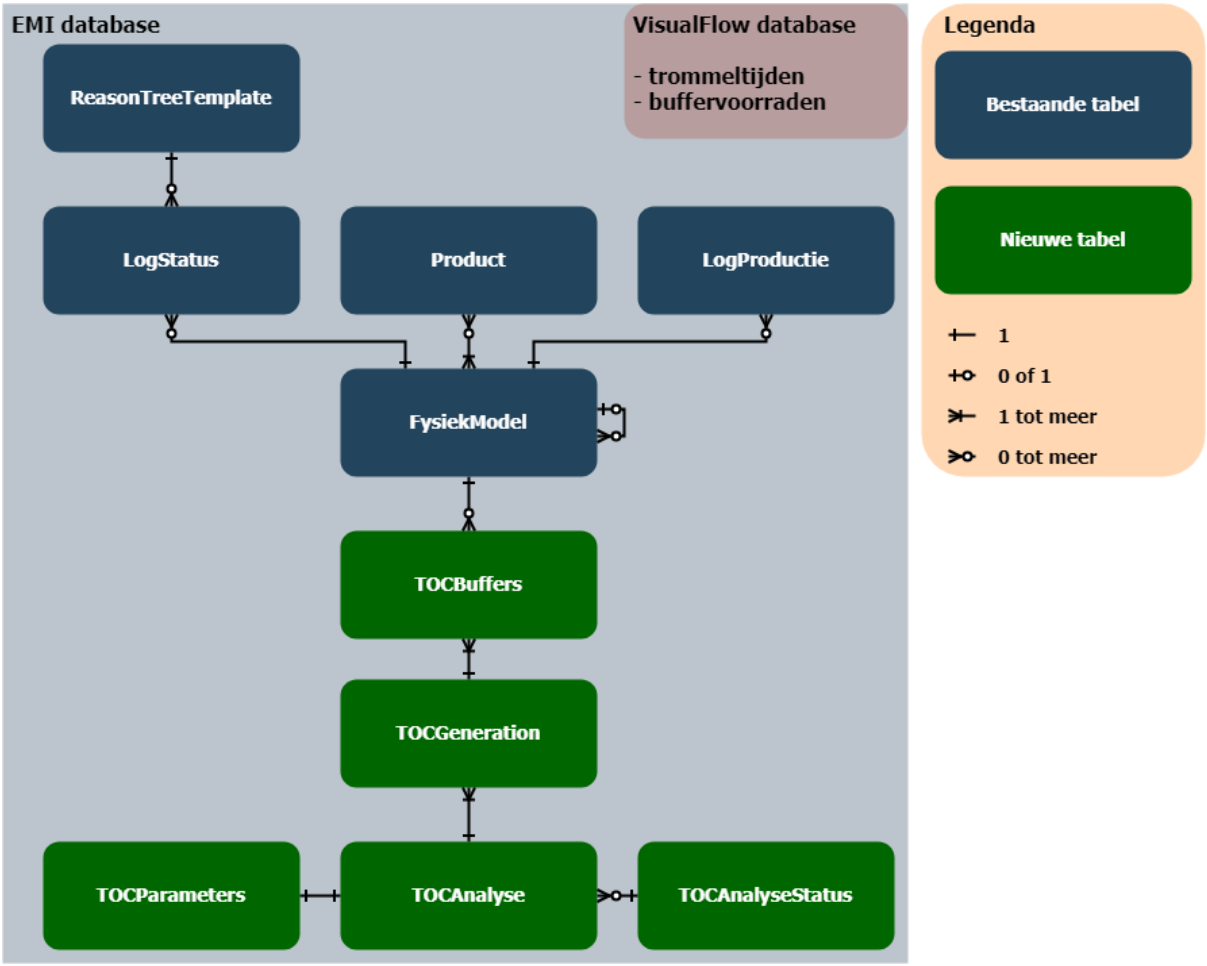
In dit hoofdstuk wordt de implementatie van de programmatuur in EMI beschreven. Deze implementatie bevat de koppeling tussen EMI, SharpNEAT en JaamSim.

6.1 Frontend

TODO!!!

6.2 EMI database

In de eerste deelvraag van dit onderzoek zijn de VisualFlow- en EMI databases onderzocht. In Figuur 23 is het resultaat hiervan terug te zien met daarbij de toevoegingen van de implementatie.

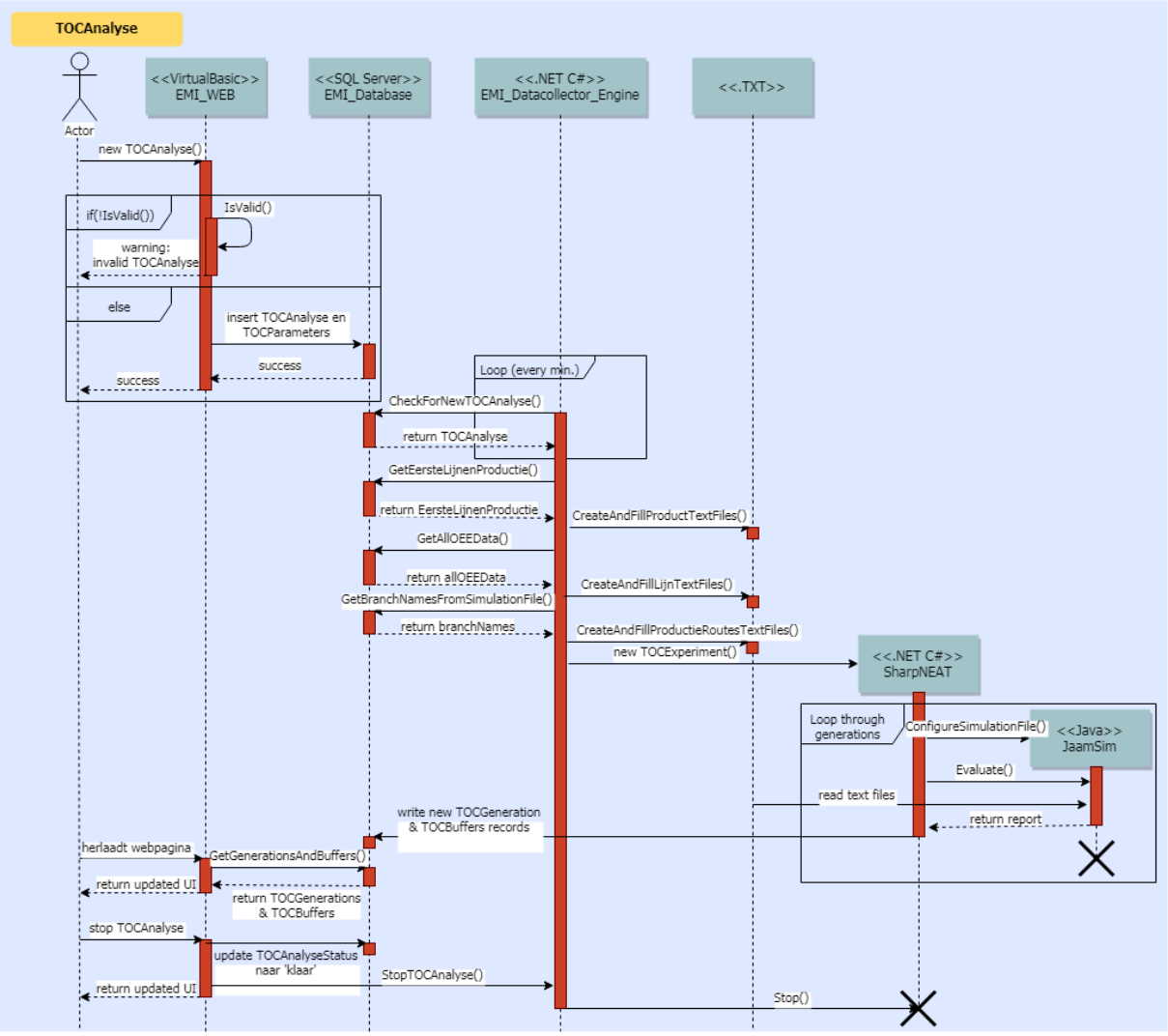


Figuur 23: EMI Entity Relationship Diagram (ERD)

Wanneer de gebruiker via de website van EMI een nieuwe analyse aanmaakt, wordt er een record aan de 'TOCAnalyse'-tabel toegevoegd met daarbij een 'aangemaakt'-status en de door de gebruiker meegegeven parameters zoals de start- en einddatum van de analyse. De generaties met daarbij behorende buffervoorraadgroottes worden gekoppeld aan dit analyse record gedurende de simulaties. De nieuwe tabellen beginnen met 'TOC'. Dit is een overblijfsel uit de beginperiode van dit onderzoek waarbij nog niet duidelijk was dat de TOC een minder grote rol zou gaan spelen dan verwacht.

6.3 Backend

In Figuur 24 wordt een samenvattend beeld van de EMI backend weergegeven. Hierin wordt duidelijk hoe de communicatie tussen de front-end en backend van EMI verloopt en de communicatie tussen de EMI backend en de geïmplementeerde applicaties SharpNEAT en JaamSim.

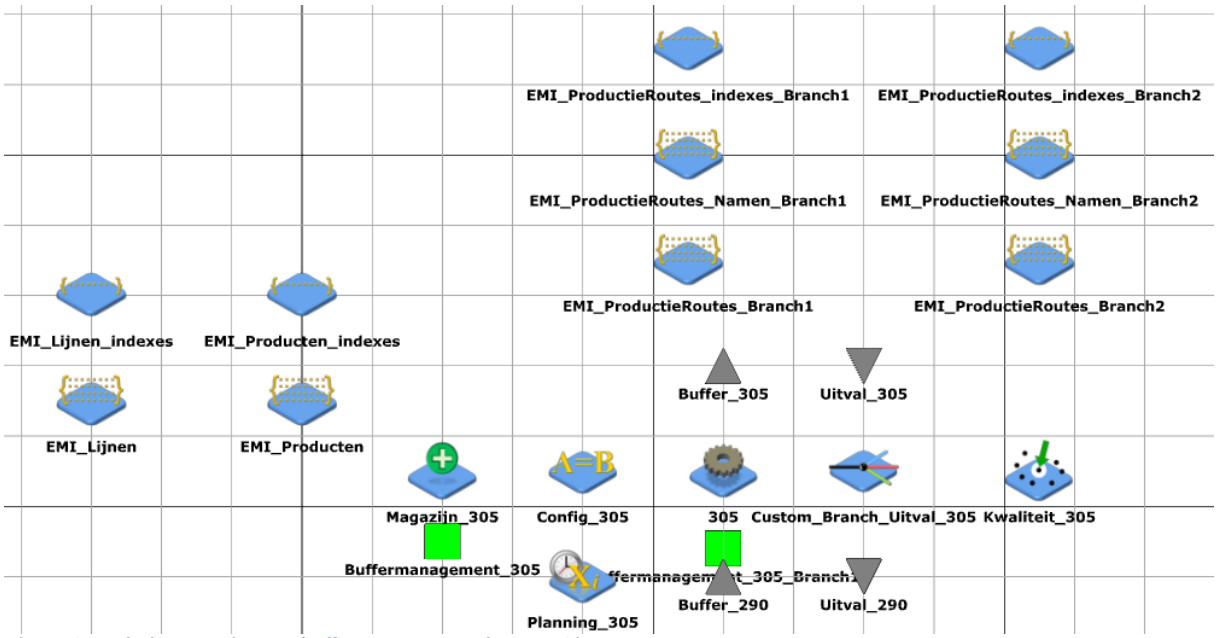


Figuur 24: EMI sequence diagram

Wanneer de gebruiker via EMI_WEB een nieuwe analyse start wordt deze aangemaakt in de database van EMI. De EMI_Datacollector_Engine die op de EKB-server draait checkt elke minuut of er een nieuwe analyse in de database aanwezig is en start deze. De tekstbestanden met de data voor de simulaties worden overschreven en bestaan al voordat de analyse aangemaakt wordt. Elk lid van de population wordt gesimuleerd waarna JaamSim een rapport van de simulatie aanmaakt die SharpNEAT kan interpreteren om de fitness, output en buffers te berekenen.
Als de gebruiker de webpagina herlaadt worden de resultaten van de analyse uit de database opgehaald en weergegeven in de browser. Ook kan de gebruiker de analyse stop zetten zodat er een andere analyse gestart kan worden.

6.4 JaamSim

De basis functionaliteiten die in de JaamSim simulaties zijn geïmplementeerd zijn onderzocht en beschreven in hoofdstuk 5.2.2. Deze functionaliteiten zijn ook terug te zien in een gedeelte van het simulatie-bestand in Figuur 25.



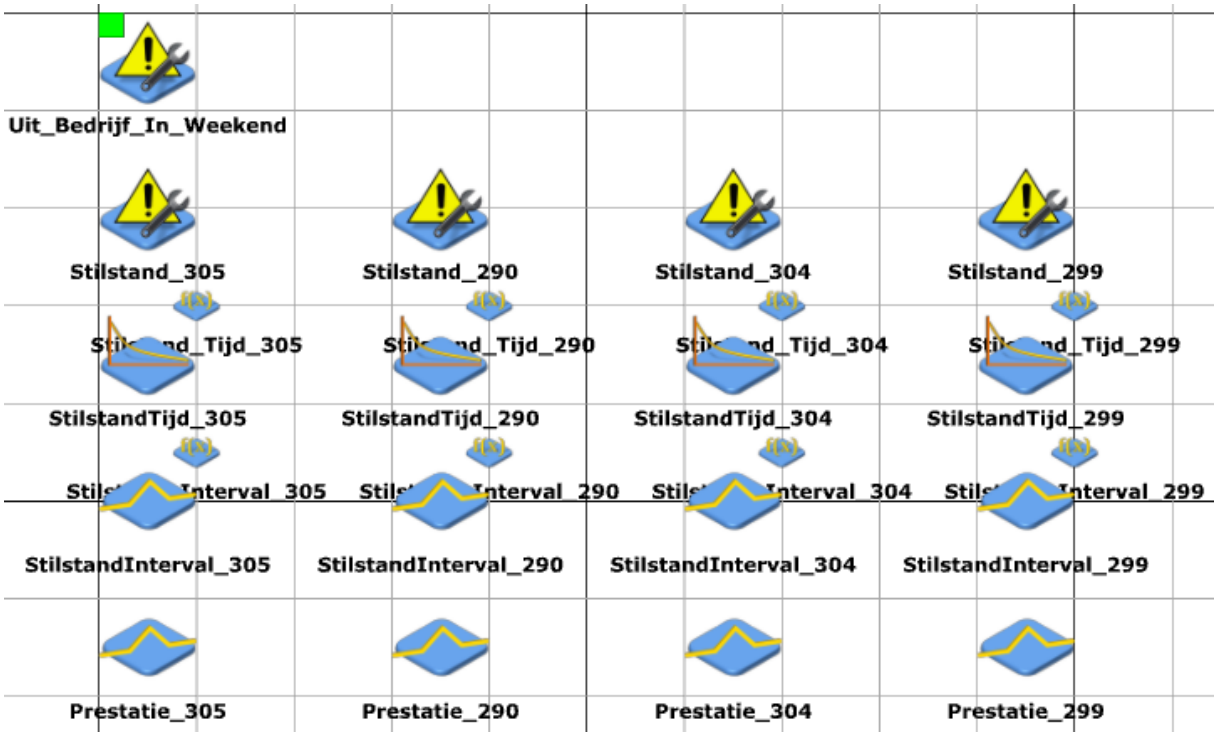
Figuur 25: Inladen van data en buffermanagement in JaamSim
Noot. Rang, S. A. (2018, 26 juni). Inladen van data en buffermanagement in JaamSim [Foto], Geraadpleegd van 'JaamSim (2018-03)'

Er worden in de simulatie zowel 'FileToVector'-objecten als 'FileToMatrix'-objecten gebruikt om de data uit de door de EMI_Datacollector_Engine gegenereerde tekstbestanden uit te lezen. De tekstbestanden bevatten de OEE-data, productiesnelheden, productgewichten en de data over welke productielijnen welke producten kunnen produceren.

De objecten die in de simulatie een nummer in de naam hebben komen overeen met het ID die de desbetreffende productielijn in de EMI database heeft. Op deze manier kan de data over de productielijnen automatisch ingeladen worden.

De groene vierkanten zijn objecten die aan de hand van een expressie het object erboven aan of uit zetten. Een voorbeeld van de expressie van het object 'Buffermanagement_305': '[Buffer_305].QueueLength < 1'. Deze expressie houdt in dat het object 'Magazijn_305', waar het buffermanagement-object bij hoort, pas nieuwe producten naar de volgende productielijn stuurt als de buffervoorraadgrootte van productielijn '305' lager is dan één entity. In dit geval zijn dat 1000 rollen zoals vastgesteld in hoofdstuk 5.4.1.

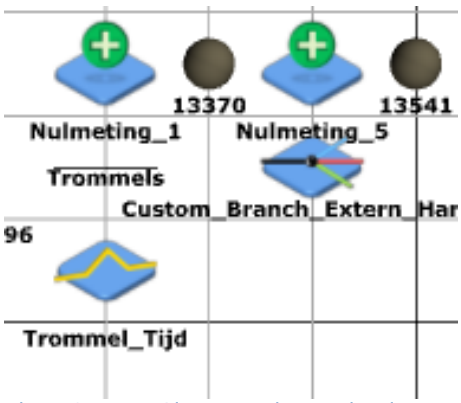
De OEE van de productielijnen wordt berekend op basis van de OEE-data uit de EMI database die via de tekstbestanden in JaamSim worden ingelezen. In Figuur 26 is te zien dat de stilstanden willekeurig worden gesimuleerd door middel van 'Probability Distribution'-objecten zoals ook omschreven in hoofdstuk 5.2.2.3. De interval tussen de stilstanden en de prestatiepercentages van de productielijnen worden berekend met een 'Triangular Distribution'. Dit is een manier om op basis van een minimum-, maximum en gemiddelde waarde willekeurige waardes te genereren. De tijdsduur van de stilstanden zijn veel vaker kort dan lang in het geval van TN. Vandaar dat voor deze simulatie een exponentieel object is gekozen voor de willekeurige stilstandtijden.
Het 'Uit_Bedrijf_In_Weekend'-object simuleert het feit dat de perserij en slijperij in de weekenden uit staan.



Figuur 26: JaamSim OEE-objecten
Noot. Rang, S. A. (2018, 26 juni). JaamSim OEE-objecten [Foto], Geraadpleegd van 'JaamSim (2018-03)'

De twintig trommels worden in JaamSim niet apart gesimuleerd (Figuur 27). De trommels vormen namelijk geen probleem voor TN. De minimale-, gemiddelde- en maximale trommeltijd die uit VisualFlow berekend is wordt ook met een 'Triangular Distribution'-object gesimuleerd. Het trommel-object is geen 'Server'-object zoals de andere productielijnen in JaamSim, maar een 'Entity Delay'-object. Dit houdt in dat de producten met een bepaalde vertraging, de trommeltijd, over deze lijn lopen richting de volgende productielijnen.
Ook het extern harden van sommige producten van TN wordt gesimuleerd door middel van deze objecten.

De buffervoorraden die aan het begin van de simulatieperiode aanwezig zijn worden toegevoegd door de 'Entity Generator'-objecten met 'Nulmeting' in de naam. De zwarte bollen ernaast zijn de verschillende ProductID's uit de EMI database. In de VisualFlow database waar deze buffervoorraden vandaan komen worden namelijk ongeveer dezelfde productnamen gebruikt als in de EMI database waardoor de juiste ProductID's opgezocht konden worden.



Figuur 27: JaamSim trommels en nulmetingen
Noot. Rang, S. A. (2018, 26 juni). JaamSim trommels en nulmetingen [Foto], Geraadpleegd van 'JaamSim (2018-03)'

7 CONCLUSIE

Dit hoofdstuk geeft een antwoord op de hoofdvraag van dit onderzoek:

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

Uit onderzoek naar de benodigde data is gebleken dat het merendeel van deze data in de database van EMI terug te vinden is. Ook is gebleken dat de overige data uit de VisualFlow database van TN gehaald kan worden.

Om het algoritme te kunnen trainen op de data zijn er meerdere simulatie softwarepakketten vergeleken op functionele- en niet-functionele eisen. Uiteindelijk is in overleg de simulatiesoftware JaamSim gekozen welke voldoet aan de gestelde eisen.

Verder is er een analyse van de verschillende soorten machine learning algoritmes uitgevoerd waaruit naar voren is gekomen dat GA's het meest geschikt zijn vanwege de mogelijkheid om problemen op te lossen waarvan de oplossing onbekend is.

Uit een vergelijking tussen verschillende GA-frameworks is gebleken dat het SharpNEAT-framework het meest geschikt is om in EMI te implementeren.

TODO!!! (Eindconclusie - Deelvraag 4)

TODO!!! (Eindconclusie)

8 AANBEVELINGEN

TODO!!!

9 EVALUATIE

TODO!!!

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

Accord. (2018, 27 maart). Accord GitHub code. Geraadpleegd van http://accord-framework.net/docs/html/R_Project_Accord_NET.htm

AForge. (2012). License. Geraadpleegd van http://www.aforgenet.com/framework/license.html

AForge.NET docs. (2013). License. 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/

Apache. (2018). APACHE LICENSE, VERSION 2.0 (CURRENT). Geraadpleegd van http://www.apache.org/licenses/

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

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.pdfcog3CS-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. (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). API Documentation. Geraadpleegd van http://numl.net/api/index.html

Juarez, S. (2017, 7 september). NUML GitHub code. Geraadpleegd van https://github.com/sethjuarez/numl

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

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

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

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

Rang, S. A. (2018, 26 juni). Inladen van data en buffermanagement in JaamSim [Foto]. Geraadpleegd van JaamSim (2018-03)

Rang, S. A. (2018, 26 juni). JaamSim OEE-objecten [Foto]. Geraadpleegd van JaamSim (2018-03)

Rang, S. A. (2018, 26 juni). JaamSim trommels en nulmetingen [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-inteligence/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

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

Van Otterlo, M., & Wiering, M. (2009). Reinforcement Learning and Markov Decision Processes. Geraadpleegd van http://www.ai.rug.nl/~mwiering/Intro_RLBOOK.pdf

BIJLAGEN

Bijlage A: Plan van Aanpak

Pagina's 46-72.

Vertrouwelijk: EMI buffer management

NAAM	STUDENTNUMMER	OPDRACHTGEVER	FUNCTIE	DATUM
S. A. Rang	1655299	EKB Houten	Afstudeerder	23-4-2018

Sjabloon: V4.0

INHOUDSOPGAVE

BEGRIPPENLIJST

BEGRIP	DEFINITIE
Bottleneck	De zwakste machine of productielijn van een fabriek met de laagste capaciteit waardoor het aantal geproduceerde producten van de hele fabriek gelimiteerd wordt
Buffer	De voorraad van producten of halffabricaten die staat te wachten tussen twee productielijnen of machines tot het verder verwerkt kan worden
Capaciteit	Het aantal producten dat een productielijn of machine in 24 uur kan produceren
Deep Learning	Machine learning met meerdere serie geschakelde hidden layers
Efficiëntie	Een percentage van de maximale snelheid van een productielijn of machine
EKB Manufacturing Intelligence	Een industrieel webbased automatiserings software pakket voor verzameling en visualisatie van realtime informatie over productielijnen
KPI	Key Performance Indicator, een variabele om de prestatie van, in dit geval, een productielijn of machine te meten
Machine Learning	Het verwerken van data d.m.v. een algoritme dat niet zelf geprogrammeerd is, maar d.m.v. supervised, unsupervised learning of een genetic algorithm wordt gegenereerd
Normaliseren	Het standaardiseren van numerieke gegevens voor een machine learning algoritme, meestal tussen de -1 en 1 zodat het algoritme makkelijker om kan gaan met grote cijfers.
Ploeg	Een ploeg betekent dat een bepaalde hoeveelheid van een type product er 8 uur over doet om in de slijperij verwerkt te worden. Bijv. 500kg van product A is 1 ploeg en duurt 8 uur in de slijperij, maar 500kg van product B is 2 ploegen en duurt 16 uur in de slijperij.
Productielijn	Een serie geschakelde verzameling machines waarmee in een fabriek producten worden geproduceerd
Programmable Logic Controller	Een apparaat die commando`s van buitenaf via een microprocessor omzet naar acties op productielijnen of machines
Redmine	Een open source management applicatie die gebruikt zal worden voor SCRUM
Theory of Constraints	Een maatschappelijk geaccepteerde manier om de bottleneck van een systeem te vinden en deze te exploiteren totdat een ander gedeelte van het systeem de bottleneck wordt. Dan wordt het proces herhaalt.

1 INLEIDING

In dit document worden de belangrijkste punten van deze afstudeeropdracht beschreven. Een aantal belangrijke begrippen die bij deze afstudeerstage van toepassing zijn; Machine learning, Theory of Constraints en buffers. Deze zullen in het vervolg van dit document verder worden toegelicht.

De afstudeerstage vindt plaats bij EKB Houten, een industrieel automatiseringsbedrijf dat processen binnen deze branche implementeert. De student zal op de development afdeling van EKB Houten onderzoek doen naar de huidige situatie bij Tsubaki Nakashima (TN), een klant van EKB, en de verschillende machine learning algoritmes. Daaruit voortvloeiend zal een eindproduct of proof of concept opgeleverd worden waarin het beste algoritme is geïmplementeerd om de grote hoeveelheid onoverzichtelijke data van verschillende productielijnen van TN te verwerken tot overzichtelijke, bruikbare data.

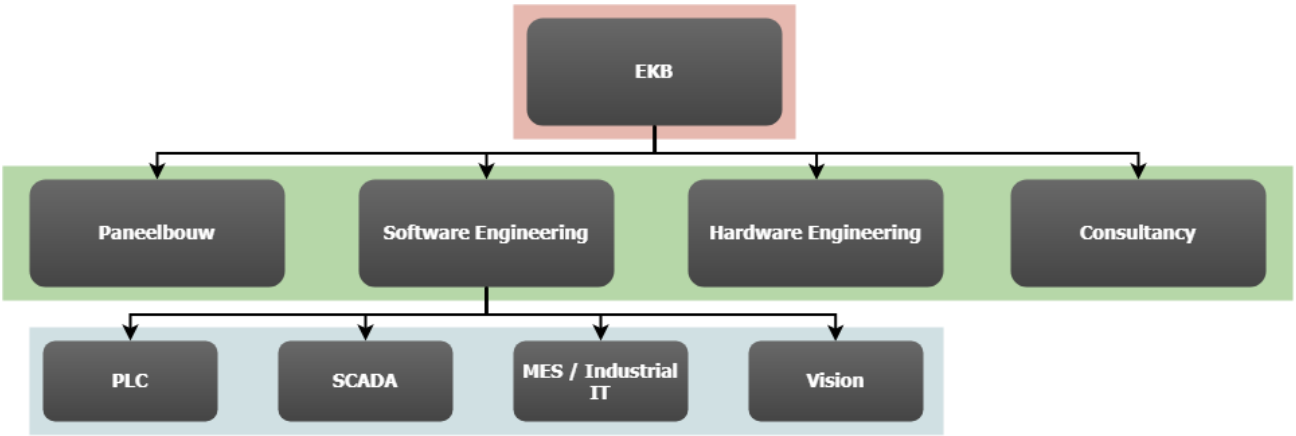
Het document is als volgt ingedeeld: Er wordt begonnen met wat achtergrondinformatie over het bedrijf en de betrokkenen van de afstudeeropdracht. Vervolgens wordt de afstudeeropdracht omschreven zoals deze is voortgekomen uit de gesprekken tussen afstudeerder, bedrijfsbegeleider en product owner. Hierna wordt beschreven hoe het onderzoek van de afstudeeropdracht eruit zal gaan zien. Ten slotte worden de eventuele risico’s van deze afstudeeropdracht omschreven en worden er mogelijke oplossingen gegeven mochten deze risico’s werkelijkheid worden.

2 CONTEXT

2.1 Het bedrijf

2.1.1 EKB Houten

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. De belangrijkste activiteiten zijn weergegeven in Figuur 1. De student zal komen te werken op de MES (Manufacturing Execution Systems) afdeling van software engineering. Andere activiteiten van EKB zijn paneelbouw, hardware engineering en consultancy.



Figuur 1: EKB activiteiten

2.1.2 Bedrijfsgegevens

Tabel 1: Bedrijfsgegevens van EKB

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

2.1.3 Persoonsgegevens

Tabel 2: Persoonsgegevens van betrokkenen

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

Frank Verbruggen is docent bij de Hogeschool Utrecht en de docentbegeleider voor deze afstudeeropdracht.

Auke Roelofsen is projectmanager bij EKB Houten en tevens bedrijfsbegeleider voor deze afstudeeropdracht. De heer Roelofsen heeft gestudeerd aan zowel de Hogeschool Saxion als de Wageningen University & Research en heeft mijns inziens genoeg kennis en ervaring om de benodigde begeleiding te bieden op HBO niveau.

Michiel de Lange is directeur van EKB Houten. De heer De Lange zal bij deze afstudeeropdracht fungeren als Product Owner.

Ger Bargeman is LEAN Manager bij Tsubaki Nakashima in Veenendaal. De heer Bargeman is de contactpersoon voor Tsubaki Nakashima waar deze afstudeeropdracht voor uitgevoerd zal voeren in opdracht van EKB Houten.

Maarten Kok is een .NET software engineer op de development afdeling van EKB Houten en heeft kennis over PLC, EMI en machine learning wat van waarde kan zijn voor deze afstudeerstage. De heer Kok heeft dan ook aangeboden deze kennis te delen om te helpen met de vooruitgang van de afstudeeropdracht.

2.2 Positie van de student binnen EKB

De student zal bij de EKB vestiging in Houten als afstuderend developer op de development afdeling de volgende taken uitvoeren:

- Onderzoek doen
- Front-end programmeren
- Back-end programmeren
- Proof of concept/eindproduct realiseren

De student zal te allen tijde verantwoordelijk zijn voor eigen documenten, onderzoek, rapportages, code en eindproduct(en).

2.3 Relaties

2.3.1 Relatie afstudeeropdracht met SIE

In deze afstudeeropdracht komen de volgende vakken uit de Propedeuse en Hoofdfase van SIE terug:

- Human Computer Interaction (HCI)
- Patterns and Frameworks
- Programmeren in .NET
- Web Application Construction (WAC)
- Object Oriented Programming (OOP)
- Professional Skills

HCI komt terug in het user experience gedeelte van de front-end van het proof of concept/eindproduct. Om de grote hoeveelheden data overzichtelijk en begrijpelijk te houden dient er een goede user experience te zijn. Daarnaast wordt er voor de front-end net als bij WAC, HTML5 en CSS gebruikt en wordt de software voorzien van versiebeheer. Voor het programmeren van de back-end wordt .NET C# gebruikt waarbij object georiënteerd geprogrammeerd zal worden en er gebruikt zal worden gemaakt van verschillende patterns. Ten slotte worden de documenten geschreven volgens de standaarden van het vak professional skills.

2.3.2 Relaties met andere projecten

Het proof of concept/eindproduct(en) moet generiek, herbruikbaar en uitbreidbaar zijn. Dit betekent ook dat het proof of concept/eindproduct(en) door verschillende klanten van EKB met andere projecten gebruikt zou kunnen worden, maar in eerste instantie gaat de afstudeeropdracht over een enkel project van een klant van EKB.

3 DE OPDRACHT

3.1 De afstudeeropdracht

3.1.1 De kwestie

Sinds 2009 ontwikkelt EKB een eigen softwarepakket genaamd EMI (EKB Manufacturing Intelligence), 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 vooral 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 wordt toegepast.

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

3.1.2 Theory Of Constraints

Een beknopte definitie van TOC volgens Procesverbeteren.nl:

> #### TOC definitie Procesverbeteren.nl
> **Theory of Constraints** (TOC) is net als Lean een (logistieke) verbetermethode die streeft naar meer *flow*, dus kortere doorlooptijden. Dit gebeurt bij de TOC echter nóg nadrukkelijker vanuit de gedachte dat alleen producten die de klant bereiken van belang zijn: *Profit = Throughput – Operating Expenses*. Een ander verschil is dat Lean uitgaat van stapsgewijs verbeteren door iedereen, terwijl de TOC bovenal grote doorbraken nastreeft en daardoor een ‘strategischer’ karakter heeft.
>
> De TOC probeert steeds het knelpunt dat de afzet *op dat moment* het meeste beperkt op te sporen, en dit vervolgens maximaal uit te nutten (exploitatie) of zelfs op te heffen. Om een knelpunt te ‘exploiteren’ kunnen zowel TOC-, Lean- als QRM-tools worden ingezet. Het ‘opheffen’ van een knelpunt gebeurt niet altijd fysiek, het kan ook gaan om een oplossing voor een schijnbaar conflict, dat dan ‘verdampt’.
>
> Knelpunten en oplossingen daarvoor vind je met behulp van de TOC *thinking processes*. Hierbij draait alles om het in kaart brengen van oorzaken en gevolgen, om “verborgen” kansen te vinden. Winst maken is binnen de TOC géén toeval, de mogelijkheden zijn schier *ongelimiteerd*. Kenmerkend is systeemdenken: een *supply chain* is zo sterk als de zwakste schakel, de *bottleneck*.
>
> Bovenstaande definitie is tot stand gekomen op basis van onze honderden case-beschrijvingen, en vergelijking met de definities die andere TOC-denkers momenteel hanteren. Zie de Lean definitie voor overeenkomsten met Lean, en de QRM definitie voor overeenkomsten met QRM. (Van Ede, 2017)

3.1.3 De afstudeeropdracht in het kort

Het verlagen van 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 TOC, machine learning en de huidige situatie bij TN (Tsubaki Nakashima) welke EKB van de student eist, dient de student een werkend product op te leveren waarin daadwerkelijk TOC 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 machine learning worden gebruikt om een algoritme tot stand te laten komen die op basis van een fabriekssimulatie de variabelen af kan wegen. Om dit te kunnen implementeren in EMI moet echter wel eerst worden onderzocht hoe de simulatie zo realistisch mogelijk te maken is terwijl het wel generiek moet blijven zodat ook andere klanten van EKB deze simulatie kunnen gebruiken om een algoritme te genereren.

3.1.4 Projectgrenzen en randvoorwaarden

Ondanks dat er tientallen scenario`s te bedenken zijn waarin machine learning zou kunnen worden gebruikt om het productieproces van TN te verbeteren kan er maar een gekozen worden voor deze afstudeeropdracht. Hierdoor kan de implementatie binnen de tijd en precies worden uitgevoerd. Ook is het dan makkelijker de implementatie herbruikbaar en generiek te houden.

TN heeft aangegeven dat de productieplanning op basis van klantorders nog gedeeltelijk met de hand wordt gedaan. Wel wordt er al een voorstel gedaan om voor bepaalde deadlines orders te produceren, maar het definitief inplannen van orders op de verschillende productielijnen wordt met de hand gedaan. Na enig onderzoek naar deze werkwijze is samen met de heer Bargeman en de heer Roelofsen besloten dit niet mee te nemen in de afstudeeropdracht omdat het lastig is om ‘menselijke’ beslissingen op basis van jarenlange ervaring aan een machine learning algoritme over te laten. Dit ook met oog op de vele uitzonderingen en aparte situaties die zorgen voor een andere productieplanning.

Een ander probleem van TN is slecht gereedschapsmanagement. In de huidige situatie worden de gereedschappen van machines op vaste tijden verwisseld. Hierdoor staan de machines tijdelijk stil en worden ook gereedschappen vervangen die nog niet vervangen hoeven te worden. Er worden op een van de productielijnen metingen gedaan waarmee een machine learning algoritme een voorspelling zou kunnen doen wanneer het gereedschap vervangen moet worden. Dit probleem valt echter buiten de scope van deze afstudeeropdracht.

Deze afstudeeropdracht zal gaan over het tunen van buffervoorraden. Oftewel, het verbeteren van de productie flow of doorstroom (Throughput). Dit zal worden gerealiseerd met een of meerdere machine learning algoritmes die generiek in EMI geïmplementeerd worden. Hierbij wordt zoveel mogelijk bestaande data van EMI gebruikt om te zorgen dat er geen externe data nodig zijn om de functionaliteiten bij een nieuwe klant te gebruiken.

3.1.5 Doelstelling van de afstudeeropdracht

De te bouwen uitbreiding van EMI moet ervoor gaan zorgen dat klanten van EKB hun buffervoorraden zo laag mogelijk kunnen houden terwijl de productie gelijk blijft en beter nog hoger wordt. Eventueel zou de software ook kunnen zorgen voor een minimale hoeveelheid omstellingen van machines en het gebruiken van minder productielijnen om dezelfde hoeveelheid productie te bereiken.

Door middel van de beschikbare data in EMI over de productielijnen moet het mogelijk worden de doorlooptijden van halffabricaten te verminderen. Hierdoor worden de voorraden tussen de verschillende machines en productielijnen kleiner en verminderen de kosten per product.
De data over deze productielijnen die in EMI beschikbaar zijn, zijn gestandaardiseerd. Hierdoor blijft de nieuwe software generiek en kan het ingezet worden voor alle klanten van EKB die voldoende data opslaan in EMI.

3.1.6 Op te leveren producten

EKB verwacht aan het einde van de afstudeerstage ten minste een proof of concept/eindproduct in EMI. In Tabel 3 is de MoSCoW analyse te zien die voortgekomen is uit het eerste contact met de bedrijfsbegeleider.

Tabel 3: MoSCoW analyse van op te leveren producten

FUNCTIONALITEIT	MOSCOW
Machine learning algoritme voor het tunen van buffers rekening houdend met buffer kosten	Must Have
Generieke en uitbreidbare implementatie in EMI zodat met minimale aanpassing ook andere klanten gebruik kunnen maken van de functionaliteiten	Should Have
Visualisatie van de buffervoorraden van de eindklant	Should Have
Visualisatie van de toe-en afname van de verschillende buffers binnen een productielocatie	Should Have
Bottleneck rapportage	Could Have
Machine learning algoritme voor productieplanning	Would Have
Machine learning algoritme voor gereedschapsmanagement	Would Have

Een belangrijk punt voor EKB is dat de implementatie uitbreidbaar en generiek is. Echter is dit een Should Have, omdat het eindproduct ook kan functioneren zonder dat het uitbreidbaar en generiek is. Het plannen van de verschillende soorten producten die in een fabriek worden geproduceerd zou volgens de bedrijfsbegeleider ook een goede functionaliteit voor een algoritme zijn. Uiteindelijk is besloten dat dit de afstudeeropdracht te breed maakt en eigenlijk uit twee opdrachten zou gaan bestaan. Hierdoor is dit een Would Have functionaliteit.

3.2 Vertrouwelijkheid

EKB heeft duidelijk gemaakt dat zowel het werk bij TN, als het eindproduct/proof of concept vertrouwelijk is en alleen gedeeld mag worden met EKB, TN en de Hogeschool Utrecht. Daarbij moet vermeld worden dat het delen met de Hogeschool Utrecht alleen is bedoeld voor deze afstudeeropdracht en niet voor openbare doeleinden die de Hogeschool Utrecht zou kunnen hebben.

4 HET ONDERZOEK

4.1 Hoofdvraag en deelvragen

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

Hoofdvraag decompositie
Om een machine learning 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 machine learning algoritmes in de praktijk toe te kunnen passen moet de training gericht zijn op de werkelijkheid. Om dit te bereiken is er aanvullend onderzoek nodig naar de huidige situatie bij Tsubaki Nakashima om een realistische simulatie te kunnen maken waarin de machine learning algoritmes op de datasets kunnen trainen.
Voordat de simulaties met de verschillende algoritmes van start kunnen gaan is er nog de vraag hoe nu precies Theory of Constraints toe te passen? Hoe hebben andere bedrijven en afstudeerders de buffervoorraden proberen te verminderen met TOC? Met deze kennis kunnen de initiële parameters van de algoritmes worden ingesteld om vervolgens te kunnen simuleren. Deze simulaties resulteren in een lijst van algoritmes gesorteerd op prestaties.
Wanneer de keuze voor een bepaald algoritme is gemaakt kan deze worden geïmplementeerd in EMI. Om dit te kunnen doen moet eerst nog wel duidelijk worden welke architectuur het meest geschikt is om te kunnen implementeren.

De hieruit voortvloeiende deelvragen zijn als volgt:

- Welke data uit EMI en externe data zijn er nodig om een zo realistisch mogelijk algoritme te kunnen trainen?
- Hoe kan de training zo realistisch mogelijk worden gemaakt met de beschikbare data?
- Hoe hebben anderen met TOC de buffervoorraden verlaagd?
- Welke soorten machine learning algoritmes zijn geschikt om in combinatie met TOC toe te passen?
- Welke architectuur is het meest geschikt om de machine learning algoritmes volgens de randvoorwaarden en requirements in EMI te implementeren?

4.1.1 Onderzoeksmethoden

Per deelvraag is in Tabel 4 vastgesteld welke onderzoeksmethoden gebruikt zullen worden om tot het gewenste resultaat te komen en uiteindelijk de hoofdvraag te kunnen beantwoorden.

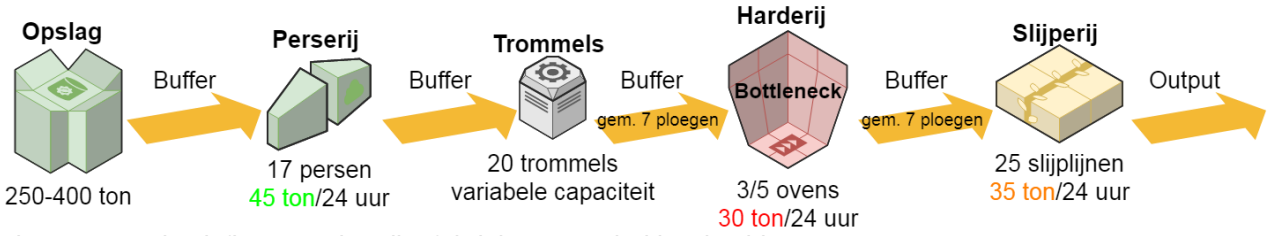
Tabel 4: Methoden matrix

DEELVRAAG	KWALITATIEF OF KWANTITATIEF?	ONDERZOEKSMETHODE	RESULTAAT
Welke data uit EMI en externe data zijn er nodig om een zo realistisch mogelijk algoritme te kunnen trainen?	Kwantitatief	Deskresearch	Verzameling van data waar het algoritme op kan trainen
Hoe kan de training zo realistisch mogelijk worden gemaakt met de beschikbare data?	Kwalitatief	Exploratief onderzoek / veldonderzoek	Een of meerdere training strategieën
Hoe hebben anderen met TOC de buffervoorraden verlaagd?	Kwalitatief	Deskresearch	Een of meerdere toepassingen van TOC ter inspiratie voor dit onderzoek
Welke soorten machine learning algoritmes zijn geschikt om in combinatie met TOC toe te passen?	Kwantitatief	Experimenteel onderzoek / laboratorium onderzoek / vergelijkend onderzoek	Een of meerdere algoritmes die gebruikt kunnen worden voor het eindproduct
Welke architectuur is het meest geschikt om de machine learning algoritmes volgens de randvoorwaarden en requirements in EMI te implementeren?	Kwalitatief	Deskresearch	Architectuur voor het eindproduct

4.2 Theoretisch kader

4.2.1 Tsubaki Nakashima

Deze afstudeeropdracht zal in eerste instantie worden uitgevoerd met de data van TN (Tsubaki Nakashima). Tijdens de implementatie zal echter wel rekening worden gehouden met het feit dat het eindproduct herbruikbaar en generiek moet zijn voor andere klanten van EKB. Omdat de algoritmes zullen leren van data van productielijnen van TN, is er tijdens het vooronderzoek informatie verzameld over de data van deze productie units en de samenhang hiertussen met interviews met contactpersoon Ger Bargeman. Hieronder volgt een samenvatting van de interviews.



Figuur 2: Productieflow van de rollenfabriek van Tsubaki Nakashima

De rollenfabriek van TN in Veenendaal maakt stalen cilindrische lagers voor o.a. auto onderdelen en hydraulische apparaten en is onderverdeeld in vijf hallen zoals te zien is in Figuur 2. In de eerste hal wordt het staal opgeslagen in hoepelvorm met een volume van gemiddeld 250 tot 400 ton. In de tweede hal wordt het staal in de juiste vorm geperst door 17 parallelle persmachines met een capaciteit van 45 ton/24 uur, maar zijn dan alleen grofweg in de juiste vorm. In de derde hal worden de halffabricaten geschuurd in 20 parallelle ronddraaiende trommels. De capaciteit van deze trommels is variabel en hangt af van zowel het type product als hoe de halffabricaten uit de perserij komen. Het is echter nog nooit gebeurd dat er te weinig trommels aanwezig waren om alle halffabricaten verder door te produceren, dus voor deze afstudeeropdracht is het niet belangrijk te veel aandacht te schenken aan de trommels. In de vierde hal worden de halffabricaten gehard door 5 parallelle ovens met een capaciteit van 30 ton/24 uur. De ovens zijn de bottleneck van de fabriek vanwege de lage capaciteit en staan daarom ook 24 uur/dag 7 dagen/week aan, maar kunnen niet elke soort product verwerken. Ook hierdoor zijn de ovens de bottleneck. Daarnaast worden er maar 3 van deze 5 ovens door EMI gemonitord en alleen van deze 3 zijn er data beschikbaar voor deze afstudeeropdracht. In de laatste hal worden de halffabricaten uiteindelijk in de juiste vorm geslepen door 25 parallelle lijnen van slijpmachines met een capaciteit van maximaal 35 ton/24 uur.

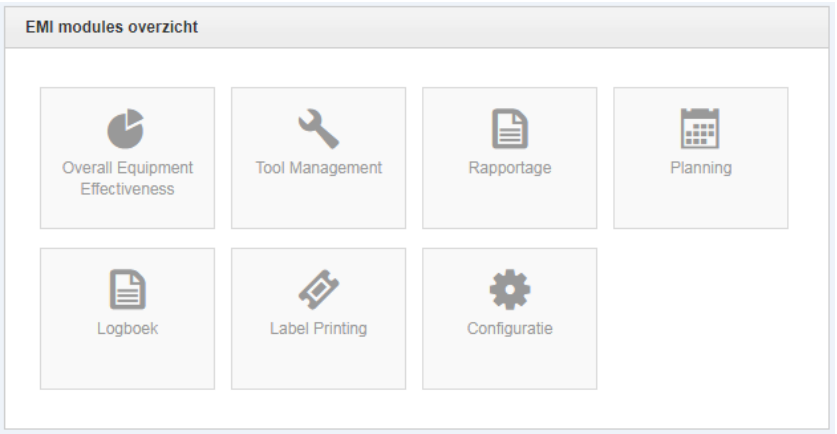
Bij TN spreken de werknemers van zogeheten ploegen. Dit is een door TN Veenendaal zelf vastgestelde maatstaf. Als bijvoorbeeld 500kg van een bepaald type product 1 ploeg is, betekent dit, dat het 8 uur duurt om deze 500kg producten door de slijperij te laten verwerken.

Tussen de hallen zijn zogenaamde buffers. Dit is opslag voor de halffabricaten die in containers worden bewaard met een track-en-trace-nummer wachtend op verwerking in de volgende hal. De buffers voor en na de ovens zijn gemiddeld 7 ploegen groot, nooit meer dan 14 en nooit minder dan 4 ploegen. Dit is zo geregeld zodat de bottleneck, oftewel de harderij, altijd kan produceren en de slijperij ook. Omdat de slijperij sneller is dan de harderij kan deze stil komen te staan als de buffer weggewerkt wordt.

Door de machines ‘om te stellen’ kunnen de machines voor verschillende producten worden gebruikt. Dit kost echter vaak uren tijd met meerdere werknemers. Het is daarom voor TN van groot belang dat het aantal omstellingen minimaal is.

4.2.2 EMI

EKB Manufacturing Intelligence is de applicatie waarin de software van deze afstudeeropdracht geïmplementeerd zal worden. Het bevat de modules Overall Equipment Effectiveness (OEE), Tool Management, Rapportage, Planning, Logboek, Label Printing en Configuratie zoals te zien in onderstaande foto (Rang, 2018).



Rang, S. (2018, 3 april). EMI modules van de rollenfabriek van Tsubaki Nakashima [Foto], Geraadpleegd van http://emi-demo.ekb.nl/EMI_NNN/Default.aspx

EMI communiceert met een SQL Server database waarin voor elke klant van EKB data voor deze zeven modules gestandaardiseerd worden bijgehouden. Deze data zijn dus goed te gebruiken om een generiek algoritme te schrijven, omdat er niet per klant dataconfiguratie plaats hoeft te vinden.

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

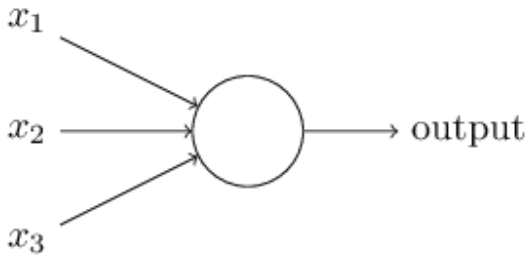
In bovenstaande foto (Rang, 2018) is een voorbeeld van de OEE analyse in EMI te zien van een persmachine genaamd 101RH1. De rode staaf van de KPI's is de beschikbaarheid van de machine, oftewel hoe lang de machine heeft geproduceerd, ongeacht de snelheid of kwaliteit. De blauwe staaf van de KPI's is de prestatie van de machine. Dit gaat over de snelheid ten opzichte van de normsnelheid. Een prestatie van 50% betekent dat de machine gedurende de geselecteerde periode op de helft van de normsnelheid heeft geproduceerd. De gele staaf van de KPI's is de kwaliteit, oftewel hoeveel van de productie geaccepteerd is na een bepaalde check. Een kwaliteit van 75% betekent dat 25% van de productie niet goed genoeg is en wordt gezien als uitval of afkeur. Deze drie percentages van de KPI's worden vermenigvuldigd om te komen tot de groene OEE staaf. Dit is de Overall Equipment Effectiveness en geeft een samenvattend inzicht op de prestatie van de machine. In het geval van de 101RH1 is de OEE van

januari 49,75%. Dit komt voornamelijk doordat de machine niet bediend is. Deze reden is terug te vinden rechtsonderin het scherm waar de status redenen gesorteerd staan gesorteerd op tijdsduur.

4.2.3 Machine Learning

4.2.3.1 Perceptron

Aan de basis van Machine Learning staan zogeheten Neural Networks. Dit is een manier om op basis van bepaalde inputs, outputs te berekenen. De meest simpele vorm hiervan is een *perceptron*. Volgens Nielsen (2017) kan een perceptron op basis van een som van meerdere inputs een output berekenen van 0 (uit) of 1 (aan). In Figuur 3 wordt een perceptron weergegeven. x_1, x_2 en x_3 zijn de drie inputs van deze perceptron. De output is te berekenen met de formule van Figuur 6. Elke lijn van de inputs in Figuur 3 is een *weight* (w in Figuur 4). Dit is in essentie een getal, meestal tussen de -5 en 5 waarmee de input vermenigvuldigd wordt.



Figuur 3: Perceptron
Noot. Herdrukt van “Using neural nets to recognize handwritten digits”, door Nielsen, M., (2017, 1 december). Geraadpleegd van http://neuralnetworksanddeeplearning.com/chap1.html

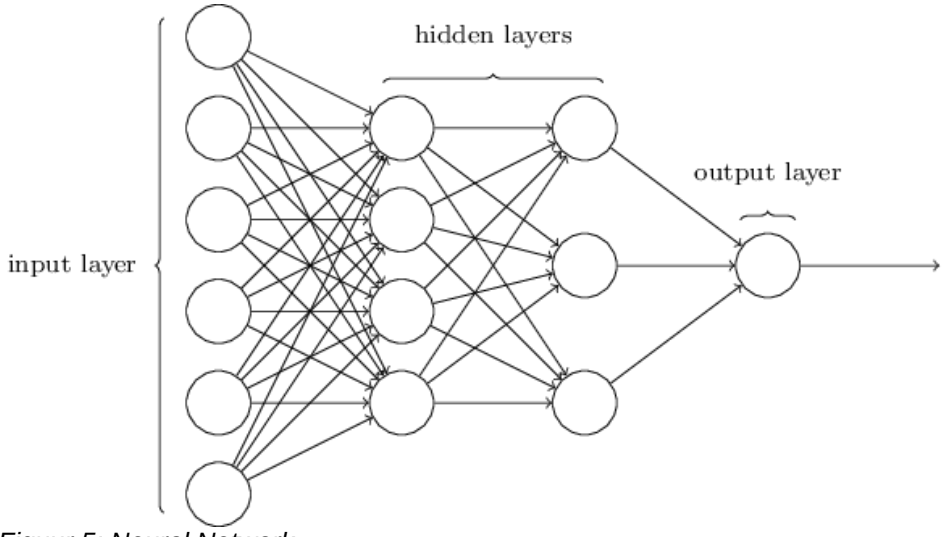
$$\text{output} = \begin{cases} 0 & if \ \sum_j w_j x_j \leq threshold \\ 1 & if \ \sum_j w_j x_j > threshold \end{cases}$$

Figuur 4: Perceptron output formule
Noot. Herdrukt van “Using neural nets to recognize handwritten digits”, door Nielsen, M., (2017, 1 december). Geraadpleegd van http://neuralnetworksanddeeplearning.com/chap1.html

Alle drie inputs worden vermenigvuldigd met hun 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 j in de formule van Figuur 4 is het nummer van de input x.

4.2.3.2 Neural Networks

Een neuraal netwerk is niet meer dan een netwerk van meerdere lagen met perceptrons. Zoals te zien is in Figuur 5 bestaat een neural network uit een *input layer*, een *hidden layer* en een *output layer*. De input layer krijgt de waardes van de buitenwereld en berekend met de perceptrons en weights de waardes van elke perceptron in de opvolgende layers totdat de output perceptrons in de output layer berekend zijn.



Figuur 5: Neural Network
Noot. Herdrukt van “Using neural nets to recognize handwritten digits”, door Nielsen, M., (2017, 1 december). Geraadpleegd van http://neuralnetworksanddeeplearning.com/chap1.html

In het geval van neural networks spreken we echter over *neurons* in plaats van perceptrons. Het verschil met perceptrons is dat neurons een extra berekening hebben om tot de output te komen en deze output is niet digitaal maar analoog. Naast de waarde x en de bijbehorende weights w_n heeft een neuron ook een *bias* b. Dit is een extra variabele per neuron die opgeteld wordt bij de som van de weights keer de waardes van de neurons in de vorige layer. Daarna wordt extra berekening gedaan genaamd de *activation function*.

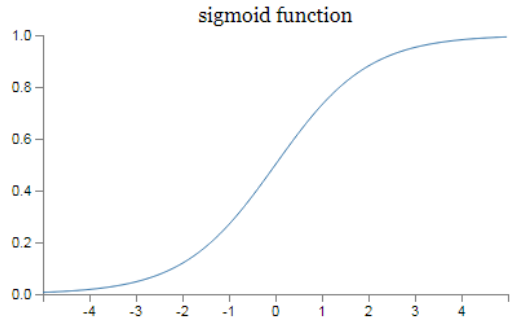
4.2.3.3 Activation Functions

Een van de meest gebruikte activation functions is de *sigmoid function*. Dit is een functie om elke willekeurige waarde om te zetten naar een getal tussen de 0 en de 1. Er zijn vele variaties op deze functie, maar de meest gebruikte versie is te zien in Figuur 6.

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

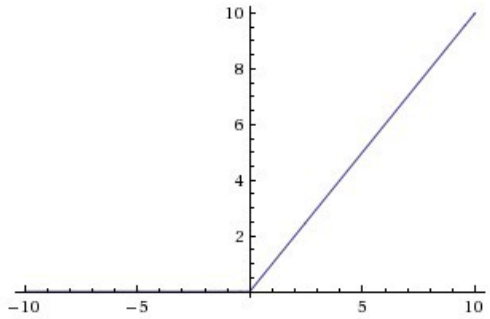
Figuur 6: Sigmoid function

Zoals te zien is in Figuur 7 worden de waardes gelijkmatig verdeeld tussen de 0 en de 1. Volgens Rojas (1996) is het noodzakelijk een functie te gebruiken met een gelijkmatige helling om te zorgen dat het algoritme de weights van de neurons ook gelijkmatig kan veranderen om zo langzaam bij het gewenste resultaat te komen.



Figuur 7: Plot van de sigmoid function
Noot. Herdrukt van "Using neural nets to recognize handwritten digits", door Nielsen, M., (2017, 1 december). Geraadpleegd van http://neuralnetworksanddeeplearning.com/chap1.html

Een andere activation function die eigenlijk steeds vaker wordt verkozen boven de sigmoid function is de *ReLU (Rectified Linear Unit)*. Dit is een hele simpele maar effectieve activation function die alle negatieve waardes omzet naar 0 en niets doet met de positieve waardes. De functie luidt: $A(x) = max(0, x)$.In Figuur 8 staat de plot van de ReLU functie.



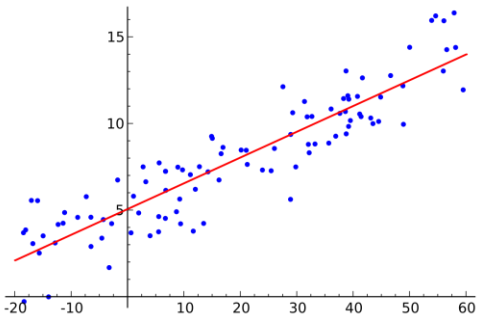
Figuur 8: Rectified Linear Unit (ReLU) activation function
Noot. Herdrukt van "Rectified-Linear unit Layer", door Santos, L., (2018, 1 januari). Geraadpleegd van https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/relu_layer.html

In 2017 onderzocht Sharma de voor- en nadelen van de ReLU function. Een belangrijk voordeel is dat de negatieve waardes van neurons niet meer doorberekend hoeven te worden omdat ze door de ReLU function nul worden. Dit scheelt dus in de snelheid waarmee het neural network berekend kan worden. Een nadeel van ReLU is dat de functie geen helling heeft voor negatieve waardes. Dit betekent dat de neuron niet meer beïnvloed kan worden door kleine aanpassingen in de weights, omdat dit gebeurt op basis van de helling. Volgens Sharma (2017) heet dit de 'dying ReLU problem' waardoor deze neurons geen nut meer hebben voor het netwerk. Dit kan eventueel opgelost worden met een zogeheten *leaky ReLU*. Deze functie is: $A(x) = 0.01x$voor negatieve waardes van x en $A(x) = x$ voor positieve waardes. Hierdoor is de helling voor negatieve waardes 0.01 waardoor de 'dying ReLU problem' wordt verholpen.

Tot slot is er ook de simpele hyperbolische tangens $f(x) = tanh(x)$ die eigenlijk een variatie is op de sigmoid function: $tanh(x) = 2\sigma(2x) - 1$. Het verschil is dat deze functie alle waardes van x omzet naar waardes tussen de -1 en 1 en dat de helling twee keer steiler is.

4.2.3.4 Supervised learning

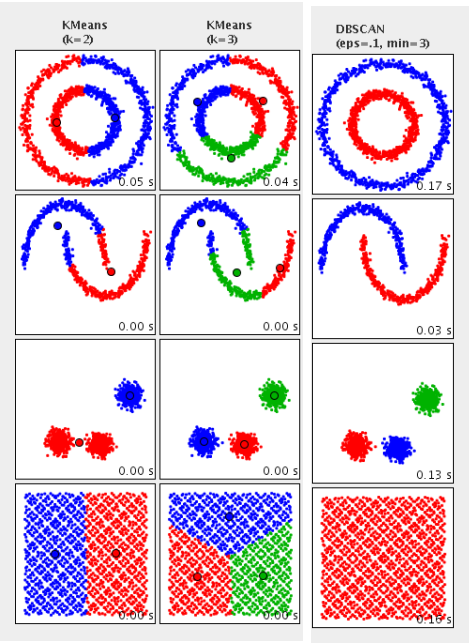
Supervised learning is een manier om een neural network of ander algoritme bepaalde patronen aan te leren. De uitkomsten van het neural network of algoritme zijn bekend. Hierdoor is het trainingsproces vaak vrij kort.
Een vorm van supervised learning is *backpropagation*. Zoals al eerder vermeld worden bij deze training de weights en biases van neurons in een neural network op basis van de helling van de activation function beetje bij beetje aangepast om steeds dichter bij het gewenste resultaat te komen. Deze techniek heet *gradient descent* en is al sinds de jaren 80 gebruikt, maar wordt steeds minder gebruikt. De reden hiervoor is dat voor veel problemen de oplossing nog niet bekend is. Wel wordt backpropagation veel gebruikt om dingen te automatiseren die mensen simpel en saai werk vinden zoals het herkennen van objecten in een video of afbeelding. Hierbij is voor een mens al van tevoren duidelijk wat er staat maar is het doel om deze patroonherkenning te automatiseren zodat nieuwe afbeeldingen op dezelfde manier verwerkt kunnen worden. Een voorbeeld is Google die in de Google Image Search dit algoritme gebruiken om relevantere resultaten te vinden.
Een andere vorm van supervised learning is *classification*. Bij deze techniek wordt niet gebruik gemaakt van een neural network, maar een algoritme om de training data te verwerken. Bijvoorbeeld een hele hoop afbeeldingen van stukken fruit die moeten worden verdeeld in de groepen appels, peren en overigen. Deze groepen worden van tevoren vastgesteld.
Ook *regression* is een supervised learning methode. Ook hierbij wordt niet gebruik gemaakt van een neural network maar van een algoritme. Dit algoritme wordt gebruikt om een schatting te maken voor een functie die het dichtst in de buurt van de training data komt. Het maakt niet uit hoeveel parameters er gebruikt worden, maar wat wel van invloed is op de uitkomst is het aantal termen dat het algoritme gebruikt om de functie te formuleren.



Figuur 9: Regression
Noot. Herdrukt van “Regression analysis”, door Wikipedia, (2018, 1 april). Geraadpleegd van https://en.wikipedia.org/wiki/Regression_analysis

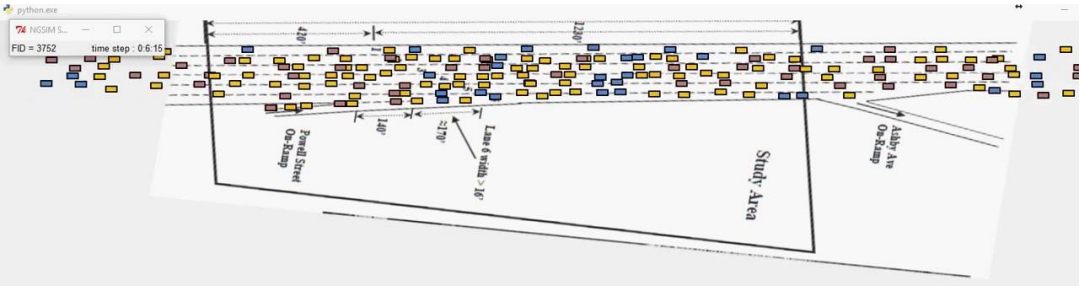
4.2.3.5 Unsupervised learning

Unsupervised learning is een manier om onbekende patronen te herkennen in data. Een voorbeeld hiervan is *clustering* van data zonder dat het aantal clusters of de namen hiervan bekend zijn. Een bekend algoritme hiervoor is *K-means* of *DBSCAN* (Figuur 10).

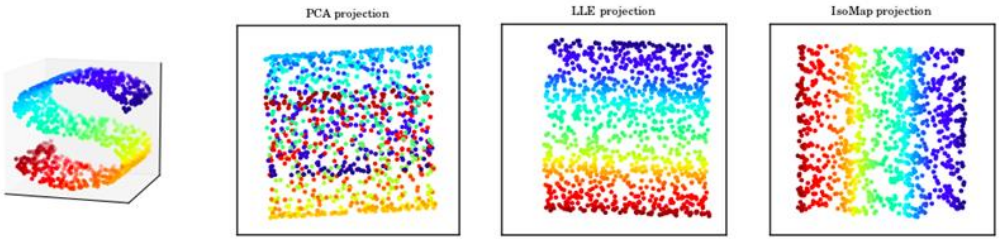


Figuur 10: K-means (2 en 3) en DBSCAN clustering algoritmes
Noot. Aangepast van “Clustering algorithms and distance measures”, door Apache, (2016, 28 augustus). Geraadpleegd van http://commons.apache.org/proper/commons-math/userguide/ml.html

Naast clustering vallen onder unsupervised learning ook *density estimation* en *dimensionality reduction*. Density estimation wordt gebruikt om van een aantal datapunten een overzichtelijk beeld van de verdeling te krijgen. Dit kan bijvoorbeeld worden gebruikt om verkeersdrukte over een groot gebied te analyseren (Figuur 11). Dimensionality reduction is een manier om redundante of overbodige parameters uit een dataset te filteren of om data overzichtelijker weer te geven met minder parameters (Figuur 12).



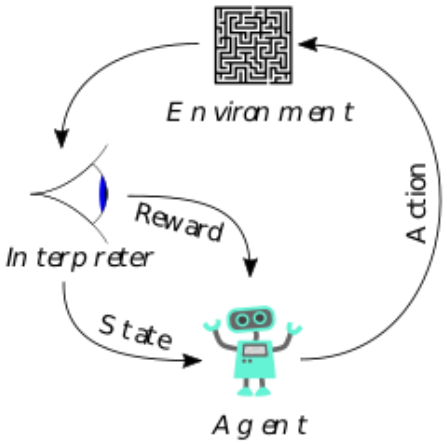
Figuur 11: Density estimation algoritme voor verkeersdrukte
Noot. Aangepast van “NGSIM Simulator to evaluate a traffic density estimation algorithm by using sensor equipped vehicles”, door Nam, D., (2017, 23 oktober). Geraadpleegd van https://www.youtube.com/watch?v=kpgR_DxhzQE



Figuur 12: Diverse dimensionality reduction algoritmes
Noot. Aangepast van “Comparison of PCA and Manifold Learning”, door Vanderplas, J., (2012). Geraadpleegd van http://www.astroml.org/book_figures/chapter7/fig_S_manifold_PCA.html

4.2.3.6 Reinforcement learning

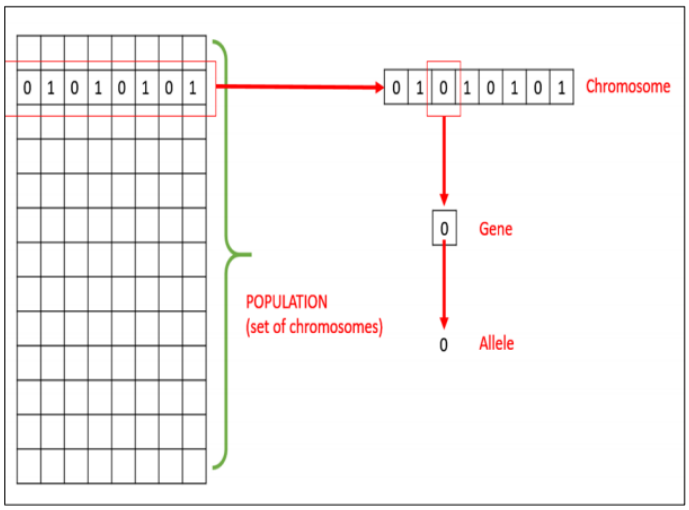
Volgens Van Otterloo en Wiering (2009) is reinforcement learning een combinatie van supervised- en unsupervised learning waarbij een *agent* in een omgeving streeft naar het maken van goede beslissingen op basis van gelimiteerde feedback. De agent voert acties uit op basis van de huidige omgeving, de *state*. Een *interpreter* analyseert vervolgens het resultaat van deze actie en geeft feedback op de agent dmv *rewards* en geeft de agent vervolgens de nieuwe state om de agent een nieuwe actie uit te laten voeren. Een voorbeeld is een spel zoals super mario waarin de state het huidige scherm is en de reward hoe dichtbij de agent is van de finish en de score in het spel. Wanneer de agent een hoge reward krijgt leert de agent dat de actie die hierbij hoorde goed was en wordt hierdoor beter in het uitvoeren van de juiste acties in vergelijkbare states.



Figuur 13: Reinforcement learning algoritme
Noot. Herdrukt van “Reinforcement learning”, door Wikipedia, (2018, 2 april). Geraadpleegd van https://en.wikipedia.org/wiki/Reinforcement_learning

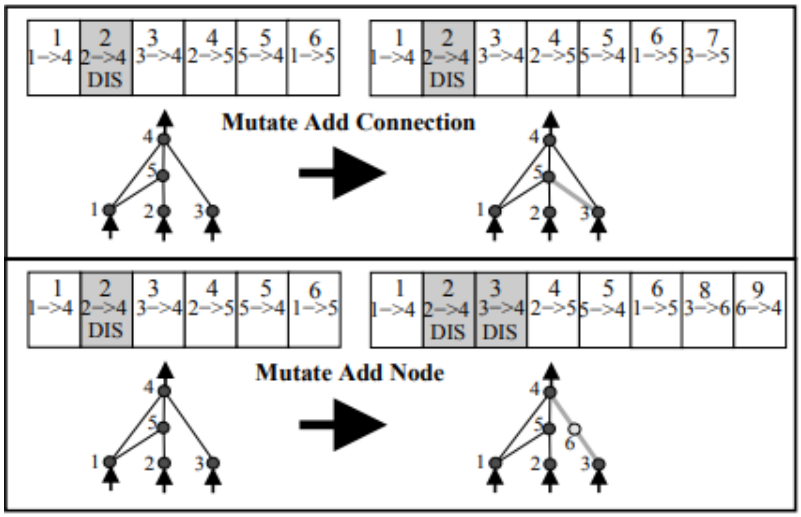
4.2.3.7 Genetic algorithms

Genetic algorithms kunnen ook weer worden onderverdeeld in vele verschillende algoritmes, maar in het kort is een genetic algorithm vergelijkbaar met reinforcement learning. Echter gebruikt een genetic algorithm een hele hoop agents tegelijkertijd zonder een interpreter. Deze agents worden *genomes* of soms *chromosomes* genoemd. Een genome is simpelweg een lijst van bits of analoge waardes die *genes* heten (Figuur 14). Deze waardes worden in de omgeving gescoord. Dit is de *fitness* waarmee via natuurlijke selectie een nieuwe *generation* (generatie of populatie) van genomes wordt gegenereerd waarvan opnieuw de fitness berekend wordt.



Figuur 14: De basis structuur en terminologie van genetic algorithms
Noot. Herdrukt van “Genetic Algorithms”, door Tutorials Point (I) Pvt. Ltd.., (2016). Geraadpleegd van https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_tutorial.pdf

Volgens Tutorials Point (I) Pvt. Ltd.. (2016) is er geen helling nodig om de genomes te trainen, kun je meerdere genomes parallel laten trainen en train je niet een goed algoritme, maar een hele lijst van genomes die goed presteren. Echter is een genetic algorithm niet geschikt voor simpele problemen waar wel helling informatie over beschikbaar is of waarvan de oplossing al bekend is.



Figuur 15: De twee soorten mutatie van een neural network dmv het NEAT algoritme
Noot. Herdrukt van “Evolving Neural Networks through Augmenting Topologies”, door Stanley, K., (2004). Geraadpleegd van http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf

Bij genetic algorithms staat het aantal genes vast en wordt er niet gebruik gemaakt van neural networks. Echter in het NEAT (NeuroEvolution of Augmenting Topologies) algoritme wordt er wel gebruik gemaakt van neural networks en wordt de architectuur van deze netwerken ook dmv natuurlijke selectie en evolutie verder ontwikkeld. Er kan zowel een weight-lijn als een een compleet nieuwe perceptron worden toegevoegd (Figuur 15). Daarnaast wordt de zoektocht naar het beste netwerk ook verbreed door het gebruik van zogeheten *species*. Volgens Stanley (2004) wordt in het NEAT algoritme innovatie van nieuwe architecturen beschermd dmv species en het verdelen van de fitness onder genomes van dezelfde species. Daarnaast begint het NEAT algoritme ook met een zo klein mogelijke architectuur van het netwerk om een zo simpel en efficiënt

mogelijke oplossing te vinden. Dit blijkt ook uit de experimenten die met NEAT zijn uitgevoerd door K. O. Stanley.

4.2.4 Te onderzoeken literatuur

Om de afstudeerstage te kunnen starten moet er eerst kennis worden opgedaan over een aantal belangrijke onderwerpen en termen. Deze zijn terug te vinden in Tabel 5.

Tabel 5: Voorlopige literatuurlijst

ONDERWERP/TERM	LITERATUUR
TOC	Goldratt, E. M. (1986). Het Doel. Houten, Nederland: Spectrum. Goldratt, E. M., & Cox, J. (2007, 2 april). The Goal, A Process of Ongoing Improvement. Geraadpleegd van http://www.2ndbn5thmar.com/lean/Notes%20on%20The%20Goal.pdf Gattiker, T. (2015, 21 oktober). Virtual Lecture: Theory of Constraints / The Goal (Part 1) [Video]. Geraadpleegd van https://www.youtube.com/watch?v=riqvCu5FBiw Gattiker, T. (2015, 21 oktober). Virtual Lecture: Theory of Constraints / The Goal (Part 2) [Video]. Geraadpleegd van https://www.youtube.com/watch?v=rzuDLFTkoIg Ede, J. van. (2017, 6 september). TOC definitie Procesverbeteren.nl. Geraadpleegd van https://www.procesverbeteren.nl/TOC/ToC.php
EMI	Volg en vind met EMI - Traceability in de procesindustrie. (2014, 1 januari). Geraadpleegd van http://www.ekb.nl/nl/nieuws/volg_en_vind_met_emi
Machine Learning	Stanley, K. O. (2004). Efficient Evolution of Neural Networks through Complexification (Report AI-TR-04-314). Geraadpleegd van http://nn.cs.utexas.edu/downloads/papers/stanley.phd04.pdf Olah, C. (2015, 27 augustus). Understanding LSTM Networks [Blogpost]. Geraadpleegd van https://colah.github.io/posts/2015-08-Understanding-LSTMs/ Heaton, J. (2014, 1 oktober). Encog 3.3: Development Guide. Geraadpleegd van https://s3.amazonaws.com/heatonresearch-books/free/encog-3_3-devguide.pdf Nielsen, M. A. (2017). *Neural Networks and Deep Learning*. Geraadpleegd van http://neuralnetworksanddeeplearning.com/index.html Rojas, R. (1996). The Backpropagation Algorithm. Geraadpleegd van https://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf Tutorials Point (I) Pvt. Ltd.. (2016). Genetic Algorithms. Geraadpleegd van https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_tutorial.pdf Otterlo, M. van, & Wiering, M. (2009). Reinforcement Learning and Markov Decision Processes. Geraadpleegd van http://www.ai.rug.nl/~mwiering/Intro_RLBOOK.pdf

5 PLAN VAN AANPAK

5.1 Globale aanpak

Nadat het Plan van Aanpak goedgekeurd is zal er gestart worden met extra kennis vergaren over machine learning, TOC en EMI om een plan te maken voor de verschillende experimenten. Vervolgens zal er een experimentele omgeving bij TN worden opgezet om met verschillende machine learning algoritmes te experimenteren op testdata van een of meerdere productielijnen van TN. Uit deze experimenten volgen een of meerdere geschikte algoritmes die ingezet kunnen worden voor het eindproduct. Daarna zullen er meerdere interviews met de opdrachtgever plaatsvinden over de uitstraling en werking van het eindproduct waarna via SCRUM het eindproduct met het machine learning algoritme wordt gerealiseerd.

5.2 Programmeermethoden

Bij het eventueel realiseren van een of meerdere machine learning algoritmes, maar ook voor het realiseren van het eindproduct zal er gewerkt worden volgens de SCRUM-project techniek waarbij M. de Lange als product owner zal fungeren en de student zelf als SCRUM-master het project zal doorlopen. Voor de backlog van SCRUM wordt Redmine gebruikt. EKB verwacht dat de software modulair opgebouwd zal worden en herbruikbaar en uitbreidbaar is. Ten slotte wordt het eindproduct van een bijgehouden versienummer voorzien en wordt de nieuwste versie en oudere versies opgeslagen in Git.

De front-end wordt geprogrammeerd in HTML5, CSS, en Javascript. De back-end zal worden gerealiseerd in het ASP .NET framework in de C# taal en er zal gebruik worden gemaakt van SQL voor communicatie met de database.

5.3 Tijdsplanning en mijlpalen

In Tabel 6 wordt de tijdsindeling en de verschillende mijlpalen van deze afstudeerstage weergegeven.

Tabel 6: Tijdsplanning en mijlpalen

ACTIVITEIT	START- EN EINDDATUM	UREN
Plan van Aanpak	Maandag 5 maart 2018 - Dinsdag 10 april 2018	216
Deelvraag 1	Woensdag 11 april 2018 - Donderdag 12 april 2018	16
Deelvraag 2	Vrijdag 13 april 2018 - Donderdag 19 april 2018	40
Deelvraag 3	Vrijdag 20 april 2018 - Maandag 23 april 2018	16
Deelvraag 4	Dinsdag 24 april 2018	8
Deelvraag 5	Woensdag 25 april 2018 - Donderdag 26 april 2018	16
Back-end programmeren	Vrijdag 27 april 2018 - Donderdag 3 mei 2018	40
Front-end programmeren	Vrijdag 4 mei 2018 - Maandag 7 mei 2018	16
Scriptie schrijven	Dinsdag 8 mei 2018 - Donderdag 24 mei 2018	104
Eerste conceptversie scriptie inleveren	Donderdag 24 mei 2018	n.v.t.
Scriptie herschrijven	Woensdag 30 mei 2018 - Dinsdag 12 juni 2018	80
Tweede conceptversie scriptie inleveren	Dinsdag 12 juni 2018	n.v.t.
Definitieve versie scriptie schrijven	Woensdag 20 juni 2018 - Dinsdag 3 juli 2018	80
Definitieve versie scriptie inleveren	Dinsdag 3 juli 2018	n.v.t.
Totaal	**Maandag 5 maart 2018 - Dinsdag 3 juli 2018**	**632**

6 RISICO'S

6.1 Risico analyse

In Tabel 7 worden de verwachte risico`s van deze afstudeeropdracht weergegeven. De impact van de risico`s is eigenlijk altijd redelijk hoog aangezien het anders geen risico is. De kans dat het risico zich tijdens het afstudeertraject voordoet samen met de impact geeft een beeld van de prioriteit van het risico.

Tabel 7: Risico analyse

OMSCHRIJVING	IMPACT	KANS	RISICO	MAATREGELEN
Onvoldoende tijd voor documentatie/scriptie	Hoog	Gemiddeld	Hoog	Aanpassen van de tijdsplanning zodat er meer tijd gereserveerd wordt voor het documenteren
Onvoldoende of niet representatieve data voor de machine learning algoritmes	Hoog	Laag	Gemiddeld	Overwegen of er meer externe data gebruikt zou kunnen worden om de data uit te breiden en representatiever te maken, of aannames doen over bepaalde variabelen die aanpasbaar gemaakt kunnen worden in de implementatie
Onvoldoende tijd voor Must Have functionaliteiten	Hoog	Laag	Gemiddeld	Deze functionaliteiten kunnen worden opgenomen in het Proof of Concept en/of worden uitbesteed aan de developers van EKB
Onvoldoende uitbreidbare en generieke implementatie	Gemiddeld	Gemiddeld	Gemiddeld	Goede documentatie, code commentaar en het Proof of Concept zorgen ervoor dat EKB over de kennis beschikt dit later alsnog te realiseren
Te lage prestaties van de machine learning algoritmes	Gemiddeld	Laag	Laag	Meer tijd reserveren voor aanvullende experimenten met andere parameters of eventueel andere algoritmes. Eventueel de implementatie van het algoritme opnemen in het Proof of Concept

De kans dat er onvoldoende of niet representatieve data beschikbaar zijn, is vrij laag, omdat er volgens M. Kok in EMI al sinds begin 2017 elke minuut van bijna alle productielijnen van TN metingen worden gedaan van de productie. Ook wordt er per productielijn bijgehouden aan welke orders er op dit moment wordt gewerkt en worden machine snelheden en storingen opgeslagen in de database van EMI. Voor de kostenafweging is er volgens G. Bargeman voldoende financiële data beschikbaar bij TN om dit te implementeren.

6.2 Persoonlijke uitdagingen

Een van de grootste uitdagingen aan de afstudeeropdracht is het schrijven van de scriptie zelf. Ik vind het lastig om beschrijvingen van experimenten of begrippen kort te houden en ga vaak snel de details in. Om de scriptie op een goede manier te schrijven zal ik mijn experimenten en begrippen kort en krachtig moeten omschrijven. Daarnaast zal het inplannen van de verschillende fases van de afstudeerstage ook een uitdaging worden, omdat zowel de onderzoeks- als de implementatiefase veel tijd in beslag zal nemen. Daarentegen zullen de meeste experimenten gedeeltelijk zonder toezicht verlopen waardoor er tijd vrij is voor andere activiteiten.

LITERATUUR

Apache, (2016, 28 augustus). Clustering algorithms and distance measures. Geraadpleegd van http://commons.apache.org/proper/commons-math/userguide/ml.html

Ede, J. van. (2017, 6 september). TOC definitie Procesverbeteren.nl. Geraadpleegd op 23 november 2017, van https://www.procesverbeteren.nl/TOC/ToC.php

EMI OEE analyse van de rollenfabriek van Tsubaki Nakashima [Foto]. (2017). Geraadpleegd op 3 april 2018, van http://emi-demo.ekb.nl/EMI_NNN/OEE/OEEAnalyse.aspx

EMI modules overzicht [Foto]. (2017). Geraadpleegd op 3 april 2018, van http://emi-demo.ekb.nl/EMI_NNN/Default.aspx

Goldratt, E. M. (1986). Het Doel. Houten, Nederland: Spectrum.

Goldratt, E. M., & Cox, J. (2007, 2 april). The Goal, A Process of Ongoing Improvement. Geraadpleegd van http://www.2ndbn5thmar.com/lean/Notes%20on%20The%20Goal.pdf

Nam, D. (2017, 23 oktober). NGSIM Simulator to evaluate a traffic density estimation algorithm by using sensor equipped vehicles [Video]. Geraadpleegd van https://www.youtube.com/watch?v=kpgR_DxhzQE

Nielsen, M. A. (2017). *Neural Networks and Deep Learning*. Geraadpleegd van http://neuralnetworksanddeeplearning.com/index.html

Nielsen, M. (2017, 1 december). Using neural nets to recognize handwritten digits. Geraadpleegd van http://neuralnetworksanddeeplearning.com/chap1.html

Otterlo, M. van, & Wiering, M. (2009). Reinforcement Learning and Markov Decision Processes. Geraadpleegd van http://www.ai.rug.nl/~mwiering/Intro_RLBOOK.pdf

Rojas, R. (1996). The Backpropagation Algorithm. Geraadpleegd van https://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf

Santos, L. (2018, 1 januari). Rectified-Linear unit Layer. Geraadpleegd van https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/relu_layer.html

Sharma, A. (2017, 30 maart). Understanding Activation Functions in Neural Networks [Blogpost]. Geraadpleegd van https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

Stanley, K. O. (2004). Efficient Evolution of Neural Networks through Complexification (Report AI-TR-04-314). Geraadpleegd van http://nn.cs.utexas.edu/downloads/papers/stanley.phd04.pdf

Tutorials Point (I) Pvt. Ltd.. (2016). Genetic Algorithms. Geraadpleegd van https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_tutorial.pdf

Vanderplas, J. (2012). Comparison of PCA and Manifold Learning. Geraadpleegd van http://www.astroml.org/book_figures/chapter7/fig_S_manifold_PCA.html

Wikipedia, (2018, 1 april). Regression analysis. Geraadpleegd van https://en.wikipedia.org/wiki/Regression_analysis

Wikipedia, (2018, 2 april). Reinforcement learning. Geraadpleegd van https://en.wikipedia.org/wiki/Reinforcement_learning

Bijlage B: 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).

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). 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 ander 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.

Bijlage C: VisualFlow buffer-data

Perserij buffer-data

PersProductieID	valuedVolume	Gewicht	ProductID	Rollen	**Totaal**	**Totaal (x1000)**
28083	500	2,335421	13370	214094,161181217	449597,7385	449,597738480557
28521	550	2,335421	13370	235503,577299339		
28917	300	2,449882	13684	122454,877418586	122454,8774	122,454877418586
29099	500	2,827970	13354	176805,270211494	176805,2702	176,805270211494
29111	500	2,743780	13372	182230,353745563	182230,3537	182,230353745563

Trommels buffer-data

TrommelProductieID	Gewicht	ProductID	Rollen	**Totaal**	**Totaal (x1000)**
290	2,827	133	176805,27021149	176805,2702114940	176,8052702114940
27867	2,335421	13370	214094,1611812170	3425506,5788994800	3425,5065788994800
28083	2,335421	13370	214094,1611812170		
28321	2,335421	13370	214094,1611812170		
28387	2,335421	13370	214094,1611812170		
28419	2,335421	13370	214094,1611812170		
28441	2,335421	13370	214094,1611812170		
28501	2,335421	13370	214094,1611812170		
28521	2,335421	13370	214094,1611812170		
28547	2,335421	13370	214094,1611812170		
28563	2,335421	13370	214094,1611812170		
28579	2,335421	13370	214094,1611812170		
28645	2,335421	13370	214094,1611812170		
28663	2,335421	13370	214094,1611812170		
28915	2,335421	13370	214094,1611812170		
28961	2,335421	13370	214094,1611812170		
28979	2,335421	13370	214094,1611812170		
29041	2,74378	13372	182230,3537455630	364460,7074911250	364,4607074911250
29073	2,74378	13372	182230,3537455630		
28657	8,085805	13373	61836,7620787293	680204,3828660230	680,2043828660230
28658	8,085805	13373	61836,7620787293		
28823	8,085805	13373	61836,7620787293		
28824	8,085805	13373	61836,7620787293		
28883	8,085805	13373	61836,7620787293		
28884	8,085805	13373	61836,7620787293		
28937	8,085805	13373	61836,7620787293		
28949	8,085805	13373	61836,7620787293		
28950	8,085805	13373	61836,7620787293		
28969	8,085805	13373	61836,7620787293		
28970	8,085805	13373	61836,7620787293		
28863	4,38	13388	114155,2511415530	684931,5068493150	684,9315068493150
28864	4,38	13388	114155,2511415530		
28907	4,38	13388	114155,2511415530		
28908	4,38	13388	114155,2511415530		
28995	4,38	13388	114155,2511415530		
28996	4,38	13388	114155,2511415530		

28927	3,93	13392	127226,4631043260	1017811,7048346100	1017,8117048346100
28941	3,93	13392	127226,4631043260		
28942	3,93	13392	127226,4631043260		
28999	3,93	13392	127226,4631043260		
29000	3,93	13392	127226,4631043260		
29039	3,93	13392	127226,4631043260		
29040	3,93	13392	127226,4631043260		
29067	3,93	13392	127226,4631043260		
28725	4,444089	13457	112508,9979071080	225017,9958142150	225,0179958142150
28873	4,444089	13457	112508,9979071080		
28471	6,896086	13459	72504,8962556441	507534,2737895090	507,5342737895090
28472	6,896086	13459	72504,8962556441		
28581	6,896086	13459	72504,8962556441		
28582	6,896086	13459	72504,8962556441		
28611	6,896086	13459	72504,8962556441		
28612	6,896086	13459	72504,8962556441		
28659	6,896086	13459	72504,8962556441		
28105	20,12329	13464	24846,8317059487	49693,6634118974	49,6936634118974
28106	20,12329	13464	24846,8317059487		
20223	24,84572	13541	20124,1904038201	321987,0464611210	321,9870464611210
20224	24,84572	13541	20124,1904038201		
20307	24,84572	13541	20124,1904038201		
20308	24,84572	13541	20124,1904038201		
22069	24,84572	13541	20124,1904038201		
22070	24,84572	13541	20124,1904038201		
28763	24,84572	13541	20124,1904038201		
28764	24,84572	13541	20124,1904038201		
28773	24,84572	13541	20124,1904038201		
28774	24,84572	13541	20124,1904038201		
28775	24,84572	13541	20124,1904038201		
28776	24,84572	13541	20124,1904038201		
28787	24,84572	13541	20124,1904038201		
28788	24,84572	13541	20124,1904038201		
28875	24,84572	13541	20124,1904038201		
28876	24,84572	13541	20124,1904038201		
28791	4,689985	13564	106610,1490729710	639660,8944378290	639,6608944378290
28792	4,689985	13564	106610,1490729710		
29027	4,689985	13564	106610,1490729710		
29028	4,689985	13564	106610,1490729710		
29069	4,689985	13564	106610,1490729710		
29070	4,689985	13564	106610,1490729710		

28265	23,94994	13600	20876,8790234965	459291,3385169230	459,2913385169230
28266	23,94994	13600	20876,8790234965		
28925	23,94994	13600	20876,8790234965		
28926	23,94994	13600	20876,8790234965		
28957	23,94994	13600	20876,8790234965		
28958	23,94994	13600	20876,8790234965		
28965	23,94994	13600	20876,8790234965		
28966	23,94994	13600	20876,8790234965		
28981	23,94994	13600	20876,8790234965		
28991	23,94994	13600	20876,8790234965		
28993	23,94994	13600	20876,8790234965		
28994	23,94994	13600	20876,8790234965		
29001	23,94994	13600	20876,8790234965		
29021	23,94994	13600	20876,8790234965		
29023	23,94994	13600	20876,8790234965		
29024	23,94994	13600	20876,8790234965		
29029	23,94994	13600	20876,8790234965		
29030	23,94994	13600	20876,8790234965		
29047	23,94994	13600	20876,8790234965		
29048	23,94994	13600	20876,8790234965		
29053	23,94994	13600	20876,8790234965		
29054	23,94994	13600	20876,8790234965		
27979	3,246635	13639	154005,6088842760	308011,2177685510	308,0112177685510
27980	3,246635	13639	154005,6088842760		
29063	2,986364	13678	167427,6812873450	334855,3625746890	334,8553625746890
29089	2,986364	13678	167427,6812873450		
28607	2,832745	13679	176507,2394444260	176507,2394444260	176,5072394444260
28987	4,997692	13682	100046,1813172960	100046,1813172960	100,0461813172960
28783	2,449882	13684	204091,4623643100	612274,3870929290	612,2743870929290
28853	2,449882	13684	204091,4623643100		
28854	2,449882	13684	204091,4623643100		
28609	13,7578	13704	36343,0199595866	72686,0399191731	72,6860399191731
28610	13,7578	13704	36343,0199595866		
29081	19,6895	13711	25394,2456639326	50788,4913278651	50,7884913278651
29082	19,6895	13711	25394,2456639326		
27259	3,388	20346	147579,6930342390	1033057,8512396700	1033,0578512396700
27825	3,388	20346	147579,6930342390		
27861	3,388	20346	147579,6930342390		
27917	3,388	20346	147579,6930342390		
27945	3,388	20346	147579,6930342390		
27973	3,388	20346	147579,6930342390		
28007	3,388	20346	147579,6930342390		
26673	0,86	20404	581395,3488372090	3488372,0930232600	3488,3720930232600
28447	0,86	20404	581395,3488372090		
28797	0,86	20404	581395,3488372090		
28877	0,86	20404	581395,3488372090		
28967	0,86	20404	581395,3488372090		
29043	0,86	20404	581395,3488372090		

27381	52,96692	25774	9439,8541580292	113278,2498963500	113,2782498963500
27382	52,96692	25774	9439,8541580292		
27385	52,96692	25774	9439,8541580292		
27386	52,96692	25774	9439,8541580292		
27393	52,96692	25774	9439,8541580292		
27394	52,96692	25774	9439,8541580292		
27399	52,96692	25774	9439,8541580292		
27400	52,96692	25774	9439,8541580292		
27401	52,96692	25774	9439,8541580292		
27402	52,96692	25774	9439,8541580292		
27407	52,96692	25774	9439,8541580292		
27408	52,96692	25774	9439,8541580292		

Harderij buffer-data

OvenProductieID	Gewicht	ProductID	Rollen	**Totaal**	**Totaal (x1000)**
29033	2,827970	13354	176805,2702114940	353610,5404229890	353,61054042
29055	2,827970	13354	176805,2702114940		
27953	2,335421	13370	214094,1611812170	642282,4835436520	642,28248354
28083	2,335421	13370	214094,1611812170		
28167	2,335421	13370	214094,1611812170		
28658	8,085805	13373	61836,7620787293	556530,8587085640	556,53085871
28699	8,085805	13373	61836,7620787293		
28700	8,085805	13373	61836,7620787293		
28801	8,085805	13373	61836,7620787293		
28802	8,085805	13373	61836,7620787293		
28845	8,085805	13373	61836,7620787293		
28846	8,085805	13373	61836,7620787293		
28905	8,085805	13373	61836,7620787293		
28906	8,085805	13373	61836,7620787293		
28863	4,380000	13388	114155,2511415530	342465,7534246580	342,46575342
28963	4,380000	13388	114155,2511415530		
28964	4,380000	13388	114155,2511415530		
28941	3,930000	13392	127226,4631043260	381679,3893129770	381,67938931
28971	3,930000	13392	127226,4631043260		
28972	3,930000	13392	127226,4631043260		
28587	2,420868	13426	206537,4898590090	413074,9797180180	413,07497972
28588	2,420868	13426	206537,4898590090		
28323	4,589249	13438	108950,2879447160	217900,5758894320	217,90057589
28379	4,589249	13438	108950,2879447160		
28769	4,444089	13457	112508,9979071080	225017,9958142150	225,01799581
28803	4,444089	13457	112508,9979071080		
27757	6,896086	13459	72504,8962556441	652544,0663007970	652,54406630
27758	6,896086	13459	72504,8962556441		
27937	6,896086	13459	72504,8962556441		
27938	6,896086	13459	72504,8962556441		
28471	6,896086	13459	72504,8962556441		
28559	6,896086	13459	72504,8962556441		
28560	6,896086	13459	72504,8962556441		
28581	6,896086	13459	72504,8962556441		
28612	6,896086	13459	72504,8962556441		
28829	4,786395	13471	104462,7532830030	104462,7532830030	104,46275328
22069	24,845720	13541	20124,1904038201	181117,7136343800	181,11771363
28821	24,845720	13541	20124,1904038201		
28822	24,845720	13541	20124,1904038201		
28835	24,845720	13541	20124,1904038201		
28836	24,845720	13541	20124,1904038201		
28857	24,845720	13541	20124,1904038201		
28858	24,845720	13541	20124,1904038201		
28867	24,845720	13541	20124,1904038201		
28868	24,845720	13541	20124,1904038201		

28813	4,689985	13564	106610,1490729710	746271,0435108000	746,27104351
28814	4,689985	13564	106610,1490729710		
28837	4,689985	13564	106610,1490729710		
28838	4,689985	13564	106610,1490729710		
28871	4,689985	13564	106610,1490729710		
28872	4,689985	13564	106610,1490729710		
28920	4,689985	13564	106610,1490729710		
28266	23,949940	13600	20876,8790234965	313153,1853524480	313,15318535
28293	23,949940	13600	20876,8790234965		
28294	23,949940	13600	20876,8790234965		
28303	23,949940	13600	20876,8790234965		
28304	23,949940	13600	20876,8790234965		
28339	23,949940	13600	20876,8790234965		
28340	23,949940	13600	20876,8790234965		
28417	23,949940	13600	20876,8790234965		
28418	23,949940	13600	20876,8790234965		
28437	23,949940	13600	20876,8790234965		
28438	23,949940	13600	20876,8790234965		
28463	23,949940	13600	20876,8790234965		
28464	23,949940	13600	20876,8790234965		
28913	23,949940	13600	20876,8790234965		
28914	23,949940	13600	20876,8790234965		
27980	3,246635	13639	154005,6088842760	154005,6088842760	154,00560888
28929	2,986364	13678	167427,6812873450	1004566,0877240700	1004,56608772
28933	2,986364	13678	167427,6812873450		
28977	2,986364	13678	167427,6812873450		
28997	2,986364	13678	167427,6812873450		
29025	2,986364	13678	167427,6812873450		
29045	2,986364	13678	167427,6812873450		
29019	4,997692	13682	100046,1813172960	200092,3626345920	200,09236263
29020	4,997692	13682	100046,1813172960		
28851	2,449882	13684	204091,4623643100	816365,8494572390	816,36584946
28853	2,449882	13684	204091,4623643100		
28951	2,449882	13684	204091,4623643100		
28952	2,449882	13684	204091,4623643100		
28669	13,757800	13704	36343,0199595866	145372,0798383460	145,37207984
28670	13,757800	13704	36343,0199595866		
28741	13,757800	13704	36343,0199595866		
28742	13,757800	13704	36343,0199595866		
28743	0,860000	20404	581395,3488372090	581395,3488372090	581,39534884
28613	4,910000	20617	101832,9938900200	509164,9694501020	509,16496945
28614	4,910000	20617	101832,9938900200		
28649	4,910000	20617	101832,9938900200		
28650	4,910000	20617	101832,9938900200		
28757	4,910000	20617	101832,9938900200		
28727	13,076870	25810	38235,4493085884	76470,8986171767	76,47089862
28728	13,076870	25810	38235,4493085884		

Totale buffer-data

PersProductie Totaal	PersProductie Totaal (x1000)
931088,2398562	931,0882398562
TrommelProductie Totaal	**TrommelProductie Totaal (x1000)**
14842782,4771877	14842,7824771877
OvenProductie Totaal	**OvenProductie Totaal (x1000)**
8617544,54435894	8617,54454435894
Totaal Generaal	**Totaal Generaal (x1000)**
24391415,2614029	24391,4152614029