Student:

Teun Alma

keuzendeel Special io– GA (Generic Algoritm) chess puzzle mate in 3

248550

Inhoud

[Inleiding 2](#_Toc137565636)

[Versiebeheer 2](#_Toc137565637)

[Doel en doelgroepen 3](#_Toc137565638)

[Programma van Eisen 3](#_Toc137565639)

[1.0 Functioneel ontwerp 4](#_Toc137565640)

[1.1 Doel 4](#_Toc137565641)

[1.2 Fitness function 4](#_Toc137565642)

[1.21 GA Testing Params 4](#_Toc137565643)

[1.22 GA Program Params 5](#_Toc137565644)

[1.3 Ontwerpschets/wireframes 5](#_Toc137565645)

[2.0 Technisch ontwerp 5](#_Toc137565646)

[2.1 Doel 5](#_Toc137565647)

[2.2 Achtergrond 5](#_Toc137565648)

[2.5 Beveiligingsaspecten 6](#_Toc137565649)

[2.6 wettelijke aspecten 6](#_Toc137565650)

[2.7 beperkingen 6](#_Toc137565651)

[2.8 Hardware 6](#_Toc137565652)

[3.0 Interactie 6](#_Toc137565653)

[4.0 Iteratie 7](#_Toc137565654)

[4.1 Functie responsieve 7](#_Toc137565655)

[4.2 Testscenario 7](#_Toc137565656)

[4.3 Testinput 7](#_Toc137565657)

[4.4 Verwachte werking/output 7](#_Toc137565658)

[4.5 Werkelijke werking/output 8](#_Toc137565659)

[4.6 Overzicht testomgeving 8](#_Toc137565660)

[4.7 Planning testactiviteiten 8](#_Toc137565661)

[4.8 Conclusie testen 8](#_Toc137565662)

[Conclusie 8](#_Toc137565663)

[Bronvermelding 8](#_Toc137565664)

[Bijlage I - reflectie STARR 9](#_Toc137565665)

[Bijlage II – Planning 10](#_Toc137565666)

[Bijlage III - Logboek 17](#_Toc137565667)

# Inleiding

In dit document beschrijf ik in het kort het verloop van dit project en deel ik de problemen waar ik tegen aan liep en probeer ik mijn werkwijzen te laten zien.

Voor dit keuzendeel moest ik op een bestaand stuk softawre/hardware extra interface of functie toevoegen hier koos ik het spel schaken en probeerde hier puzzels voor te generen met behulp van python, ga-library en stockfish\_15.1.

* De beginnend beroepsbeoefenaaris actief gericht op nieuwe ontwikkelingen.
* De beginnend beroepsbeoefenaardeelt en bespreekt nieuwe ontwikkelingen met vakgenoten/collega’s etc.
* De beginnend beroepsbeoefenaarpakt werkzaamheden op een ordelijke en systematische manier aan.
* Er moet verzorgde documentatie beschikbaar zijn

Voor de rest beschrijft dit document het functioneel ontwerp van de aplicatie, Het document bevat ook andere referenties van het desbetreffende project. Naast dit zijn alle bronnen ook vermeld samen met de planning en het logboek. Tot slot is er ook een stukje met de STARR reflectie

# Versiebeheer

|  |  |  |  |
| --- | --- | --- | --- |
| Versie | Datum | Auteur | Wijzigingen |
| 1.0 | 31-5 | Teun Alma | Extra opdracht gekregen |
| 1.1 | 1-6 | Teun Alma | Begonnen de library addon |
| 1.2 | 5-6 | Teun Alma | Local stockfish insted of api |
| 1.3 | 5-6 | Teun Alma | Piece penalty |
| 2.0 | 6-6 | Teun Alma | Piece penalty |
| 2.1 | 7-6 | Teun Alma | New console logs for testing |
| 3.0 | 12-6 | Teun Alma | Puntjes op de i en begin documentatie |
| 3.1 | 13-6 | Teun Alma | Documentatie afegemaakt |
| 4.0 | 14-6 | Teun Alma | DONE |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Doel en doelgroepen

In dit scenario is de doelgroep mensen die beter willen worden is schaken en geen toegang hebben toe een apps of programma wat hun oneindig puzzels geeft om te oefenen in het herkennen van goede zetten. In dit geval zijn het zetten waar hun tegenstander achter staat maar je meerder stappen voor uit moet kijken om dit te kunnen herkennen. De bekende schaak app chess.com heeft deze feature maar het zit achter een pay-wall. Ik vind dat soort puzzels zelf het leukst m om telossen maar omdat op chess.com ik er maar 1 per dag mocht doen besloot ik dit zelf te gaan maken.

# Programma van Eisen

<onderdelen>

<software>

Python

programmertaal

Numpy

Python library

Chess

Python library

Genetic-algoritm

Python library

VS-code  
VS code is de code editor die ik heb gebruikt om een paar bestanden te editen. Vooral gebruikt gemaakt van processing 4, eigen text editor.

# 1.0 Functioneel ontwerp

## Doel

Om een extra/nieuwe vorm van interactie toe tevoegen aan een bestaande applicatie/game. In dit geval aan het bestaande spel schaken met behulp van genetic-algoritm een groep borden te generen. Deze worden vervolgens met een fitness functie geëvalueerd en de laagste de borden met de laagste fitness functie mogen vervolgens reproduceren door de helft van 1 en de andere helft van een ander laag bord te gebruiken en zo een nieuwe groep borden te genereren ook word het aller beste bord van de vorige groep behouden zo dat we nooit vooruitgang verliezen.

## Fitness function

In de fitness functie kijkt chess libray of de borden valid zijn. Dit houd in dat er b.v.b. geen pionnen of de backranks staan of meerder koningen in het spel zijn. Als dit het geval is word de penalty meteen op 10 gezet waardoor deze borden nooit voor new generation gekruikt worden. Daarna kijkt stockfish naar de huidige positie en evalueert hoeveel van de zetten schaak mat in 1 of 2 moves zijn. Er van uit gaande dat zwart altijd de beste zet in reactie speelt. Als er sprake is van mate in 1 krijgt dat bord een penalty van 9. Hetzelfde gebeurt voor mate in 2 alleen dan 8. Dit maakt dat ook deze borden niet snel reproduceren. Ook krijgt een bord penalty’s voor onmogelijke hoeveelheid pieces. Denk dan aan 3 of meer Queens voor 1 kleur of meer dan 2 bishops terwijl en geen queen van die kleur is dit zijn scenario’s dit niet in een echt potje schaken zou gebeuren dit zijn redelijk veel if cases maar het principe is het zelfde hoe groter de onenigheden hoe groter de penalty. Vervolgens zijn er een aantal borden over die niet oneerlijk zijn en ook niet onmogelijk. Van deze borden kijken we naar of **ALEEN** de beste moves op het bord leid tot checkmate in 3. Als de 2e move ook goed is (dit checken we door de score van de positie te kijken na dat de move gespeeld is) word de score van het bord gedeeld door 100 en dat word dan door min(10, second move) gedaan. Dit zorgt er voor dat als je move heel goed is je alsnog 10 penalty word toegevoegd aan het bord. Als de 2e beste move wel slecht krijgt dit bord hier geen penalty. Als je dan naar alle borden in deze generatie kijkt is het bord met de laagste functie het bord wat de beste mate in 3 puzzel heeft. Na 1x is het laagste bord technisch het beste bord maar de penalty zal alsnog erg hoog zijn. dit word vervolgens 15000 keer gedaan met elke generatie gebaseerd of de laagste borden in de generatie ervoor. Dit lijd dan uiteindelijk tot een bord met een super lage penalty dat wiskundig dan een goeie mate in 3 puzzle zou moeten zijn.

### 1.21 GA Testing Params

algorithm\_param = {'max\_num\_iteration': 15000,

'population\_size': 10,

'mutation\_probability': 0.095,

'elit\_ratio': 0.0105,

'crossover\_probability': 0.1,

'parents\_portion': 0.3,

'crossover\_type': 'uniform',

'max\_iteration\_without\_improv': 4200}

### 1.22 GA Program Params

algorithm\_param = {'max\_num\_iteration': 50000,

'population\_size': 20,

'mutation\_probability': 0.05,

'elit\_ratio': 0.01,

'crossover\_probability': 0.9,

'parents\_portion': 0.3,

'crossover\_type': 'two\_point',

'max\_iteration\_without\_improv': 5000}

## Ontwerpschets/wireframes

**N.V.T.**

# Technisch ontwerp

## 2.1 Doel

In het Technisch ontwerp beschrijf ik de werking en gebruik van de applicatie

## 2.2 Achtergrond

**N.V.T.**

Het project was al goed genoeg voordat ik mijn exames aanvroeg dus kreeg ik nog een extra opdracht om mijn documentatie te maken en voormalig gebruikte ik de FEN notatie van schaak borden en stopte dat vervolgens in een de browser om vervolgens aan de docenten de positie te laten zien. Dit heb ik als opdracht nog verandert dat de positie meteen in de zelfde aplicatie komt te staan.  
2.3 Infrastructuur   
Om momenteel de aplicatie te kunnen gebruiken heb je een werkende instalatie van python op je device hebben staan. Daarnaast heb je de hierboven genoemde librarys nodigen en de source code van mijn project (GITHUB). En het is ook belangrijk dat de zelfde map structuur van toepassing is maar als dit project van github cloned zal dat geen probleem zijn.  
2.4 Netwerk   
N.V.T.

ik had eerst alleen een verbinding met de stock Fish api maar voor het optimaliseren van het programma draait stockfish nu local dus is er geen sprake meer van netwerk of servers

<Geef hier een visueel overzicht van het netwerk. Waar staan de routers, de firewalls en  
de servers? Hoeveel bandbreedte is er beschikbaar? Geef ook duidelijk aan waar>  
systeemgrenzen worden overschreden>

2.5 Beveiligingsaspecten  
Ik zie weinig beveiliging problemen want er in met niks een verbinding of database connectie misschien zouden er via de librarys malicious code in contact kunnen komen met de applicatie maar anders dan dat is er alleen pysiek een risico en dat is er bijna altijd.

## 2.6 wettelijke aspecten

Op het moment dat deze applicatie is gemaakt was alles opensource en copyright free. Verder is alle source code origineel en komt het anders uit opensource documentatie

## 2.7 beperkingen

Momenteel is de applicatie extreem beperkt op snelheid een snelle generatie kost 5 a 7 minuten en een indepth tegen een kwartier. Dit is ondanks redelijk up-to-date hardware en een beetje optimalisatie. Dat kan makkelijk opgelost worden kost alleen veel geld en kennis maar het is niet onmogelijk

2.8 Hardware   
<device met python>

<device met alle python library’s>  
2.9 Software  
- Python

- Source code

- stockfish-windows-2022-x86-64-avx2.exe

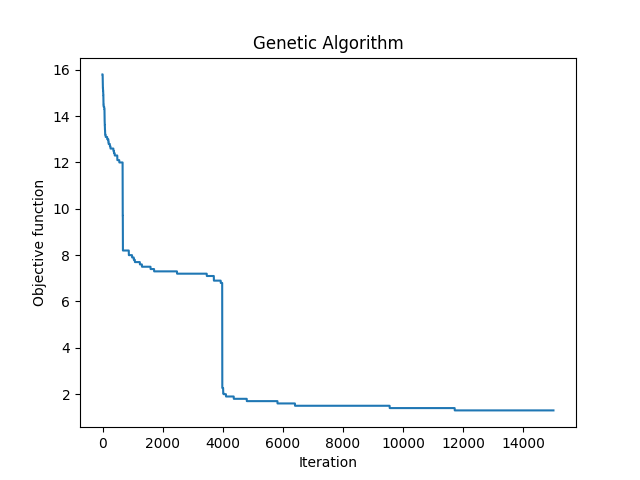
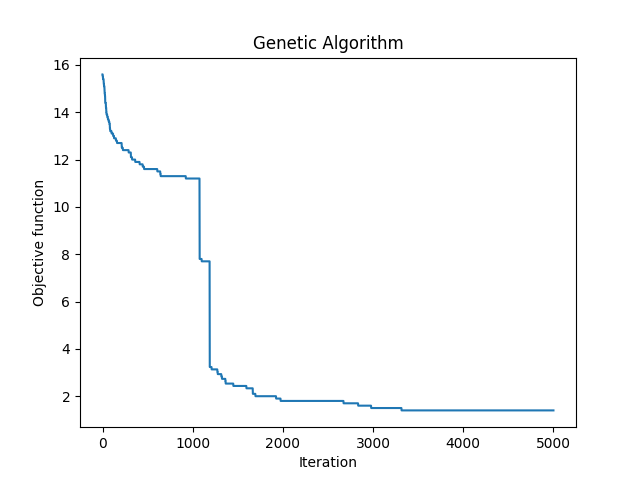
3.0 Interactie  
Programmeertalen

python– hele aplicatie

# 4.0 Iteratie

4.1 Functie responsieveIk heb de params het meest getest maar ook een aantal manieren van chess notaties gebrobeerd en ook veel van de piece penalty’s contant verandert tot dat het meer op echte schaak potjes zou lijken. Er was ook wat heen en weer tussen 2 point, single point of uniform crossovver types.

4.2 TestscenarioBij de meeste van deze test was het een nummer veranderen dan 5 a 15 minuten wachten op het resultaat van de fitness functie en deze elke iteration bijhouden. Hier kreeg ik dan grafieken van die ik achter af met elkaar kon vergelijken welke snellere vooruitgang had



4.3 Testinput

Test input was voornamelijk google wat andere mensen gebruikte voor snelle en rendabele resultaten en als het over specifiekere variable vaak laag beginnen dan dan met 1, 0.1 of 0.01 increments om hoog te gaan

4.4 Verwachte werking/outputik had meestal geen verwachte werking er is alleen een variable in algorithm\_param dat heet max\_interation\_without\_improv dit zorgde er simpel gezegd voor dat elke keer als mijn veranderingen er voor zorgde dat ik x aantal keer bleef vasthangen zonder vooruitgang het programma automatisch stopte en me een warning gaf dit zorgde er voor dat het soms wat sneller ging maar hielp niet bij de correct variabel vinden

4.5 Werkelijke werking/output10 van de 10 keer geeft het de verwachte uitput allen soms lukt het niet om een oplossing binnen de tijd te vinden omdat de eerste generatie random gegenereerd wordt dit lijdt dus altijd tot dingen waar je geen controle over hebt. Maar als tijd geen probleem is lukt het altijd

4.6 Overzicht testomgevingvs code break points en log console, ook meestal vereiste mijn code een set back als het fout was gegaan dus dat deed in gewoon met git.

4.7 Planning testactiviteitener is niet meer test activiteit nodig want heb tot nu toe na tijd geen problemen met mijn applicatie gevonden

4.8 Conclusie testen  
had mijn doel groep van mensen waar ik de puzzels op test kunnen vergroten allen is in de moeilijkheid van de puzzel lastig te bepalen dus niet iedereen heeft genoeg verstand van schaken om sommige puzzels op te lossen. Dit leidde tot at ik vaak zelf handmatig posities moest na checken alleen kan ik niet een manier daaromheen bedenken want ik gebruikte zelf al een hoopt tools om het sneller te laten verlopen.

# Conclusie

# Bronvermelding

Hieronder vind je de bronvermelding.

**Numpy**

**https://numpy.org/**

**Chess.com**

**https://www.ches.com**

**Chess library**

**https://python-chess.readthedocs.io/en/latest/**

**Stockfish**

**https://stockfishchess.org/**

**GA**

**https://pypi.org/project/geneticalgorithm/**

# Bijlage I - reflectie STARR

**S – Situatie**

1. Wat ging eraan vooraf?
2. Wanneer speelde dit project?
3. Wie waren erbij betrokken?
4. Met wie werkte jij hieraan samen?
5. Waar speelde het zich af?
6. Wat was jouw rol binnen het project?

**T – Taak**

1. Wat was jou functie?
2. Waarom moest juist jij dit doen?
3. Welke doelen had jij?

**A - Actie**

1. Wat deed jij toen?
2. Hoe reageerde jij
3. Waarom pakte jij het op die manier aan?
4. Wat deed je eerst, wat kwam daarna?

**R – Resultaat**

1. Wat was het resultaat van jou actie?
2. Hoe reageerden de anderen hierop?
3. Hoe is het afgelopen?
4. Zijn er nog problemen voorgekomen/teruggekeerd?

**R – Reflectie**

1. Zou je een zelfde probleem/actie volgende keer zo aanpakken?
2. Wat zou je volgende keer anders doen?
3. Heb jij daarna een vergelijkbare situatie meegemaakt? Zo ja wat heb jij toen anders gedaan?
4. Kun je een voorbeeld geven waarin een dergelijke actie verkeerd uitpakte?

# Bijlage II – Planning

# Bijlage III - Logboek

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **VAN  (begin tijd)** | **TOT**  **(eind tijd)** | **Totaal**  **(minuten)** | **Datum**  **(dag)** | **WERKZAAMHEDEN** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |