Friedrich-Schiller-Universität Jena
Wirtschaftswissenschaftliche Fakultät
Lehrstuhl für Wirtschaftsinformatik
Prof. Dr. J. Ruhland

Entwicklung eines Prototyp für einen Estimation of Distribution Algorithmen in Python

Projektarbeit

im Studiengang Wirtschaftsinformatik

WS 2020/2021

Eingereicht von:	Martrikelnummer:	Betreuer:
Dario Dubberstein	180296	Wiss. Assistent Sven Gehrke

Schmölln, 17. Januar 2021

I Inhaltsverzeichnis

II Abbildungsverzeichnis

III Nomenklatur

Abkürzungen:

E

EDA	Estimation of Distribution Algorithmen
EA	Evolutionäre Algorithmen

G

GA	Genetische Algorithmen

P

PMBGA	Probabilistische, modellbildende, genetische Algorithmen

S

SUS	Stochastik Universal Sampling

1 Einleitung

1.1 Hintergrund

Evolutionstechniken sind eines der erfolgreichsten Paradigmen auf dem Gebiet der Optimierung. Die Estimation of Distribution Algorithmen (EDAs) zielen darauf ab, die Wahrscheinlichkeitsverteilung der Qualitätslösungen für das zugrunde liegende Problem explizit zu modellieren. Durch iteratives Filtern nach Qualitätslösungen aus konkurrierenden Lösungen nähert sich das Wahrscheinlichkeitsmodell schließlich der Verteilung globaler optimaler Lösungen an. Im Gegensatz zu klassischen evolutionären Algorithmen (EAs) ist das EDA-Framework flexibel und kann mit der Abhängigkeit zwischen Variablen umgehen, was klassischen EAs normalerweise Schwierigkeiten bereitet. Der Erfolg von EDA beruht auf einer effektiven und effizienten Erstellung des Wahrscheinlichkeitsmodells.

1.2 Ziel der Arbeit

Im Rahmen dieser Arbeit soll ein Programm entwickelt werden, welches einen Estimation of Distribution Algorithmus realisiert. Dabei gilt darauf zu achten das dieses für die Veranschaulichung von Beispielen für die Lehre eingesetzt werden kann. Weitere Kriterien sind eine „saubere“ Implementierung, sodass dieses Programm für weitere Beispiele oder andere Funktionalitäten leicht erweiterbar ist.

1.3 Vorgehen und Aufbau

Kapitel 2 bietet einen kurzen Exkurs zu genetischen Algorithmen. Es werden grundsätzliche Begrifflichkeiten, Methoden und Vorgehen genauer erläutert. Dabei werden auch Schritte aufgezeigt die evolutionären Algorithmen gemeinsam haben und zeigt am Ende auf, wie sich diese zu einem Estimation of Distribution Algorithmus unterscheiden.

Im letzten Kapitel wird der Aufbau des zu entwickelten Programms dargestellt und welche Methoden für die einzelnen Schritte des Algorithmus implementiert wurden. Es han-

delt sich dabei aber nicht um eine Dokumentation, die jeden Programmierabschnitt genaustens protokolliert. Dies wurde bereits im Programmcode in Form von Kommentaren abgedeckt.

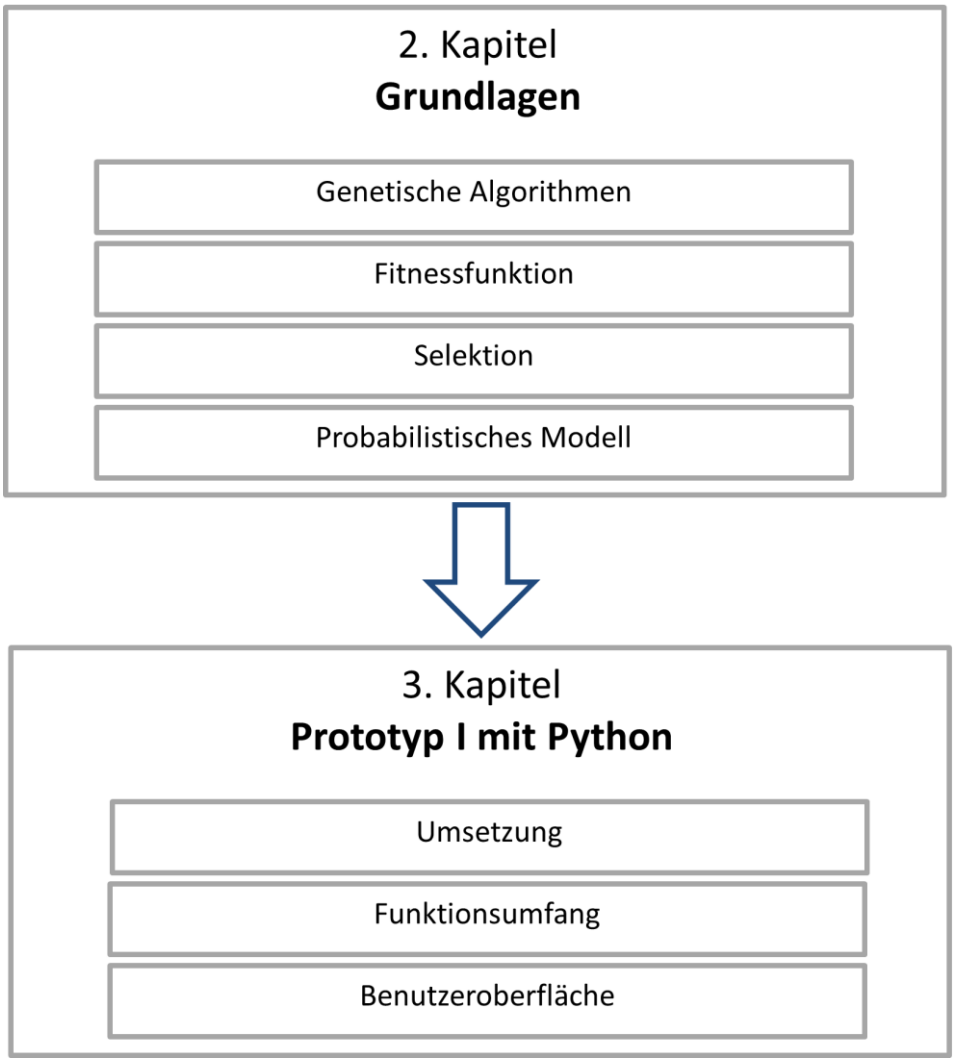


Abbildung 1.1 Schematischer Aufbau der Seminararbeit (Eigene Darstellung)

2 Grundlagen

Die Estimation of distribution Algorithmen (EDAs), manchmal auch als probabilistische, modellbildende, genetische Algorithmen (PMBGAs) bezeichnet,[1] gehören zu den evolutionären Algorithmen. Diese sind eine Klasse populationsbasierter, stochastischer, metaheuristischer Optimierungsverfahren. Ein Evolutionäre Algorithmen (EA) verwenden Mechanismen, die von der biologischen Evolution inspiriert sind, wie Reproduktion, Mutation, Rekombination und Selektion.[2] Im Fall von EDA handelt es sich um stochastische Optimierungsmethoden, die die Suche nach dem Optimum leiten, indem explizite probabilistische Modelle vielversprechender Kandidatenlösungen erstellt und abgetastet werden. Die Optimierung wird als eine Reihe von inkrementellen Aktualisierungen eines Wahrscheinlichkeitsmodells angesehen, beginnend mit dem Modell, das einen nicht informativen Prior über zulässige Lösungen codiert, und endend mit dem Modell, das nur die globalen Optima generiert. [3][4]

EDAs gehören zur Klasse der evolutionären Algorithmen. Der Hauptunterschied zwischen EDAs und den meisten herkömmlichen evolutionären Algorithmen besteht darin, dass evolutionäre Algorithmen neue Kandidatenlösungen unter Verwendung einer impliziten Verteilung generieren, die von einem oder mehreren Variationsoperatoren definiert wird, während EDAs eine explizite Wahrscheinlichkeitsverteilung verwenden, die von einem Bayes'schen Netzwerk, einer multivariaten Normalverteilung oder einer anderen Modellklasse codiert wird.

Die Verwendung expliziter Wahrscheinlichkeitsmodelle bei der Optimierung ermöglichte es EDAs, Optimierungsprobleme zu lösen, die für die meisten konventionellen evolutionären Algorithmen und traditionellen Optimierungstechniken notorisch schwierig waren, wie z. B. Probleme mit einem hohen Grad an Epistase. Der Vorteil von EDAs besteht jedoch auch darin, dass diese Algorithmen eine Reihe von Wahrscheinlichkeitsmodellen zur Verfügung stellen, die viele Informationen über das zu lösende Problem enthalten. Diese Informationen können wiederum verwendet werden, um problemspezifische Nachbarschaftsoperatoren für die lokale Suche zu entwerfen, zukünftige Läufe von

[1] (Vikhar, 2016)
[2] (Tan, Shi, & Tan, 2010)
[3] (Larrañaga & Lozano, 2002)
[4] (Lozano, Larrañaga, Inza, & Bengoetxe, 2006)

EDAs auf ein ähnliches Problem auszurichten oder um ein effizientes Rechenmodell des Problems zu erstellen.[5]

2.1 Fitnessfunktion

Insbesondere in den Bereichen genetische Programmierung und genetische Algorithmen wird jede Entwurfslösung üblicherweise als eine Folge von Zahlen dargestellt (als Chromosom bezeichnet). Nach jeder Testrunde oder Simulation besteht die Idee darin, die n schlechtesten Entwurfslösungen zu löschen und n neue aus den besten Entwurfslösungen zu züchten. Daher muss jeder Konstruktionslösung eine Leistungszahl verliehen (Fitness) werden, um anzugeben, wie nahe sie an der Erfüllung der Gesamtspezifikation (Zielfunktion) liegt.[6] Wie auch evolutionäre Algorithmen haben Fitnessfunktionen ein biologisches Vorbild, die biologische Fitness, die den Grad der Anpassung eines Organismus an seine Umgebung angibt. Bei evolutionären Algorithmen beschreibt die Fitness eines Lösungskandidaten, wie gut er das zugrunde liegende Optimierungsproblem löst. Die Fitnessfunktion berechnet aus den Eigenschaften eines Lösungsversuchs, wie gut sich dieses „Individuum“ bzgl. des gestellten Problems als Lösung eignet.[7]

Eine Fitnessfunktion muss nicht zwangsläufig einen absoluten Wert berechnen können, da es oft reicht, Kandidaten zu vergleichen, um den besseren auszuwählen. Eine relative Angabe der Fitness (Kandidat a ist besser als b) genügt für viele Selektionsverfahren.[8]

2.2 Selektion

Elternauswahl ist der Prozess der Auswahl von Eltern, die sich paaren und neu kombinieren, um Nachkommen für die nächste Generation zu schaffen. Die Auswahl der Eltern ist sehr wichtig für die Konvergenzrate der EA, da gute Eltern den Individuen zu besseren und fitteren Lösungen führen.

Es sollte jedoch darauf geachtet werden, dass in wenigen Generationen nicht verhindert wird, dass eine äußerst geeignete Lösung die gesamte Bevölkerung übernimmt, da dies

[5] (Larrañaga & Lozano, 2002)
[6] (Carr, 2014)
[7] (Jansen, 2013)
[8] (Back, Fogel, & Michalewicz , 2000)

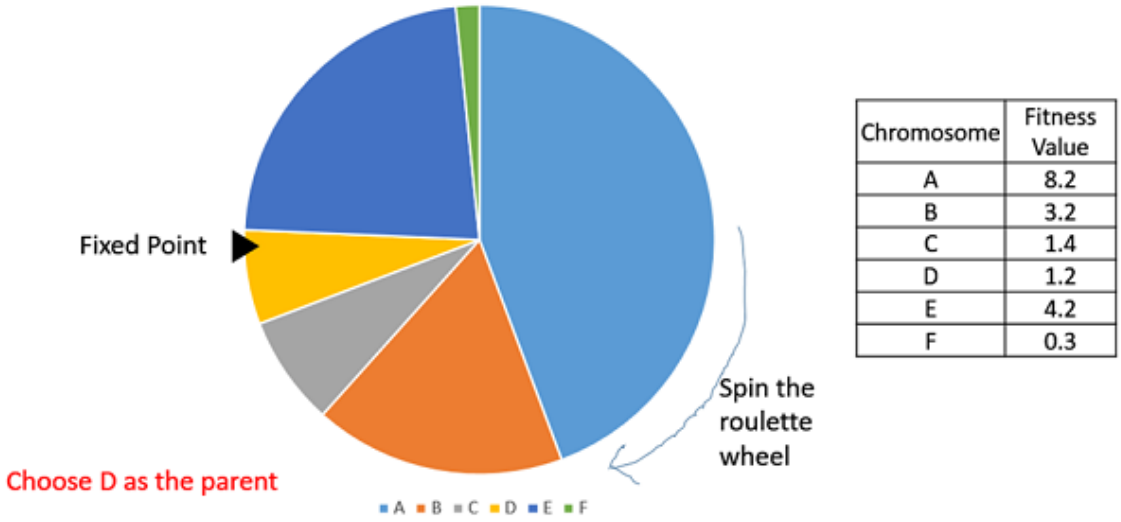
dazu führt, dass die Lösungen im Lösungsraum nahe beieinander liegen, was zu einem Verlust an Diversität führt. Die Aufrechterhaltung einer guten Vielfalt in der Bevölkerung ist äußerst wichtig für den Erfolg einer EA. Diese Aufnahme der gesamten Bevölkerung durch eine äußerst geeignete Lösung wird als vorzeitige Konvergenz bezeichnet und ist eine unerwünschte Erkrankung bei einer EA.

2.2.1 Fitness Proportionale Auswahl

Die proportionale Fitnessauswahl ist eine der beliebtesten Methoden zur Elternauswahl. Dabei kann jedes Individuum mit einer Wahrscheinlichkeit Eltern werden, die proportional zu seiner Fitness ist. Daher haben fittere Individuen eine höhere Chance, ihre Merkmale zu paaren und an die nächste Generation weiterzugeben. Eine solche Auswahlstrategie wirkt einen Auswahldruck auf die fitteren Personen in der Bevölkerung aus und entwickelt im Laufe der Zeit bessere, im Bezug zur Zielfunktion, Personen. Visualisieren lässt sich dieses durch ein kreisförmiges Rad. Das Rad wird in n Stücke unterteilt, wobei n die Anzahl der Personen in der Bevölkerung ist. Jedes Individuum erhält einen Teil des Kreises, der proportional zu seinem Fitnesswert ist.

2.2.1.1 Roulette-Radauswahl

Bei einer Roulette-Radauswahl wird das kreisförmige Rad wie zuvor beschrieben geteilt. Wie in Abbildung 2.1 dargestellt wird ein fester Punkt am Radumfang gewählt und das Rad gedreht. Der Bereich des Rads, der vor dem Fixpunkt liegt, wird als übergeordnetes Element ausgewählt. Für den zweiten Elternteil wird der gleiche Vorgang wiederholt.



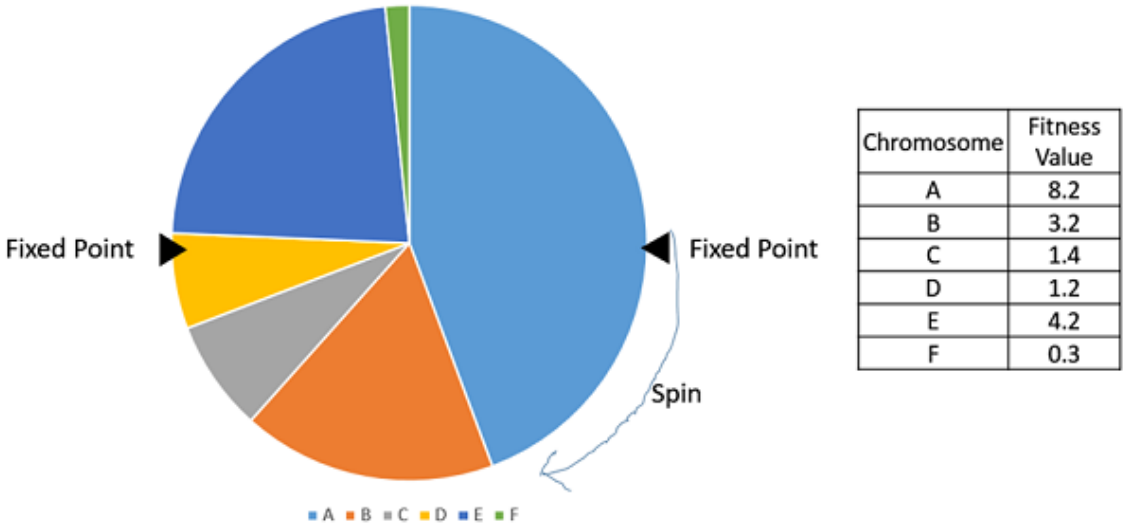
Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Abbildung 2.1 Roulette-Radauswahl Quelle: https://www.tutorialspoint.com/

Anhand der Abbildung lässt sich zudem erkennen, dass eine fitte Person eine größere Teilmenge auf dem Rad hat und daher eine größere Chance hat, vor dem festen Punkt zu landen, wenn das Rad gedreht wird. Daher hängt die Wahrscheinlichkeit, eine Person auszuwählen, direkt von ihrer Fitness ab.

2.2.1.2 Stochastik Universal Sampling (SUS)

Die stochastische universelle Abtastung ist der Auswahl des Roulette-Rads ziemlich ähnlich. Statt nur eines festen Punkts werden jedoch mehrere feste Punkte bestimmt, wie in der folgenden Abbildung gezeigt. Daher werden alle Eltern in nur einer Drehung des Rades ausgewählt. Ein solches Anordnung führt für Individuen mit hoher Fitness immer noch zu einer hohen Wahrscheinlichkeit gewählt zu werden. Hingegen zum normalen Roulette-Radverfahren ermöglicht das SUS eine höhere Wahrscheinlichkeit Individuen mit einem niedrigeren Anteil am Rad gewählt zu werden. Dies ist inhärent nicht schlecht, da einer höheren Vielfalt an genetischen Informationen zu einer möglichen Vermeidung einer Fokussierung auf ein lokales Optimum führen kann.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Abbildung 2.2 Stochastik Universal Sampling Quelle: https://www.tutorialspoint.com/

Es ist zu beachten, dass Fitness-proportionale Auswahlmethoden in Fällen, in denen die Fitness einen negativen Wert annehmen kann, nicht funktionieren.

2.2.2 Turnierauswahl

Bei der Turnierauswahl werden mehrere "Turniere" unter wenigen Personen (oder "Chromosomen") durchgeführt, die zufällig aus der Bevölkerung ausgewählt werden. Der Gewinner jedes Turniers (derjenige mit der besten Fitness) wird für den Crossover ausgewählt. Der Auswahldruck, ein probabilistisches Maß für die Wahrscheinlichkeit der Teilnahme eines Chromosoms am Turnier basierend auf der Größe des Teilnehmerauswahlpools, kann leicht durch Ändern der Turniergröße angepasst werden. Der Grund dafür ist, dass schwache Personen bei einer größeren Turniergröße eine geringere Chance haben ausgewählt werden, denn wenn eine schwache Person für ein Turnier ausgewählt wird, besteht eine höhere Wahrscheinlichkeit, dass sich auch eine stärkere Person in diesem Turnier befindet. Die Turnierauswahl bietet gegenüber alternativen Auswahlmethoden

für genetische Algorithmen mehrere Vorteile: Sie ist effizient zu codieren, arbeitet mit parallelen Architekturen und ermöglicht eine einfache Anpassung des Auswahldrucks.

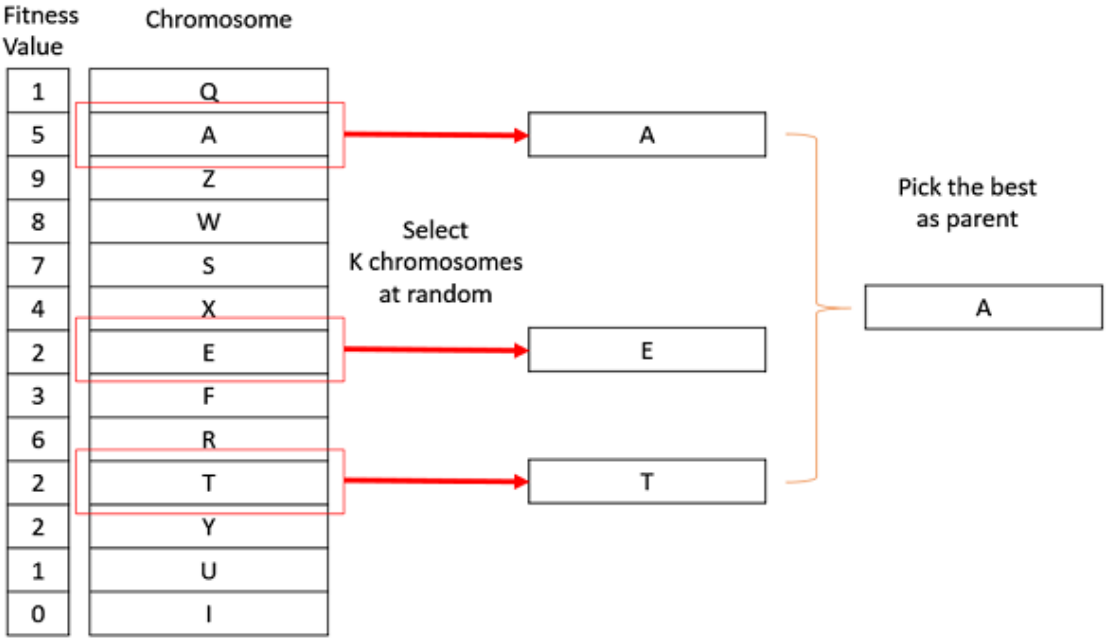


Abbildung 2.3Tunierauswahl Quelle: https://www.tutorialspoint.com/

2.2.3 Zufällige Auswahl

Bei dieser Methode werden zufällig Eltern aus der bestehenden Bevölkerung ausgewählt. Es besteht kein Selektionsdruck gegenüber fitteren Personen und daher wird diese Strategie normalerweise vermieden.

2.3 Berechnung der nächsten Generation

Für die Berechnung der sog. Kind Generation werden bei EA typischerweise verschiedene Verfahren verwendet, um die Informationen der Chromosomen der Eltern zu mischen und dadurch neue und eventuell fittere Individuen zu erstellen. Wie in der Natur werden dafür Methoden für einen Crossover für jedes Allel angewendet und zum Teil auch Wahrscheinlichkeiten für eine Mutation hinterlegt. Im Fall der Estimation of Distribution Algorithmen kommt für den Austausch von Ausprägungen eines Allels ein probabilistisches Modell zum Einsatz. Wie bei anderen EA wird eine Selektion anhand der Fitness und eines für die Problemstellung passendes Selektionsverfahren die Eltern der nächsten Generation festgelegt. Anschließend wird für jedes Allel aller Chromosomen

eine Wahrscheinlichkeit berechnet. Hierbei wird auch die Wahrscheinlichkeitsverteilung von der Problemstellung abhängig gewählt. Anhand der nun erhaltenen Wahrscheinlichkeiten, für die Ausprägung der Allele, lässt sich eine neue Generation erstellen. Für jede Generation wird somit anhand der Wahrscheinlichkeit für das Auftreten einer genetischen Ausprägung diese weiter gestärkt oder abgeschwächt. Wie auch für andere EA spielt die Berechnung der Fitness eine ausschlaggebende Rolle für eine Annäherung an die Zielfunktion.

3 Realisierung in Python

Für ein anschauliches Beispiels eines Estimation of Distribution Algorithmus wurde ein kleines Programm in Python geschrieben. Python eignet sich hierbei wunderbar aufgrund seiner Fähigkeit auch komplexe Problemstellungen mit kompakten Programmen zu realisieren. Es ist zudem System unabhängig und lässt sich gut in andere Projekte und Programme integrieren. Die Numpy Bibliothek bietet perfekte Funktionalitäten, in Bezug auf Arrays, die für Fitness, Selektion und das probabilistische Modell von Nöten sind.

Eine hoher Flexibilität für eventuelle Erweiterungen oder Änderungen des Funktionsumfanges wurde durch einen funktionsbasierten Aufbau sichergestellt. Dabei wurde darauf geachtet das alle wichtigen Berechnungen in gesonderten Funktionen (Fitness, Selektion, probabilistisches Modell, neue Generation erstellen) durchgeführt werden. Das Programm wurde zudem basierend auf weiteren Eigenschaften in einzelne Skripte unterteilt, darunter Berechnungen, Funktionsaufrufe und Benutzeroberfläche.

Um eine „saubere“ Programmierung zu gewährleisten wurde auf die folgenden Punkte geachtet:

- Kommentare: Es wurde viel Wert auf sinnvolle Kommentare gelegt. Dies enthält auch Beschreibungen, warum ein Befehl in seiner Art durchgeführt wird und bietet zudem eine kompakte Dokumentation des Programmcodes.
- Optische Faktoren: Es wurde kein spezifischer Style Guide eingesetzt, aber es wurde auf saubere und konsistente Einrückung, Codelänge, Funktionsumfang, Codekomplexität, etc. geachtet.
- Namen: Bei allen Variablen, Funktionen wurde eine eindeutige Benennung benutzt, es ist dadurch ersichtlich was eine Funktion macht und was sich hinter den einzelnen Variablen verbirgt.

- Sprache: Sprache wurde einheitlich in Englisch gehalten, dies ermöglicht es auch internationalen Studenten/Mitarbeitern Kommentare/Dokumentation und den Programmcode zu lesen

3.1 Fitness

Die Fitness wird anhand der Summe aller Allele in Bezug zu einem bestmöglichen Chromosom bestimmt. Es wird hierbei keine weitere Gewichtung der Allele berücksichtigt.

3.2 Selektion

Die Wahl der erfolgt anhand der zuvor bestimmten Fitness für jedes Chromosom. Dabei wird eine nach Fitness geordnete Liste erstellt. Für die vorgegebenen Funktionen werden zwei bis drei Eltern gewählt. Bei der Durchführung mit individuellen Parametern kann die Anzahl selbst bestimmt werden.

3.3 Probabilistisches Modell

Die Wahrscheinlichkeit wird basierend auf den übergebenen Eltern berechnet. Aktuell wird dabei die Normalverteilung verwendet. Eine Erweiterung für weitere Verteilungen ist denkbar und dank dem funktionalen Programmaufbau mit wenig Aufwand realisierbar.

Im Rahmen der Arbeit wurde eine erste Erweiterung in Form der Laplace Korrektur implementiert. Diese ermöglicht es Werte, die in einem lokalen Minimum feststecken zu korrigieren und somit die Möglichkeit auch bei „schlechten“ Startwerden ein globales Optimum für die Zielfunktion zu finden.

3.4 Nächste Generation erstellen

Anhand der Wahrscheinlichkeit für jedes Allel werden neue Individuen (Chromosomen) erstellt. Die Anzahl der neuen Individuen ist identisch mit der Menge an Individuen, die für die Selektion zur Verfügung standen. Jede Generation hat somit denselben Umfang

an Individuen. Was sich jede Generation ändert ist die Wahrscheinlichkeit für die Ausprägung der einzelnen Allele, bis ein Optimum an der Stelle des Chromosoms erreicht wurde.

3.5 Funktionen

In der aktuellen Version stehen dem Benutzer vier Funktionen zur Verfügung. Unterschiede gibt es in der Interaktivität zwischen Benutzer und Programm nur zwischen der ersten und allen anderen Optionsmöglichkeiten. Bei der ersten Option steht es dem Anwender frei alle Parameter selbst zu bestimmen und sich somit eine individuell erstellte Problemstellung berechnen zu lassen. Option drei und vier orientieren sich an der Vorlesungsfolie zum Thema EDA und Optimierung. Durch die feste Vorgabe der Parameter eignet sich diese Option als Veranschaulichung für das gezeigte Beispiel.

3.6 Benutzeroberfläche

Die Bedienung erfolgt über eine einfache Oberfläche. Alle im vorherigen Kapitel beschriebenen Optionen können über einen Radiobutton selektiert werden. Für die individuellen Parameter stehen dem Benutzer entsprechende Eingabefelder zur Verfügung. Für die Ergebnisse wurde eine Vektorbasierte Ausgabe für die Chromosomen aller Individuen gewählt. Neben der Ausgabe der Startpopulation gibt es eine weitere für die wichtigsten Informationen aller Generationen, darunter befinden sich die selektierten Eltern, Wahrscheinlichkeit der Allele und beste Fitness. Um den Verlauf der Fitness über alle Generationen zu veranschaulichen wird nach Ausführen des Programms eine grafische Darstellung dessen eingebunden.

Es wurde bewusst auf Menüs und Popupfenster verzichtet, damit der Anwender die ausgewählte Funktion, alle Parameter und die Ergebnisse auf einem Blick hat. Dies wurde mit dem Hintergrund der Verwendung für die Lehre gewählt.

4 Schlussbetrachtung

Eine einfache Ausprogrammierung für einen Estimation of Distribution Algorithmus ist mit Python schnell zu realisieren. Unter Berücksichtigung der Lehre, Wiederverwendbarkeit und Erweiterbarkeit des Programmcodes musste einiges an Mehraufwand investiert werden. Unter Berücksichtigung der in Kapitel 3 genannten Punkte, in Bezug auf „sauberes“ programmieren, konnte direkt von Projektstart an ein ordentliches Programm gewährleistet werden. Was sich besonders bei der Erweiterung der Bedienbarkeit schnell bemerkbar machte. Erste Versionen starteten mit der Befehlszeile und endeten mit einer Benutzeroberfläche für die aktuelle Version.

IV Literaturverzeichnis

Back, T., Fogel, D., & Michalewicz , Z. (2000). Evolutionary Computation 2: Advanced Algorithms and Operators.

Carr, J. (2014). An Introduction to Genetic Algorithms.

Jansen, T. (2013). Analyzing Evolutionary Algorithms: The Computer Science Perspective.

Larrañaga, P., & Lozano, J. (2002). *Estimation of Distribution Algorithms a New Tool for Evolutionary Computation.*

Lozano, J. A., Larrañaga, P., Inza, I., & Bengoetxe, E. (2006). *Towards a new evolutionary computation advances in the estimation of distribution algorithms. .*

Tan, Y., Shi, Y., & Tan, K. (2010). Advances in Swarm Intelligence.

Vikhar, P. (2016). Evolutionary algorithms: A critical review and its future prospects. In *Proceedings of the 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)* (S. 261–265).

V Anhang

Eidesstattliche Erklärung

Ich erkläre hiermit eidesstattlich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung rechtliche Folgen haben kann.

_____ _____

Ort, Datum Vorname und Nachname

Veröffentlichung der Arbeit

Ich stimme der Veröffentlichung der Arbeit im Rahmen von Forschung und Lehre an der Friedrich-Schiller-Universität Jena zu.

_____ _____

Ort, Datum Vorname und Nachname