Doku - Genetische Algorithmen

1 Allgemeines

Genetische Algorithmen sind eine heuristische Suchmethode, um Optimierungsprobleme zu lösen. Sie basieren dabei auf den prinzipiellen Grundlagen der Darwin'schen Evolutionstheorie (deshalb Teilmenge innerhalb der evolutionären Algorithmen) und beziehen sich dabei vor allem auf die Ideen der natürlichen Selektion sowie der genetischen Rekombination („survival of the fittest"). [1]

1.1 Komponenten

Ähnlich zum Ablauf im biologischen Modell wird auch bei genetischen Algorithmen Gebrauch von Populationen, Chromosomen und Genen gemacht.

1.1.1 Gen

Ein Gen bildet im Rahmen der genetischen Algorithmen einen Teilbaustein von „genetischen Informationen". Es trägt dabei also Teilinformationen, die zur Entstehung einer potenziellen Lösung benötigt wird, da mehrere Gene analog dem natürlichen Prozess zu einer möglichen Lösungsausprägung zusammengesetzt werden.

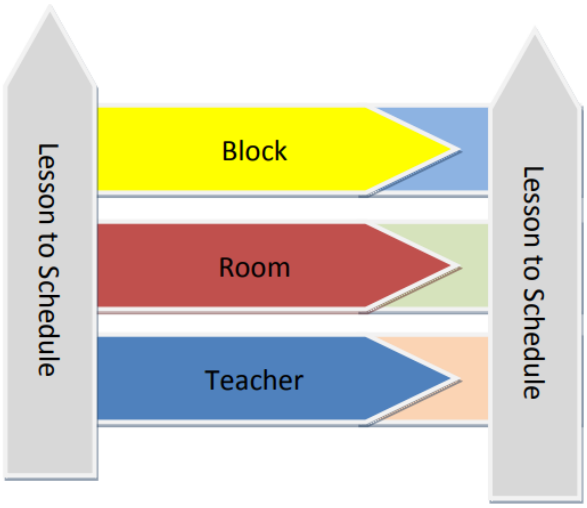

Abb. 1: Beispiel für ein Gen

1.1.2 Chromosom

Ein Chromosom setzt sich aus einer linearen Anordnung mehrerer Gene zusammen und bildet somit in seiner Gesamtheit eine mögliche Lösungsausprägung im jeweils eingesetzten Kontext ab.

1.1.3 Population

Eine Population besteht aus einer Vielzahl an Chromosomen, welche sich untereinander jeweils unterscheiden, sie sind also entsprechend „unterschiedlich fit".

1.2 Ablauf

Der allgemeine Ablauf eines genetischen Algorithmus ist folgendermaßen aufgebaut:

1. Generierung einer Initialpopulation von Chromosomen.
2. Berechnung der „Fitness“ eines jeden Chromosoms.
3. Auswahl von Chromosomen aus der Population zur Fortpflanzung.
4. „Kindschromosomen“ (Nachwuchs) erschaffen durch Fortpflanzung.
5. Ausführung willkürlicher Mutationen an den Genen.
6. Wiederholung Schritt 3-5, bis eine neue Generation entstanden ist.
7. Abschluss, wenn Algorithmus eine optimale/ausreichend optimale Lösung gefunden hat oder sich der Score der derzeit optimalen Lösung nicht mehr ändert, auch nach einer vorher festgelegten Anzahl an Nachfolgegenerationen.

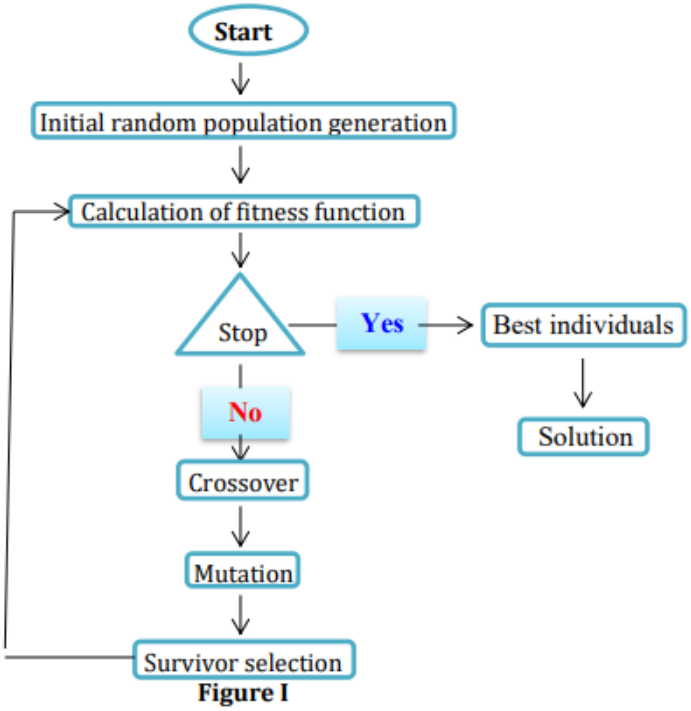

Abb. 2: Einfaches Ablaufdiagramm genetischer Algorithmus

1.2.1 Initiale Population

Der Algorithmus erstellt eine initiale Population, welche aus einer Vielzahl an verschiedenen Chromosomen besteht. Jedes Chromosom stellt dabei eine mögliche Lösung für das vorliegende Problem dar.

1.2.2 Bestimmung der Fitness

Für jedes Chromosom wird nun seine „Fitness“ bestimmt. Dies geschieht mithilfe einer Fitnessfunktion. Sie ist repräsentativ für die „Umwelt“ der einzelnen Chromosomen/Individuen der Population. Aus diesem Grund wird über sie auch bestimmt, wie „überlebensfähig“ ein

Chromosom in der Umwelt ist, d.h., es wird bestimmt, wie gut ein bestimmtes Chromosom abschneidet, relativ zur optimalen Lösung.

1.2.3 Auswahl Chromosomen zur Fortpflanzung

Nachdem die „Fitness“ für jedes Chromosom der Population bestimmt wurde, soll es nun zur Fortpflanzung kommen. Analog zum natürlichen Selektionsprozess, haben auch hier die Individuen/Chromosomen, die „fitter“ sind, eine höhere Überlebens- und damit Fortpflanzungschance. Zur Auswahl können hierbei mehrere Selektionsprozesse genutzt werden.

1.2.3.1 Glücksrad

Wie bei einem Glücksrad kann bei der Auswahl zweier Chromosomen als Fortpflanzungspartner folgendermaßen vorgegangen werden: Die zuvor berechneten Fitnesswerte aller Chromosomen werden normalisiert, sodass die Chromosomen mit der höchsten Fitness auch die höchste normalisierte Fitness haben und somit auch die höchste Wahrscheinlichkeit, im Fortpflanzungsprozess ausgewählt zu werden. Wichtig dabei ist, auch den „wenig fitten“ Chromosomen eine Chance zur Fortpflanzung zu gewähren, da auch diese für die optimale Lösung kostbares Erbgut innerhalb ihrer Gene tragen können. Im Anschluss an die Normalisierung werden basierend auf den jeweiligen Wahrscheinlichkeiten zufällig zwei Chromosomen für die Fortpflanzung ausgewählt.

1.2.3.2 Turnierauswahl

Hierbei wird eine festgelegte Anzahl an Chromosomen aus der Population entnommen, die an einem Turnier teilnehmen müssen. Es treten dabei immer eine bestimmte Anzahl an Chromosomen gegeneinander an, wobei analog zur Glücksrad-Methode ihre Fitnesswerte im direkten Vergleich zueinander normalisiert werden und eins der beiden zufällig ausgewählt wird. Der Gewinner des Duells kommt eine Runde weiter. Dies geschieht so lange, bis am Ende ein Gewinner übrigbleibt. Im Anschluss muss noch ein Partner über ein zweites Turnier bestimmt werden.

1.2.4 Fortpflanzung

Die eigentliche Fortpflanzung erfolgt, indem jeweils einzelne Genteile der beiden Chromosomen, die sich durchgesetzt haben, neu kombiniert werden, und somit ein neues „Kind“-Chromosom entsteht. Hierbei gibt es zwei Vorgehensmöglichkeiten:

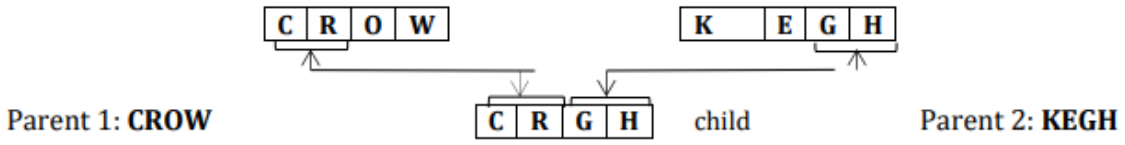

Abb. 3: 50/50 Rekombination

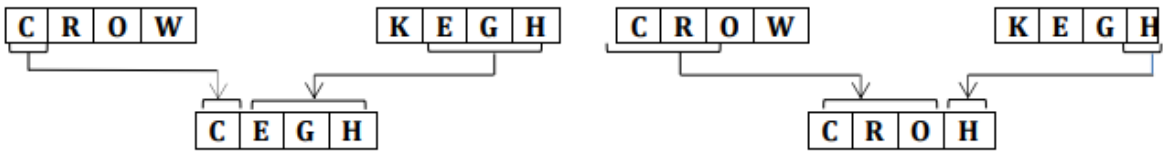

Figure VI: Selecting a random point.

Figure VII:

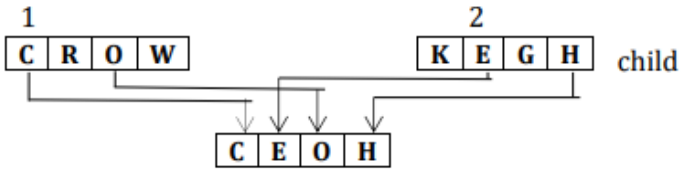

Abb. 4: Zufällige Rekombination

Möglichkeit 1 (vgl. Abb. 3) stellt das „one-point crossover“ dar. Hierbei werden jeweils 50% des Erbgutes aus einem Elternchromosom entnommen. Abb. 4 spiegelt die Rekombination in Form zufälliger Auswahl der Gene für das Kindchromosom wider. Innerhalb der Implementierung eines genetischen Algorithmus wird die zufällige Rekombination dabei präferiert, da hierdurch mehr Diversität in der Folgegeneration gegeben ist.

1.2.5 Mutation

Um zusätzliche Abwechslung in der Folgegeneration zu gewährleisten, treten mit einer vorab definierten Wahrscheinlichkeit Mutationen in den neuen Chromosomen auf. Es besteht also die Chance, dass ein oder auch mehrere Gene des neuen Chromosoms zufällig ausgewählt und verändert werden.

2 Das Stundenplanproblem

Das Stundenplanproblem ist ein typisches Optimierungsproblem. Ein genetischer Algorithmus, der für dieses Problem implementiert werden soll, basiert grundlegend auf den Dimensionen Dozenten, Räume, Zeitslots und Veranstaltungen/Module mit einer zugehörigen Menge an Studiengängen, welche das Modul belegen. Die Optimierung eines Stundenplans ist dabei an eine Vielzahl an Bedingungen geknüpft.

2.1 Hard Constraints

Hard Constraints bilden die Bedingungen, die erfüllt werden **müssen**, damit ein möglicher Stundenplan in Realität umgesetzt werden kann. Bezogen auf das Stundenplanproblem an der FH ergeben sich folgende harte Constraints:

1. Zu einem bestimmten Zeitslot darf in einem bestimmten Raum maximal eine Veranstaltung zu Zeit stattfinden.
2. Ein Dozent darf zu keinem Zeitpunkt mehr als eine Veranstaltung pro Zeitslot halten.
3. Es dürfen keine zwei Veranstaltungen, die zu einem Studiengang (jeweiliges Semester) gehören, im gleichen Zeitslot stattfinden.
4. Ein Raum muss die Anforderungen bzgl. der Raumart erfüllen, die für eine Veranstaltung notwendig sind.
5. Eine Veranstaltung muss in einem Raum abgehalten werden, die eine ausreichende Anzahl an Sitzplätzen bietet.
6. Dozentenwünsche, die erfüllt werden **müssen**, sind auch zu erfüllen.

2.2 Soft Constraints

Neben den Hard Constraints gibt es je nach spezifischem Anwendungsfall zudem eine Reihe an Soft Constraints. Diese müssen nicht zwangsläufig alle erfüllt werden, der Fitnesswert der Lösung verbessert sich aber mit jeder erfüllten Bedingung. Auf die FH bezogen, könnten folgende Soft Constraints mit in der Auswertung betrachtet werden:

1. Dozenten sollten nicht mehr als 3 (?) Veranstaltungen am Stück halten müssen.
2. Studenten einer Studiengangausprägung sollten nicht mehr als 3-4 (?) Veranstaltungen am Stück besuchen müssen.
3. Veranstaltungen einer Stundenplan-Ausprägung an einem Tag sollten nicht zu weit auseinander sein (z.B. 8:00-9:15 und 17:00-18:15).
4. Dozentenwünsche, die **nicht zwangsläufig** erfüllt werden müssen.
5. Die Raumgröße sollte der Veranstaltungsgröße angemessen sein (z.B. Veranstaltungen mit 20 Teilnehmer/innen nicht im Audimax, obwohl theoretisch möglich)
6. Möglicherweise Veranstaltungen im frühesten/spätesten Zeitslot minimieren

2.3 Fitnessfunktion

Eine effiziente Fitnessfunktion für das Stundenplanproblem aufzustellen ist das Herzstück des genetischen Algorithmus. Hier bestimmt sich, wie groß die Aussicht auf eine erfolgreiche Implementierung im Endeffekt sein wird. Dabei gibt es mehrere Ansätze, die wählbar sind. Ein Ansatz wird im Folgenden näher erläutert:

Zu jedem fertig generierten Stundenplan lässt sich über die Fitnessfunktion ein Ergebnis in Form eines errechneten Fitnesswertes bestimmen. Zu Beginn der Auswertung beträgt dieser Wert initial 0. Um nun zu bestimmen, wie gut ein Stundenplan tatsächlich ist, muss jede platzierte Veranstaltung analysiert werden. Zu jeder Veranstaltung müssen alle festgelegten harten Constraints betrachtet werden und geprüft werden, ob das jeweils untersuchte Constraint erfüllt ist oder nicht. Pro erfülltem Hard Constraint wird der Fitnesswert des Stundenplans ***inkrementiert***. Nach erfolgter Auswertung liegt für ein „Stundenplan-Chromosom“ ein Fitnesswert **vor** Betrachtung der Soft Constraints vor. Hierfür lässt sich folgendes festhalten:

$$Fitness = Anzahl\ Hard\ Constraints\ \times Anzahl\ Veranstaltungen$$

entspricht einer validen Lösung eines Stundenplans, die so unter Betrachtung aller harten Constraints in Realität umgesetzt werden kann.

Anschließend an die Auswertung der harten Constraints werden nun die Soft Constraints betrachtet. Auch hier werden die Veranstaltungen iteriert und geprüft, ob die definierten Soft Constraints eingehalten werden oder nicht. Folgende Überlegungen müssen an dieser Stelle noch vorgenommen werden:

- Wie gewichtet man die einzelnen Soft Constraints (SC)?
  Bsp.(!!): SC1 am wichtigsten, SC6 am unwichtigsten → Inkrementieren bei erfüllter Bedingung wird folgend mit z.B. Gewichtung 1 * 0,8, bzw. 1 * 0,1 durchgeführt.

- Wie wird die allgemeine relative Gewichtung von Soft Constraints zu Hard Constraints aussehen? Ein mögliches Problem könnte nämlich sein, dass Stundenpläne, die viele Soft Constraints erfüllen, aber wenig Hard Constraints, einen zu hohen Fitnesswert bekommen und sich daraus ableitend die Folgegenerationen möglicherweise falsch gerichtet entwickeln.

2.4 Datenstruktur(en)

Als oberste Struktur wird ein Populations-Objekt benötigt. Dieses stellt eine Generation da und enthält dabei alle Individuen/Chromosomen dieser Generation. Wenn es zur Fortpflanzung und zu Mutationen kommt, wird auf Grundlage des Populations-Objektes über entsprechende Verfahren (vgl. Kapitel 1) bestimmte Chromosomen gewählt. Die notwendigen Operationen werden aber nicht auf der Population, sondern auf den Chromosomen ausgeführt. Im Vorhinein müssen für die Auswahl bei Fortpflanzung und Mutation Parameter festgelegt werden. Beispielsweise wird über alle Individuen des Populationsobjektes traversiert und auf jedem

Chromosom geschieht eine Mutation mit einer Wahrscheinlichkeit p=0,05. Zudem muss festgelegt werden, wie viele Nachfahren pro Generation entstehen sollen, und wie groß eine Population generell sein soll. Dieser Wert sollte von Generation zu Generation konstant gehalten werden. Darüber hinaus müssen die Chromosomen bestimmt werden, die aus der aktuellen Population in die Folgegeneration übernommen werden. *Ein Vorgehen kann hierbei sein, die Top 3 Individuen mit dem höchsten Fitnesswert zu übernehmen, und den Rest über z.B. die Glücksradmethode (vgl. 1.2.3.1) zu bestimmen*.

Geht man in der Datenstruktur eine Ebene tiefer, gelangt man zu den individuellen Chromosomen-Objekten. Diese wiederum bestehen aus einer festen Anzahl an Gen-Objekten, welche der Anzahl an Veranstaltungen entspricht. Auf Basis dieser Objekte werden Rekombination und Mutationen ausgeführt. Eine Mutation kann dabei wie folgt aussehen: Es werden zufällig zwischen 0 und 10 Gene ausgewählt, denen ein neuer Zeitslot und Raum zugewiesen wird. Heuristische Verbesserungen bzgl. der Laufzeit können hier beispielsweise vorgenommen werden, indem mit einer Wahrscheinlichkeit von p=0.5 nur der Zeitslot und nicht der Raum geändert wird, wenn der gespeicherte Raum im Gen bereits zur Veranstaltung passt (bzgl. der Größe). Die Rekombination kann mit einer der in Kapitel 1 beschriebenen Möglichkeiten vorgenommen werden. Alternativ gibt es noch die Möglichkeiten „double point crossover“ oder „circular crossover“.

Auf einem Chromosom-Objekt wird außerdem der für dieses Objekt geltende Fitnesswert berechnet. Hier ist Vorsicht bzgl. der Laufzeit geboten, da die Berechnungen der Fitnesswerte den Großteil der Berechnungszeit des Algorithmus ausmachen. Jedes Gen-Objekt muss nämlich beispielsweise gegen jedes andere Gen des Chromosoms geprüft werden, um beispielsweise das Constraint zu prüfen, ob ein Raum in einem Zeitfenster bereits von einem Gen des Stundenplans belegt ist. Dies bringt also eine quadratische Laufzeit mit sich (O(n^2), mit n = Anzahl der Veranstaltungen/Gene). Die Nutzung von Threads und Threadpools sollte in diesem Zusammenhang in jedem Fall genauer untersucht werden.

Den untersten Baustein der Datenstruktur bildet das Gen. Ein Gen besteht dabei jeweils aus einer Veranstaltung, einem Dozenten, einem Zeitslot und einem Raum (+ einer Menge an Studiengängen, die diese Veranstaltung im Curriculum enthalten). Mithilfe dieser Information ist es möglich die Fitnesswerte für alle Chromosomen einer Population zu bestimmen. Da die Gene als Objekte sehr umfangreich sind, sollte eventuell geprüft werden, ob Gene nur einmal instanziiert und nachfolgend referenziert werden.

2.5 Ergebnis

Zum Abschluss muss bestimmt werden, ab welchem Zeitpunkt der Algorithmus ausreichend erfolgreich durchgelaufen ist. Hierfür steht noch aus, aus den bestehenden Soft Constraints, ihrer Gewichtungen und der Fitnessfunktion ein Fitnesswert zu bestimmen, an dem ein Stundenplan/Chromosom als ausreichend optimal angesehen wird und der Algorithmus terminiert wird. Da es bei genetischen Algorithmen aber wie in der Natur oftmals dazu kommen kann, dass Folgegenerationen nach und nach homogener in ihrer Struktur werden und es dadurch zu Stagnation kommen kann, sollte zudem eine maximale Anzahl an Generationen festgelegt werden, die entstehen sollen. Wenn diese Anzahl erreicht ist, sollte der Algorithmus ebenfalls terminiert werden.

2.6 Benchmark-Tests

Es sollten verschiedene Benchmark-Tests zu den Parametern ausgeführt werden, die vor Ausführung des Algorithmus festgelegt werden müssen, da die Effizienz und das Ergebnis stark von der Auswahl dieser abhängig sein können. Dazu zählen u.a. Populationsgröße, Mutationsrate oder auch Anzahl der Nachwuchschromosomen.

Beispiele aus anderen Implementierungen, die zeigen, wie stark die Ergebnisse abhängig von den Parametern variieren können:

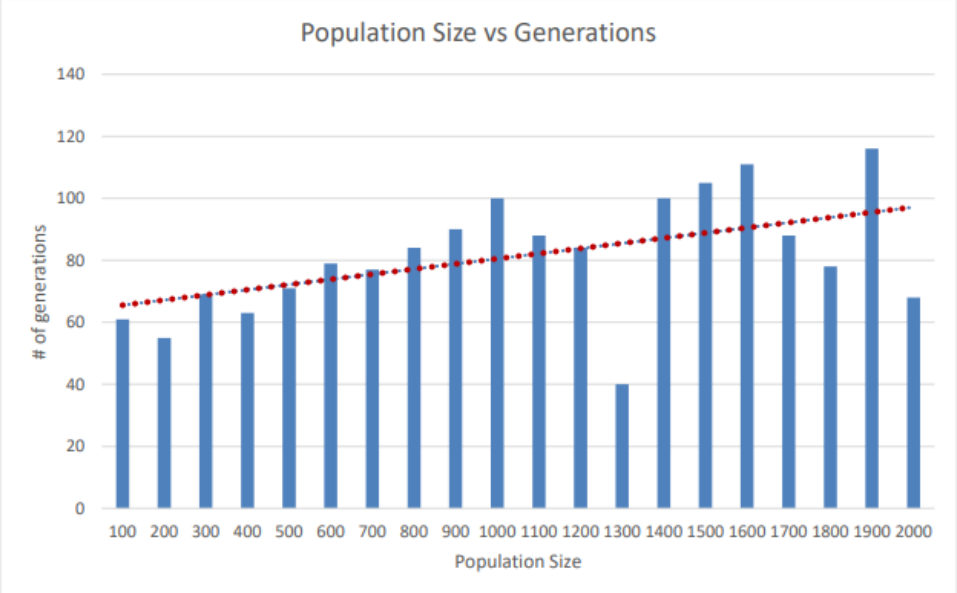

Figure XII: Population size increase vs number of generations to reach a solution.

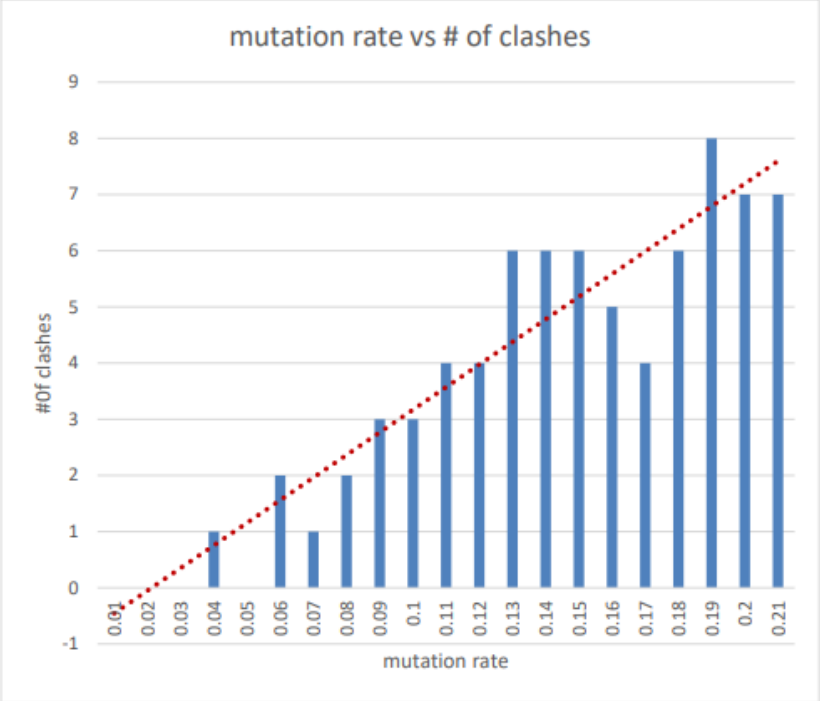

Figure XIII: mutation rate vs number of clashes.

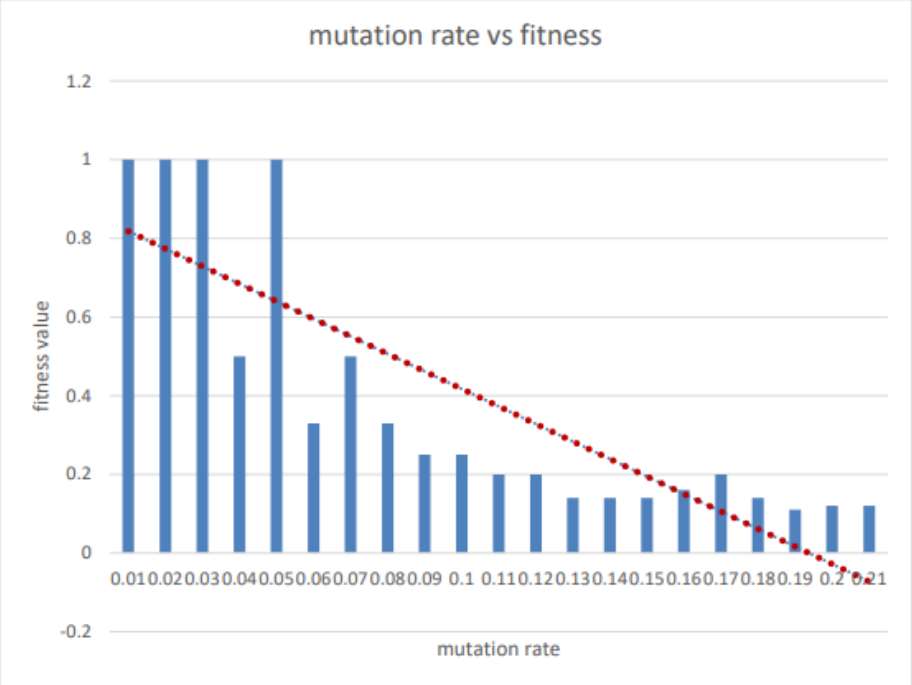

Figure XIV: mutation rate vs fitness value.

2.7 Weitere Überlegungen (zu erweitern!!)

- Fitnesswert von Chromosomen, die von einer Generation in die nächste Generation übernommen werden, könnte gecashed werden, um redundante Berechnungen der Fitnesswerte zu vermeiden

- Eventuell mit temporären denormalisierten DB-Tabellen arbeiten, um Effizienz beim Datenzugriff zu steigern
-

[1] https://egrove.olemiss.edu/cgi/viewcontent.cgi?article=1442&context=etd

[2] https://andreweast.net/wp-content/uploads/2019/06/Timetable-Scheduling-via-Genetic-Algorithm-Andrew-Reid-East.pdf

[3] https://eprints.ost.ch/id/eprint/95/1/Stundenplan-Algorithmen.pdf

[4] https://ieeexplore.ieee.org/document/7159319