FH Bielefeld
University of
Applied Sciences

AMMO
Angewandte Mathematische Modellierung & Optimierung

AMMO – Berichte aus Forschung und Technologietransfer

,

Genetische Algorithmen zur Ermittlung von Hamiltonzyklen in Digraphen

Karim Kai Abdelhak, Bernhard Bachmann, Hermann-Josef Kruse

Heft Nr. 10

Juli 2018

Veröffentlichungsreihe (Onlinepublikation):
AMMO – Berichte aus Forschung und Technologietransfer

ISSN
2198-4824

Erscheinungsort
`https://www.fh-bielefeld.de/ium/forschung/forschungsschwerpunkte/ammo/veroeffentlichungen`

Herausgeber
Sprecher FSP AMMO, Fachhochschule Bielefeld

Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
FSP Angewandte Mathematische Modellierung und Optimierung
Interaktion 1
33619 Bielefeld

Die Autoren

Karim Kai Abdelhak (B. Sc.)
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
karim.abdelhak@fh-bielefeld.de
Fachgebiete: Algorithmik, Graphentheorie, Bionische Methoden der Optimierung, Signalverarbeitung

Absolvierte den Bachelor-Studiengang *Angewandte Mathematik* an der Fachhochschule Bielefeld und studiert dort momentan *Optimierung und Simulation* im Master. Durch die Bachelorarbeit und ein Vertiefungsprojekt spezialisierte er sich zunächst auf die mehrdimensionale Signalverarbeitung, erweiterte seine Kerngebiete jedoch durch private Projekte um Graphentheorie und Optimierungsalgorithmen speziell im Bereich der bionischen Optimierungsverfahren.

Prof. Dr. phil. Bernhard Bachmann
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
bernhard.bachmann@fh-bielefeld.de
Fachgebiete: Numerische Mathematik, Optimierung, Symbolische und numerische Behandlung großer hybrider differential-algebraischer Gleichungssysteme

Seit 1999 als Professor für Mathematik und ihre technischen Anwendungen an der Fachhochschule Bielefeld tätig und lehrt dort im Bachelor-Studiengang *Angewandte Mathematik* und im Master-Studiengang *Optimierung & Simulation* des Fachbereichs *Ingenieurwissenschaften & Mathematik*. Gründungsmitglied des Forschungsschwerpunktes *Angewandte Mathematische Modellierung & Optimierung* (FSP AMMO) der FH Bielefeld. Gründungsmitglied der *Modelica Association*. Gründungs- und Vorstandsmitglied des *Open Source Modelica Consortium* (OSMC). Nationale und internationale F&E-Projekte im Bereich der Modellierung, Simulation und Optimierung hybrider dynamischer Systeme.

Prof. Dr. rer. pol. Hermann-Josef Kruse
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
hermann-josef.kruse@fh-bielefeld.de
Fachgebiete: Wirtschaftsmathematik, Operations Research

Seit 1995 als Professor für Wirtschaftsmathematik (insbesondere Operations Research) an der Fachhochschule Bielefeld tätig und lehrt dort im Bachelor-Studiengang *Angewandte Mathematik* und im Master-Studiengang *Optimierung & Simulation* des Fachbereichs *Ingenieurwissenschaften & Mathematik*. Gründungsmitglied des Forschungsschwerpunktes *Angewandte Mathematische Modellierung & Optimierung* (FSP AMMO) der FH Bielefeld. F&E-Projekte im Bereich der Optimierung und Simulation diskreter Systeme zur Entscheidungsunterstützung bei betrieblichen Problemstellungen.

Inhaltsverzeichnis

1 Einleitung

In dieser Ausarbeitung geht es darum, eine stochastische Heuristik zu entwickeln, die Lösungen für das sogenannte ***gerichtete Hamiltonkreisproblem*** oder auch ***Hamiltonzyklusproblem*** ermitteln kann. Dieses *NP-vollständige* Problem wurde nach dem irischen Astronomen und Mathematiker Sir William Rowan Hamilton benannt, der 1857 das Spiel *„The Icosian Game“* erfand. In diesem Spiel geht es darum, alle Ecken eines Dodekaeders durch Entlanggehen der anliegenden Kanten so abzulaufen, dass keine Ecke doppelt genutzt wird und zum Schluss zur Ausgangsecke zurückgekehrt wird. Wie beispielsweise in [1] und [5] beschrieben, gibt es einige analytische Ansätze, mit denen bewiesen werden kann, dass ein Digraph einen ***Hamiltonzyklus*** beinhaltet, jedoch gilt dies nur für bestimmte Digraphen. Nach Diracs Theorem ist jeder Digraph mit $n \geq 3$ Knoten und kleinstem Knotengrad von $\frac{n}{2}$ oder mehr ***hamiltonsch***, dies lässt sich jedoch nicht auf andere Digraphen übertragen. Da dieses Problem für eine allgemeine Digraphtopologie analytisch unlösbar scheint, lohnt es sich, effiziente Heuristiken zu entwickeln, die in der Lage sind, Hamiltonzyklen zu ermitteln.

Anders als typische Lösungsalgorithmen für Hamiltonzyklusprobleme, arbeitet der hier vorgestellte Algorithmus direkt auf einem Digraphen. Üblicherweise wird der zu untersuchende Digraph zuvor in eine nahezu äquivalente Darstellung als ungerichteter Graph[1] umgewandelt, hierbei verdreifacht sich jedoch die Anzahl der Knoten, wodurch der Lösungsaufwand enorm ansteigt. Dieser Vorgang ist schematisch in Abbildung 1 dargestellt.

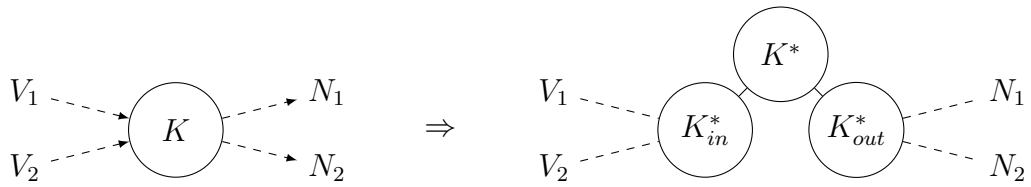

Abbildung 1: *Transformation eines Digraphen zu einem Graphen exemplarisch dargestellt an der Transformation eines Knotens.*

Ein Knoten K wird durch drei neue Knoten K^*, K^*_{in} und K^*_{out} ersetzt, hierbei werden alle ursprünglich eingehenden gerichteten Kanten mit K^*_{in} und alle ausgehenden mit K^*_{out} verbunden. K^* dient als Brückenknoten und stellt sicher, dass in einem ermittelten Hamil-

[1] Die Darstellung ist nur nahezu äquivalent, da bei der Rücktransformation der Lösung die Richtung des Zyklus ohne Kenntnis des ursprünglichen Digraphen mehrdeutig ist.

tonkreis des entstehenden Graphen dieses Dreiertupel von Knoten hintereinander durchlaufen werden muss.

1.1 Hamiltonzyklen

Die Grundlage der folgenden Untersuchungen bilden ***gerichtete Graphen*** oder ***Digraphen***. Gegeben sei eine nichtleere, endliche Menge V und eine Menge $E \subset V \times V$, wobei zudem die „Diagonalpaare“ (i, i) ausgeschlossen werden; o.B.d.A. sei $V = \{1, 2, \ldots, n\}$. Das Mengenpaar $G = (V, E)$ heißt ***gerichteter Graph*** oder ***Digraph***. Die Elemente von V werden ***Knoten***, die von E ***gerichtete Kanten*** oder ***Pfeile*** genannt. Die Mächtigkeit der Knotenmenge V bzw. der Pfeilmenge E wird mit $|V| = n$ bzw. $|E| = m$ bezeichnet.

Jeder endliche Digraph lässt sich durch eine binäre ***Adjazenzmatrix*** $A = (a_{ij})_{i,j=1,\ldots,n}$ eindeutig beschreiben, wobei gilt[2]:

$$a_{ij} = \begin{cases} 1, & \text{falls } (i, j) \in E, \\ 0, & \text{sonst.} \end{cases}$$

Jeder Knoten $i \in V$ besitzt eine Menge von Nachfolgern und Vorgängern[3]:

$$S_i = \{k \in V | (i, k) \in E\} \text{ ist die } \textbf{\textit{Nachfolgermenge}} \text{ von Knoten } i,$$
$$P_i = \{k \in V | (k, i) \in E\} \text{ ist die } \textbf{\textit{Vorgängermenge}} \text{ von Knoten } i.$$

Zur Nachfolgermenge S_i bzw. Vorgängermenge P_i lässt sich auf kanonische Weise die ***Ausgangskantenmenge*** $\Psi_i = \{(i, k) | k \in S_i\}$ bzw. die ***Eingangskantenmenge*** $\Phi_i = \{(k, i) | k \in P_i\}$ bilden. Die Mächtigkeit von S_i (bzw. Ψ_i) heißt die ***Ausgangsvalenz*** (oder der *positive Grad*), die Mächtigkeit von P_i (bzw. Φ_i) heißt die ***Eingangsvalenz*** (oder der *negative Grad*) des Knotens $i \in V$.

Eine Folge von gerichteten Kanten (Pfeilen) wird durch $(i_1, i_2, \ldots, i_k, i_{k+1}, \ldots, i_s)$ beschrieben, wenn gilt: $(i_k, i_{k+1}) \in E$ für alle $k = 1, \ldots, s - 1$. Falls in dieser Pfeilfolge jeder Knoten i_k höchstens einmal vorkommt, spricht man von einem ***Pfad*** in G, der i_1 mit i_s verbindet. Falls diese Eindeutigkeit allein dadurch aufgehoben wird, dass $i_1 = i_s$ gilt, spricht man von einem *geschlossenen Pfad* oder ***Zyklus*** in G. Die Anzahl der gerichteten Kanten (Pfeile) in einem Pfad bzw. Zyklus wird die (Pfad- bzw. Zyklus-)***Länge*** genannt. Ein Zyklus G mit der Länge $l = |V|$ heißt ein ***Hamiltonzyklus*** in G.[4] Falls ein Digraph

[2] Ist die Adjazenzmatrix A symmetrisch, d.h. es gilt $a_{ij} = a_{ji}$ für alle $i, j \in V$, spricht man auch von einem *ungerichteten Graphen* oder *Graph* schlechthin.

[3] Engl. *successor* bzw. *predecessor*.

[4] In einem ungerichteten Graphen spricht man von einem *Hamiltonkreis*.

G einen Hamiltonzyklus enthält, nennt man G ***hamiltonsch*** oder einen ***Hamiltondigraph***.

Beispiel 1.1:
Es sei ein Digraph mit $V = \{1, 2, 3, 4, 5, 6, 7\}$. Die Kanten aus E sind in Abbildung 2 dargestellt. Die zugehörige Adjazenzmatrix wird in Tabelle 1 dargestellt. Wie leicht zu erkennen ist, existiert ein Hamiltonzyklus: $1 - 6 - 3 - 2 - 4 - 7 - 5 - 1$.

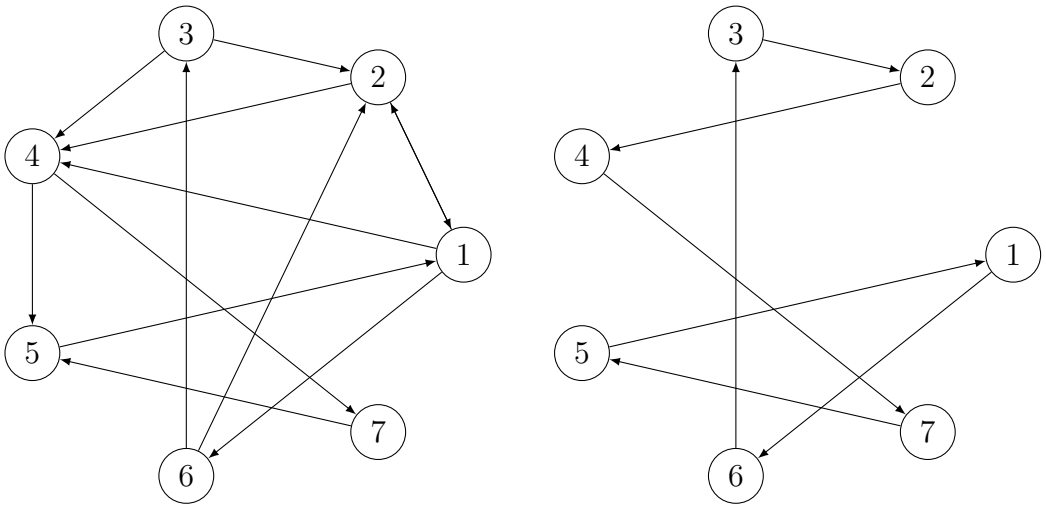

Abbildung 2: *Digraph mit 7 Knoten (links) und existierendem Hamiltonzyklus (rechts).*

	1	2	3	4	5	6	7	Σ
1	0	1	0	1	0	1	0	3
2	1	0	0	1	0	0	0	2
3	0	1	0	1	0	0	0	2
4	0	0	0	0	1	0	1	2
5	1	0	0	0	0	0	0	1
6	0	1	1	0	0	0	0	2
7	0	0	0	0	1	0	0	1
Σ	2	3	1	3	2	1	1	13

Tabelle 1: *Die Adjazenzmatrix des Digraphen aus Abbildung 2.*

1.2 Genetische Algorithmen

Genetische Algorithmen sind aufgrund der Erkenntnisse der **biologischen Evolutionstheorie** von Charles R. Darwin, erstmals 1859 veröffentlicht in *On the Origin of Species by Means of Natural Selection*, entstanden. Es wurde versucht das grundsätzliche Prinzip der natürlichen Auslese von vorteilhaften Eigenschaften, die durch zufällige Variation entstehen, mathematisch nachzubilden. Nach dem Vorbild der sogenannten ***Chromosomen*** werden Lösungsanwärter für ein Problem kodiert, sodass jedes ***Gen*** eines *Chromosomes* eine Eigenschaft repräsentiert. Jede mögliche Ausprägung eines *Gens* wird ***Allel*** genannt. Die ***Fitness*** eines Chromosomes sagt aus, wie wahrscheinlich es ist, dass das Lebewesen überleben und sich fortpflanzen kann. Diese *Fitness* ist abhängig von den Eigenschaften und somit den *Allelen* der einzelnen *Gene* des *Chromosomes*. Als Minimalbeispiel kann man sich ein Tier vorstellen, welches in der Savanne überleben möchte. Ein *Gen* des repräsentativen *Chromosomes* sei hier die Haarfarbe des Tieres und die betrachteten *Allele* seien *schwarz* und *hellbraun*. Da mögliche Raubtiere das betrachtete Tier mit schwarzen Haaren in der Savanne eher entdecken würden, ist ein Tier mit dem *Allel hellbraun* im Vorteil und hat eine höhere *Fitness*. Dieses Tier hat nun höhere Überlebenschancen und somit auch mehr Gelegenheit sich zu vermehren. Durch dieses Prinzip der *natürlichen Auslese* setzen sich nützlichere Eigenschaften gegenüber den weniger vorteilhaften auf ganz natürliche Art und Weise durch.

Zusammenfassend spielen die folgenden drei Eigenschaften eine große Rolle:

Diversität Alle Lebewesen sind voneinander verschieden, dennoch bilden alle Lebewesen zusammen (*Population*) nur einen Bruchteil des prinzipiell Möglichen.

Vererbung Gute Eigenschaften werden an die nachfolgende Generation mit höherer Wahrscheinlichkeit vererbt als schlechtere.

Variation Durch Mutation und Rekombination entstehen laufend neue Variationen der vorhandenen Lebewesen.

Zusätzliche Informationen zur Vererbungslehre können mit Empfehlung des fünften Kapitels *Fitness – ein Begriff und seine Deutung* dem Buch *Gene, Zufall, Selektion* von Veiko Krauß entnommen werden [4]. Für nähere Erläuterungen zu Genetischen Algorithmen wird das Buch *Genetische Algorithmen und Optimierung* empfohlen [3].

2 Entwicklung des Algorithmus

Grundlage des Algorithmus ist ein sogenanntes *Chromosom*. Es repräsentiert eine mögliche Lösung des betrachteten Problems, wobei die *Gene* dieses Chromosomes gewisse Eigenschaften darstellen. Zu Beginn wird eine *Population* solcher Chromosome zufällig erzeugt, der Algorithmus mutiert, rekombiniert und selektiert nun diese Chromosome, bis eine Lösung gefunden oder ein Iterationslimit erreicht wird.

2.1 Kodierung des Problems

Als Grundlage der Kodierung wird jedem Knoten ein *Gen* zugeordnet. Da in einem Hamiltonzyklus jeder Knoten genau eine Eingangskante und eine Ausgangskante hat, kann hier zwischen zwei Möglichkeiten gewählt werden.

Primal: Die Ausprägung j des Genes g_i besagt, dass Knoten i auf Knoten j zeigt.

Dual: Die Ausprägung j des Genes g_i besagt, dass Knoten j auf Knoten i zeigt.

Somit werden je nach gewählter Kodierung entweder die Ausgänge oder die Eingänge betrachtet. Hierbei dürfen die Gene nicht jeden Wert annehmen, denn die sogenannten *Allele* eines primal kodierten Genes g_i sind beschränkt auf alle möglichen $k \in S_i$, d.h. $(i, k) \in \Psi_i$, bzw. für ein dual kodiertes Gen g_i beschränkt auf alle möglichen $k \in P_i$, d.h. $(k, i) \in \Phi_i$. Das erste Gen aus Beispiel 1.1 kann also primal die Werte 2, 4, 6 und dual die Werte 2 und 5 annehmen.

Ein *Chromosom* besteht aus n Genen, wobei n die Anzahl an Knoten des betrachteten Digraphen ist. Jedes Gen repräsentiert hierbei einen anderen Knoten. In Abbildung 3 ist das primale Chromosom

c :

2	4	2	5	1	2	5

für Beispiel 1.1 dargestellt. Jedem Knoten ist genau eine Ausgangskante zugeordnet. In einem dualen Chromosom wäre jedem Knoten genau eine Eingangskante zugeordnet. Die Anzahl λ an möglichen Chromosomen lässt sich ermitteln, indem die Anzahl der

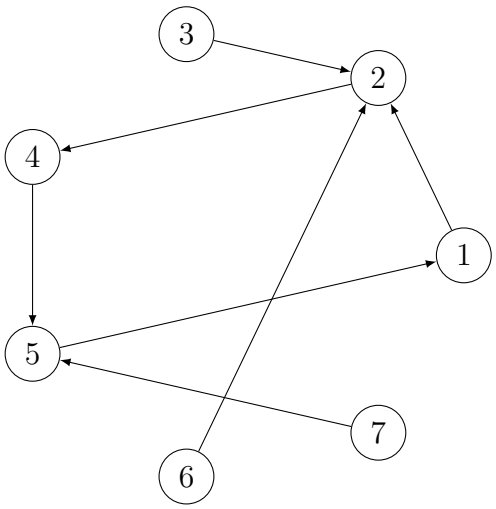

Abbildung 3: *Ein primales Chromosom erzeugt aus dem Digraphen aus Abbildung 2.*

Allele eines jeden Genes miteinander multipliziert werden. Für primal bzw. dual kodierte Chromosomen wäre dies unterschiedlich:

$$\text{primal:} \quad \lambda_p = \prod_{i=1}^{n} |\Psi_i| \tag{1}$$

$$\text{dual:} \quad \lambda_d = \prod_{i=1}^{n} |\Phi_i| \tag{2}$$

Laut Tabelle 1 ist die Anzahl an primalen Chromosomen $\lambda_p = 3 \cdot 2 \cdot 2 \cdot 2 \cdot 1 \cdot 2 \cdot 1 = 48$ und die Anzahl an dualen Chromosomen $\lambda_d = 2 \cdot 3 \cdot 1 \cdot 3 \cdot 2 \cdot 1 \cdot 1 = 36$. Es fällt auf, dass hier trotz gleicher Kantenanzahl die duale Kodierung einen kleineren Suchraum induziert. Ein kleinerer Suchraum führt im Allgemeinen zu einer schnelleren Ermittlung eines Ergebnisses und dies könnte zu einer Vorauswahl der Kodierungsart genutzt werden. Wie jedoch später beschrieben wird, ist es meist sinnvoller, bei Stagnation zwischen den beiden Kodierungsarten zu wechseln (vgl. Abschn. 2.4.1).

2.2 Startpopulation und Fitness

Zur Ermittlung einer Startpopulation muss zunächst die Populationsgröße P festgelegt werden. Es bietet sich an, diese abhängig von der Anzahl der Kanten und einem Steuerparameter ϕ zu machen:

$$P = \phi \cdot m \tag{3}$$

Allerdings wird im Folgenden die Populationsgröße als *Quadratzahl* gewählt, um die Populationen besser in einem Raster darstellen zu können.

Die Chromosomen der ersten Population werden durch eine zufällige Auswahl der Ausprägungen der Gene erzeugt. Hierbei ist wieder zu beachten, dass die Ausprägung

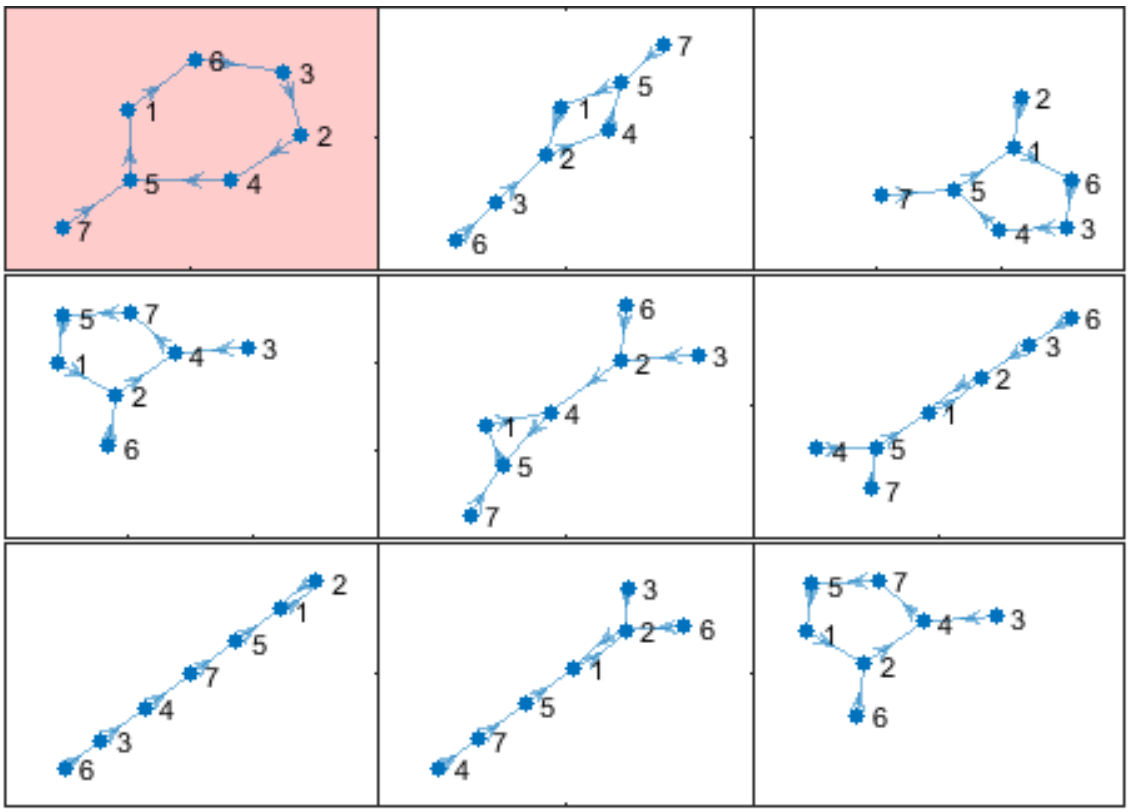

Abbildung 4: *Startpopulation mit $P = 9$ Chromosomen.*

eines Gens auf seine Allele beschränkt ist. Eine zufällig erzeugte Startpopulation für das Problem aus Beispiel 1.1 mit $P = 9$ Chromosomen ist in Abbildung 4 dargestellt.

Die Chromosomen werden nun bewertet, um festzustellen welche Chromosome überleben werden. Das beste Chromosom einer Population ist in den jeweiligen Abbildungen **rot** hinterlegt. Diese Bewertung wird auch *Fitness* genannt und muss durch einen numerischen, vergleichbaren Wert dargestellt werden, der sich umgekehrt proportional zum „Abstand“ des Chromosomes zu einem Hamiltonzyklus verhält. Diese Definition ist sehr vage und mathematisch noch nicht verwertbar, es können jedoch folgende Eigenschaften eines Chromosomes ermittelt werden, um daraus die Fitness F zu bestimmen:

Zyklenanzahl ζ: Die Anzahl an Zyklen im Chromosom ($\zeta > 0$).

Zyklenknoten Z: Die Anzahl aller Knoten, die sich in Zyklen befinden.

Dominanz D: Die Knotenanzahl des längsten Pfades.

In einem Hamiltonzyklus gehören alle Knoten zu einem Zyklus und der längste Pfad umfasst ebenfalls alle Knoten, daher sollte die Fitness proportional zu diesen Eigenschaften

steigen. Die Anzahl an Zyklen sollte minimiert werden, da ein Hamiltonzyklus nur einen einzigen Zyklus beinhaltet.[5] Diese Zusammenhänge können durch die Gleichung

$$F^* = \frac{D \cdot Z}{\sqrt{\zeta}} \tag{4}$$

dargestellt werden.[6] Maximal nimmt dieser Fitnesswert F^* den Wert n^2 an. Es ist praktikabel, diesen Fitnesswert zu normieren und einen weiteren Steuerparameter α einzuführen:

$$F = \left(\frac{F^*}{n^2}\right)^{\alpha} = \left(\frac{D \cdot Z}{\sqrt{\zeta} \cdot n^2}\right)^{\alpha}. \tag{5}$$

Durch die Normierung gilt $0 \leq F \leq 1$ und die Fitnesswerte können unabhängig von der Knotenanzahl n interpretiert werden. Mit Hilfe des Steuerparameters α kann die *Selektivität* des Verfahrens verändert werden. Je höher die *Selektivität* eines genetischen Algorithmus ist, umso stärker werden gute *Chromosome* den schlechten gegenüber bevorzugt. Da der potenzierte Wert durch die Normierung immer zwischen 0 und 1 liegt, führt $\alpha > 1$ zu einer Stauchung der Fitnessfunktion, sodass der Unterschied zwischen den Bewertungen kleiner ausfällt. Im Umkehrschluss führt $\alpha < 1$ zu einer Streckung der Abstände zwischen diesen Bewertungen und somit zu einer höheren *Selektivität*. Des Weiteren muss $\alpha > 0$ gelten, da sich die Fitnessfunktion F sonst antiproportional zu der ursprünglichen Fitness F^* verhalten würde. Empirische Untersuchungen von Problemen mit einer Knotenmenge von bis zu 150 Knoten haben ergeben, dass je nach Zielsetzung und Topologie (vgl. Abschnitt 3.2) ein $\alpha \in [0.9, 1.2]$ geeignet ist.

Das Optimum der Fitness F liegt bei 1. Wird dieses Maximum als Fitnesswert für ein Chromosom ermittelt, ist sichergestellt, dass es sich um einen Hamiltonzyklus handelt, da alle Knoten zu einem Zyklus gehören müssen und es nur einen Zyklus geben darf, damit dieser Wert erreicht wird.

2.2.1 Markierungsalgorithmus zur Ermittlung der Fitness

Um ein Chromosom so zu markieren, dass die Anzahl an Zyklen (ζ), die Anzahl an Knoten in allen Zyklen (Z) und die Knotenanzahl des längsten Pfades (D) ermittelt werden kann, werden jedem Gen vier Eigenschaften zugesprochen:

Nachfolger: Der nachfolgend betrachtete Knoten,

Vorgänger: Der zuvor betrachtete Knoten,

[5] Durch die Kodierung existiert immer mindestens ein Zyklus. Dies wird im folgenden Markierungsalgorithmus zur Ermittlung dieser Eigenschaften noch genauer erläutert.

[6] Die „Dämpfung“ des Produktes $D \cdot Z$ durch den Faktor $1/\sqrt{\zeta}$ könnte allgemein durch $1/\zeta^{\beta}$ mit $0 \leq \beta \leq 1$ ersetzt werden. Im Folgenden wird der Steuerungsparameter β für die Dämpfungsstärke durchgängig auf $\beta = 1/2$ gesetzt. In der Parameteroptimierung aus Kapitel 3 wird dieser Parameter nicht betrachtet, für genauere Untersuchungen könnte er jedoch hinzugezogen werden.

Länge: Knotenanzahl des längsten Pfades durch das Chromosom von diesem Knoten aus

Status: Bearbeitungsstatus

- **0:** Unbearbeitet,
- **1:** Initialisiert,
- **2:** Als Kreisknoten erkannt,
- **3:** Als Pfadknoten erkannt.

Hier ist jedoch darauf zu achten, dass sich für dual kodierte Gene die Bedeutung von *Nachfolger* und *Vorgänger* in Bezug auf die Kantenrichtung umdreht. Knoten i als Nachfolger von Knoten j bedeutet nur in der primalen Kodierung, dass die Kante auch von j nach i zeigt. Dual dreht sich diese Betrachtungsweise um. Bei geschickter Implementierung des nachfolgend vorgestellten Algorithmus macht die Art der Kodierung bei der Ermittlung der Fitness jedoch keinen Unterschied.

1: **procedure** F = RATE(CHR)
2: **Input:**
3: *CHR* //Chromosom mit N Genen
4: *Parameter:* α //Steuerparameter Streckung/Stauchung

5: **Initialisierung:**
6: **Rest** //Liste der Gene von CHR
7: **Nachfolger** //Array der Genausprägungen von CHR
8: **Vorgänger** //Nullwertiges Array der Länge N
9: **Status** //Nullwertiges Array der Länge N
10: **Länge** //Nullwertiges Array der Länge N
11: $\zeta = 0$ //Anzahl detektierter Zyklen
12: $Z = 0$ //Anzahl detektierter Knoten in Zyklen
13: $D = 0$ //Knotenanzahl des längsten detektierten Pfades

14: **while Rest** nicht leer **do**
15: Wähle Startknoten K als erstes Element aus **Rest**
16: $L = 1$ //Aktuelle Länge
17: $V = 0$ //Aktueller Vorgänger

18: **while Status**$(K) == 0$ **do**
19: **Status**(K) = 1 //Initialisierung
20: **Länge**(K) = L
21: **Vorgänger**(K) = V

22: $V = K$
23: Entferne K aus **Rest**
24: $K =$ **Nachfolger**(K)
25: $L = L + 1$
26: **end while**

27: **if Status**$(K) == 1$ **then** //Kreis detektiert
28: $O = L -$ **Länge**(K) //Bestimmung der Kreislänge
29: $Z = Z + O$
30: $\zeta = \zeta + 1$
31: **while Status**$(K) == 1$ **do** //Zweitdurchlauf des Kreises
32: **Status**$(K) = 2$
33: **Länge**$(K) = O$
34: $K =$ **Nachfolger**(K)
35: **end while**
36: **else**//Einzelpfad erkannt
37: $O =$ **Länge**(K) //Länge des folgenden Knotens
38: **end if**

39: **while Vorgänger**$(K) \neq 0$ **do** //Zurückverfolgung des Pfades
40: $K =$ **Vorgänger**(K)
41: **Status**$(K) = 3$
42: $O = O + 1$
43: **Länge**$(K) = O$
44: **end while**
45: $D = \max(D,$ **Länge**$(K))$ //Ermittlung der Knotenanzahl des längsten Pfades
46: **end while**

47: $F = [(D \cdot Z)/(\sqrt{\zeta} \cdot N^2)]^{\alpha}$ //Berechnung des Fitnesswertes

48: **Output:** F
49: **end procedure**

In der zweiten **while**-Schleife, beginnend in Codezeile 18, wird im ersten Durchlauf das erste Gen betrachtet, als Startknoten verwendet und initialisiert (Status $= 1$). Die Ausprägung dieses Gens wird als nächster betrachteter Knoten genutzt und dessen Status wird ebenfalls initialisiert. Dies passiert so lange, bis der Algorithmus auf einen Knoten trifft, der nicht den Status 0 hat. Da zu Beginn alle Status auf 0 gesetzt sind und diese Schleife den Status einzelner Gene nur auf 1 setzen kann, stoppt diese Schleife zwangsläufig bei einem Gen mit Status 1. Dies bedeutet, dass ein Zyklus gefunden wurde. Logisch

kann man sich dies klarmachen, indem man in einem primal kodierten Chromosom einen beliebigen Startpunkt sucht und von dort aus die Kanten in vorgeschriebener Richtung verfolgt, bis man auf einen bereits betrachteten Knoten trifft. Jedem Knoten bzw. Gen ist genau eine solche Kante zugeordnet, daher ist dieses Vorgehen eindeutig. Wird ein Digraph mit endlicher Knotenzahl betrachtet, trifft man zwangsläufig auf einen Knoten, den man in genau diesem Prozess bereits betrachtet hat und schließt einen Zyklus. Für duale Chromosome müsste man für dieses Verfahren die Kanten in umgekehrter Richtung verfolgen. Somit ist sichergestellt, dass immer mindestens ein Zyklus existiert und $\zeta \neq 0$ gilt.

Trifft der Algorithmus auf ein Gen mit Status 0, setzt er diesen auf 1, trifft er auf ein Gen mit Status 1, wird dessen Status auf 2 gesetzt, falls ein Zyklus erneut durchschritten wird, oder auf 3 gesetzt, falls ein Pfad zurückverfolgt wird. Insgesamt werden die n Kanten eines Chromosomes genau zweimal durchlaufen. Die Rechenkomplexität hängt somit linear von der Anzahl der Knoten ab:

$$\mathcal{O}(n) \tag{6}$$

Im Anhang wird dieser Markierungsalgorithmus exemplarisch an einem Chromosom durchgeführt und die Einzelschritte werden genauer erklärt.

Dieser Algorithmus könnte durch die Ermittlung weiterer Eigenschaften des Chromosomes verbessert werden. Beispielsweise könnte die Anzahl von freien Enden, d.h. die Anzahl von Pfaden vor Zyklusbildung, ermittelt und in die Bewertung mit aufgenommen werden. Je nach Struktur der zu untersuchenden Digraphen könnte dies von Vorteil sein. Auch einige Veränderungen der Chromosomen aufgrund der gesammelten Informationen könnten hier stattfinden, einige Beispiele dazu sind im Ausblick in Kapitel 4 erläutert. Im Folgenden wird jedoch die in Gleichung (5) definierte Fitnessfunktion verwendet.

2.2.2 Selektion

Als Grundlage der Selektion dient die Bewertung durch den in Abschnitt 2.2.1 vorgestellten Markierungsalgorithmus. Als Referenzwert wird zunächst die Summe der Fitnesswerte ermittelt:

$$\bar{F} = \sum_{\ell=1}^{P} F_\ell, \tag{7}$$

wobei P die Populationsgröße und F_ℓ die Fitness des ℓ-ten Chromosomes darstellt. Nun wird eine Zwischenpopulation der Größe P erzeugt, indem jedes Chromosom dieser Population wahrscheinlichkeitsgewichtet zufällig durch eine sogenannte ***Rouletterad-Selektion*** ermittelt wird. Die Wahrscheinlichkeit p_ℓ jedes alten Chromosomes c_ℓ, über-

nommen zu werden, ist hierbei der Quotient aus dem eigenen Fitnesswert F_ℓ und dem Referenzwert $\bar{F}$:

$$p_\ell = \frac{F_\ell}{\bar{F}}. \tag{8}$$

Dass die Summe aller Wahrscheinlichkeiten 1 ergibt, ist gegeben durch

$$\sum_{\ell=1}^{P} p_\ell = \sum_{\ell=1}^{P} \frac{F_\ell}{\bar{F}} = \frac{\sum_{\ell=1}^{P} F_\ell}{\sum_{\ell=1}^{P} F_\ell} = 1. \tag{9}$$

Die nun entstandene Population beinhaltet mit hoher Wahrscheinlichkeit die besten Individuen der Vorpopulation, es kann jedoch auch sein, dass schlechtere Chromosome überleben und eine Chance bekommen, durch die im folgenden Abschnitt 2.3 erläuterten Kreuzungen und Mutationen einen besseren Fitnesswert und somit höhere Überlebenschancen zu erlangen. Um die Varianz und Flexibilität dieses Verfahrens zu erhöhen, wird das letzte Chromosom dieser Zwischenpopulation, wie zur Bildung der Startpopulation, zufällig und unabhängig von den anderen Chromsomen erzeugt. Dieses ***Alien*** hat sich in empirischen Untersuchungen bewährt, da es den ***Genpool***, d.h. die aktuell betrachteten *Allele*, ganz ähnlich der in Abschn. 2.3.2 genauer erläuterten Mutation, positiv beeinflusst. In Abbildung 5 ist die Startpopulation aus Abbildung 4 nach der Selektion dargestellt.

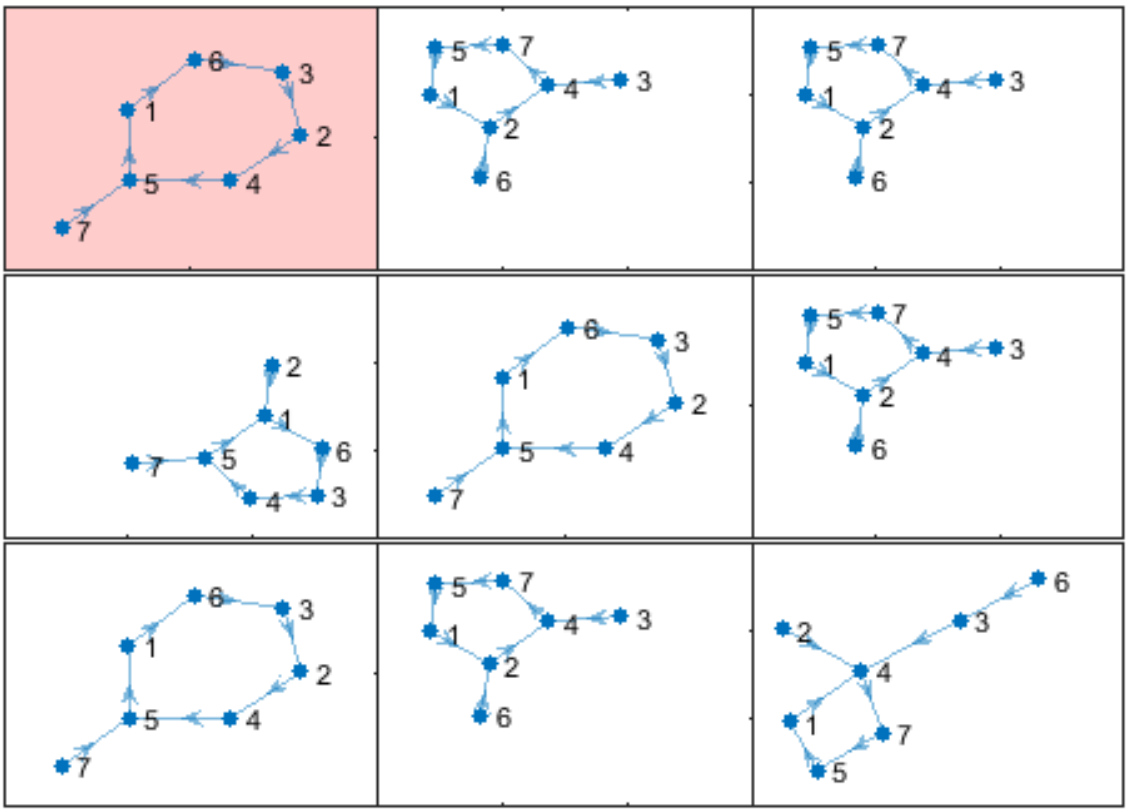

Abbildung 5: *Zwischenpopulation nach der Selektion.*

2.3 Erzeugung weiterer Generationen

Die Zwischenpopulation, bestehend aus vorausgewählten Chromosomen, muss nun leicht verändert werden, sodass sich neue Chromosomen entwickeln können. In Genetischen Algorithmen werden dazu ***Mutation*** und ***Kreuzung***, nach dem biologischen Vorbild der Evolutionstheorie, verwendet. Durch Kreuzung zweier Chromosomen können Ausprägungen ausgetauscht werden und durch zufällige Mutation werden bestimmten Genen zufällig neue Allele zugeteilt. Diese Methoden werden in den folgenden Kapiteln ausführlicher erläutert.

2.3.1 Kreuzungen

Zur Kreuzung werden zunächst zwei Chromosome aus der Zwischenpopulation, die noch nicht ausgewählt wurden, zufällig ausgewählt. Da die Anordnung der Chromosome durch zufallsbasierte Auswahl der Vorpopulation geschehen ist, muss hier keine weitere Zufallsauswahl vorgenommen werden, sodass die ersten beiden Chromosome der Zwischenpopulation ausgewählt werden können. In den darauf folgenden Schritten werden, der Reihenfolge nach, die darauf folgenden zwei Chromosome ausgewählt. Dies spart Rechenzeit, da keine Zufallszahlen zur Auswahl der Chromosomen generiert werden müssen. Handelt es sich um eine ungerade Populationsgröße, bleibt das letzte Chromosom unbeachtet.

Sind zwei Chromosomen ausgewählt, wird nun bestimmt, ob sie sich kreuzen. Dazu wird ein Kreuzungsparameter $p_K \in [0, 1]$ festgelegt, der die Wahrscheinlichkeit einer Kreuzung zweier Chromosomen darstellt. Im Folgenden wird mit dem Parameter $p_K = 0.5$ gearbeitet, da so das Verhältnis der gekreuzten und der unveränderten Chromosome der Wahrscheinlichkeit nach ausgeglichen ist. Dieser Parameter könnte jedoch auch von den Eigenschaften der Chromosomen oder Funktionen, wie der *Hamming-Distanz* eines Chromosompaares, abhängen.[7]

Kreuzen sich zwei Chromosome, werden alle Genpaare mit gleichem Index betrachtet und abhängig von einem weiteren Parameter $p_W \in [0, 0.5]$ vertauscht. Dieser Parameter stellt die Wahrscheinlichkeit des Wechsels jedes einzelnen Gens dar. Zur Ermittlung von p_W wird die Formel

$$p_W = \frac{\mu_W}{n} \tag{10}$$

empfohlen, wobei μ_W den Erwartungswert der Anzahl getauschter Gene darstellt. Alle hier dargestellten Algorithmen und Grafiken wurden mit dem Erwartungswert $\mu_W = 1$

[7] *Hamming-Distanz* eines Chromosompaares: Die Anzahl von Genenpaaren mit gleichem Index und unterschiedlicher Ausprägungen aus je einem Chromosom. Diese Distanz wird berechnet durch: $H(c_u, c_v) = n - \sum_{i=1}^{n} \delta(c_{u,i} - c_{v,i})$, wobei δ das Kronecker-Delta ist.

und somit einer Wechselwahrscheinlichkeit von $p_W = n^{-1}$ erzeugt. Im folgenden Algorithmus wird die Vorgehensweise dieses sogenannten ***Uniform-Crossover*** oder auch ***Kreuzung nach dem Binomialschema*** genauer beschrieben.

1: **procedure** [CHR1*, CHR2*] = CROSS(CHR1, CHR2)
2: **Input:**
3: *CHR1* - Erstes Chromosom der Länge N
4: *CHR2* - Zweites Chromosom der Länge N
5: *Parameter:* p_K - Kreuzungswahrscheinlichkeit
6: *Parameter:* p_W - Wechselwahrscheinlichkeit

7: **Initialisierung:**
8: *CHR1*=CHR1*
9: *CHR2*=CHR2*

10: **if** rand$(0, 1) < p_K$ **then**
11: **for** $i = 1$ bis N **do**
12: **if** rand$(0, 1) < p_W$ **then**
13: Tausche Ausprägungen des i-ten Genes von CHR1* und CHR2*
14: **end if**
15: **end for**
16: **end if**

17: **Output:** CHR1* und CHR2*
18: **end procedure**

In Abbildung 6 wird eine gekreuzte Population, erzeugt aus der Population in Abbildung 5, dargestellt. Wie zu erkennen ist, wurden unterschiedliche Ausprägungen nur zwischen Chromosom 5 und 6 ausgetauscht. Da in diesem Beispiel die Anzahl an Allelen sehr begrenzt ist, haben die Gene der Chromosomen zu einem großen Anteil die gleichen Ausprägungen, daher sind einige Vertauschungen nicht erkennbar.

2.3.2 Mutationen

Allgemein, reicht es nicht aus die Chromosomen zu kreuzen, da durch diese Operation der Suchraum nicht zwangsläufig komplett durchsucht werden kann. Unter Umständen kann es vorkommen, dass gewisse Allele nicht in der Startpopulation erzeugt wurden, diese können durch das reine Austauschverfahren der *Kreuzung* nicht erzeugt werden. Neben

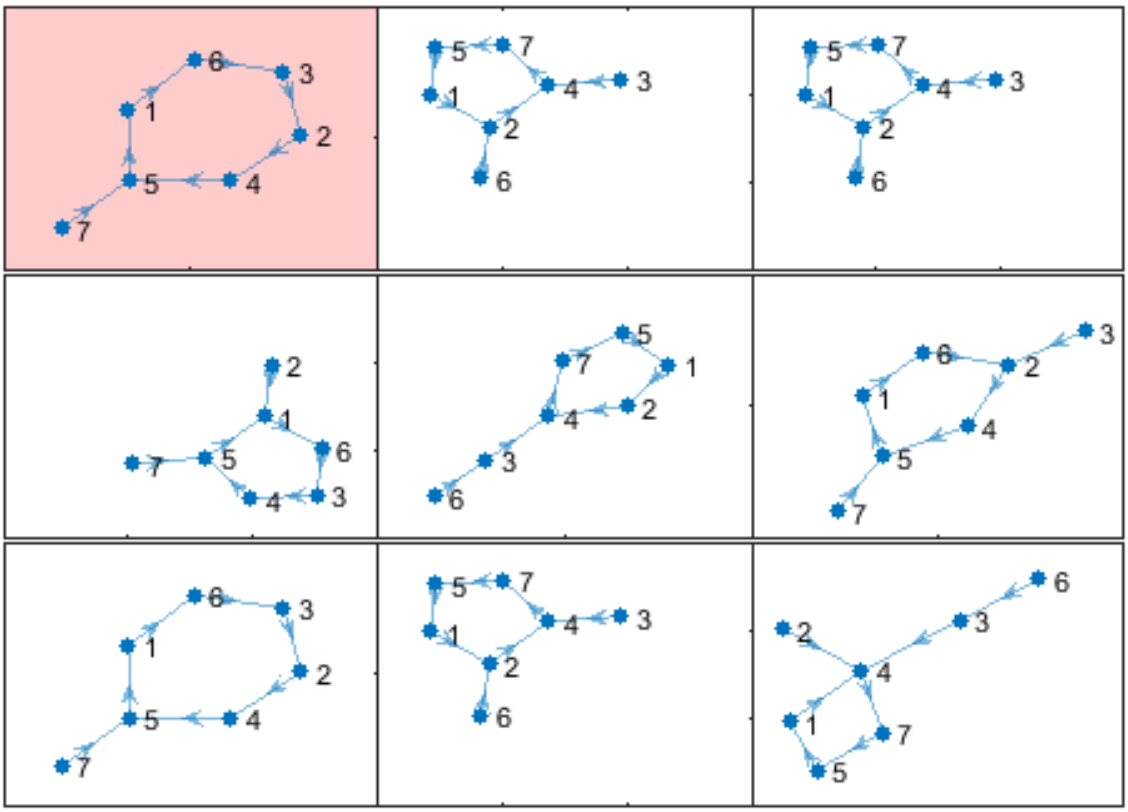

Abbildung 6: *Zwischenpopulation nach der Kreuzung.*

der Einführung eines zufällig neu generierten *Alien* in jedem Iterationsschritt, können die Gene eines Chromosomes zufällig mutieren, dies bedeutet, dass sich die Ausprägungen der Gene, abhängig von einem Wahrscheinlichkeitsparamater $p_M \in [0, 1]$, zufällig verändern können. Hierbei ist wieder zu beachten, dass die neue Ausprägung eines Genes g_i nur mögliche Allele, festgelegt durch Ψ_i bzw. Φ_i, annehmen kann. Die Auswahl des Mutationsparameters sollte abhängig von der Gesamtanzahl an Knoten n sein. Die empfohlene Formel zur Ermittlung des Parameters ist

$$p_M = \frac{\mu_M}{n}, \tag{11}$$

wobei μ_M den Erwartungswert der Anzahl veränderter Gene darstellt. Alle hier dargestellten Algorithmen und Grafiken wurden mit dem Erwartungswert $\mu_M = 1$ und somit einem Mutationsparameter von $p_M = n^{-1}$ erzeugt. Das genaue Verfahren der Mutation wird im folgenden Algorithmus schematisch dargestellt.

1: **procedure** [CHR*] = MUTATE(CHR)
2: **Input:**
3: *CHR* - Chromosom der Länge N
4: *Parameter:* p_M - Mutationswahrscheinlichkeit

Initialisierung:
CHR=CHR*

for $i = 1$ bis N **do**
  if rand$(0, 1) < p_M$ **then**
    Setze Ausprägung von Gen g_i aus CHR* zufällig und gleichverteilt
    Primal: Auswahl aus Ψ_i
    Dual: Auswahl aus Φ_i
  end if
end for

Output: CHR*
end procedure

Abbildung 7 zeigt die Population aus Abbildung 6 nach der Mutation. Wie zuvor beschrieben gilt hier für den Mutationsparameter $p_M = n^{-1} = \frac{1}{7}$. Auch hier ist wieder zu berücksichtigen, dass jedes Gen nur einige wenige Allele zur Auswahl hat und daher nicht jede Mutation erkennbar ist. Das fünfte und siebte Chromosom sind jedoch sichtlich mutiert.

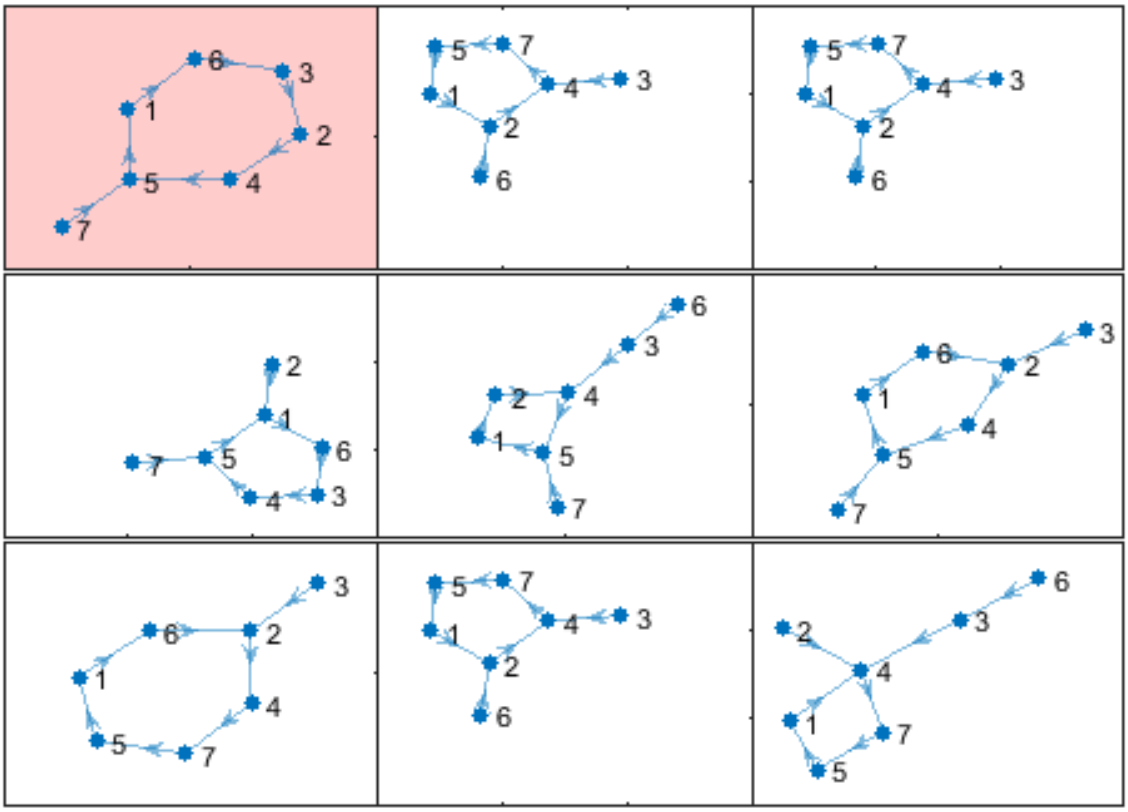

Abbildung 7: *Zwischenpopulation nach der Mutation.*

2.4 Abbruchbedingungen und Konvergenz

Wie bereits in Abschnitt 2.2 beschrieben, ist das Problem eindeutig gelöst, wenn als Fitnesswert eines Chromosomes 1 ermittelt wurde. Dieses Chromosom kodiert einen gesuchten Hamiltonzyklus durch den Digraphen. Überprüft man den Fitnesswert innerhalb des Algorithmus auf diese Bedingung, kann dies als erste Abbruchbedingung verwendet werden. Dies reicht jedoch meist nicht aus, da nicht sichergestellt ist, dass ein Hamiltonzyklus gefunden wird, geschweige denn überhaupt einer existiert. Bei der Entwicklung eines solchen Algorithmus muss immer eine weitere Abbruchbedingung hinzugefügt werden, um sicher zu stellen, dass der Algorithmus beendet wird. Eine einfache und effektive, weitere Abbruchbedingung, wäre eine maximale Anzahl an erzeugten Generationen t_{max}. Diese maximale Iterationsanzahl sollte abhängig von der Größe der Population und einem Steuerparameter δ gewählt werden:

$$t_{max} = \delta \cdot P^2 \tag{12}$$

Die Rechenkomplexität der Bewertung eines Chromosomes ist nach Gleichung (6) linear abhängig von der Anzahl der Knoten. Da sowohl die Kreuzung, als auch die Mutation ebenfalls linear abhängig davon sind (einfache **for**-Schleife), ist die Verarbeitung eines Chromosomes innerhalb des Algorithmus insgesamt linear abhänig von der Anzahl der Knoten. Die Rechenkomplexität der Selektion ist linear abhängig von der Populationsgröße und diese ist per Konstruktion linear abhängig von der Anzahl an Kanten m (vgl. Gleichung (3)). Daher ergibt sich für die Erzeugung einer neuen Population:

$$\mathcal{O}(n \cdot m) \tag{13}$$

2.4.1 Stagnation

Hat sich der Fitnesswert des besten Chromosomes jeder Generation nach einer festgelegten Anzahl an erzeugten Generationen Δt nicht verbessert, *stagniert* das System. Es ist davon auszugehen, dass mit hoher Wahrscheinlichkeit auch in folgenden Schritten keine Verbesserung erzielt werden kann. An dieser Stelle ist es sinnvoll, die Kodierung der Chromosomen bzw. der Gene zu wechseln, d.h. von *primal* auf *dual* und umgekehrt. Diese Umkodierung ist jedoch nicht immer vollständig möglich, das heißt es können nicht alle Kanten der alten Kodierung in die neue übertragen werden. Primal kann jeder Knoten nur eine Ausgangskante, aber mehrere Eingangskanten haben und dual dreht sich dieses Verhältnis um. Betrachtet man das primale Chromosom aus Abbildung 3, ist zu erkennen, dass mehrere Gene die gleiche Ausprägung haben. Die Ausprägung 2 in Gen g_1 wird umkodiert zu der Ausprägung 1 in g_2, nun kann jedoch nicht mehr die Ausprägung 2 von Gen

g_3 umkodiert werden, da die Ausprägung von g_2 bereits gesetzt wurde. Zunächst werden jedoch alle möglichen Umkodierungen vorgenommen:

Primal:	2	4	2	5	1	2	5
				↓			
Dual:	5	1	*	2	4	*	*

Die Gene g_3, g_6 und g_7 sind nun nicht ausgeprägt, da in der vorherigen primalen Darstellung kein Gen die Ausprägung 3, 6 oder 7 hatte. Diese Gene müssen nun repariert werden, indem sie zufällige Ausprägungen aus den möglichen Allelen annehmen. Hierbei ist wieder zu beachten, dass die möglichen Allele für ein primales Gen g_i alle k sind, wofür $(i, k) \in \Psi_i$ gilt. Die Allele eines dualen Gens g_j sind gegeben durch alle k, sodass $(k, j) \in \Phi_j$ gilt. Eine vollständige Umkodierung ist hier also nur möglich, falls alle Ausprägungen des ursprünglichen Chromosomes unterschiedlich sind. Dies bedeutet, dass das Chromosom nur aus Zyklen bestehen darf.

Mit Hilfe dieses Verfahrens bleiben die positiven Eigenschaften des Chromosomes erhalten, denn je mehr unterschiedliche Ausprägungen die Gene eines Chromosomes annehmen, umso näher ist es an einem nur aus Zyklen bestehenden System. Die Mutation von Genen, die ein mehrfach vorkommendes Allel haben, wird hier erzwungen, somit wird versucht eine negative Eigenschaft zu entfernen. Des Weiteren wird der Suchraum verändert und die Neukodierung der Chromosome ermöglicht das Durchsuchen anderer Pfade durch den Digraphen. Der Parameter Δt, der bestimmt, wann der Prozess stagniert, sollte abhängig von einem oder mehrerer folgender Parameter sein:

- n: Knotenanzahl,
- m: Kantenanzahl,
- t_{max}: Maximale Anzahl erzeugter Generationen,
- λ: Größe des Suchraumes.

In den hier vorgestellten Algorithmen ist der Stagnationsparameter nur abhängig von der maximalen Anzahl erzeugter Generationen:

$$\Delta t = \left\lceil \sqrt{t_{max}} \right\rceil \tag{14}$$

Das in Beispiel 1.1 dargestellte Problem konnte jedoch gelöst werden, ohne dass es zu einer Stagnation kam. In Abbildung 8 wird die 8. Generation dargestellt, in der für das fünfte und siebte Chromosom der maximal mögliche Fitnesswert $F = 1$ ermittelt wurde. Wie gut zu erkennen ist, stellen diese Chromosome den zuvor bereits mit bloßem Auge ermittelten Hamiltonzyklus $1 - 6 - 3 - 2 - 4 - 7 - 5 - 1$ dar.

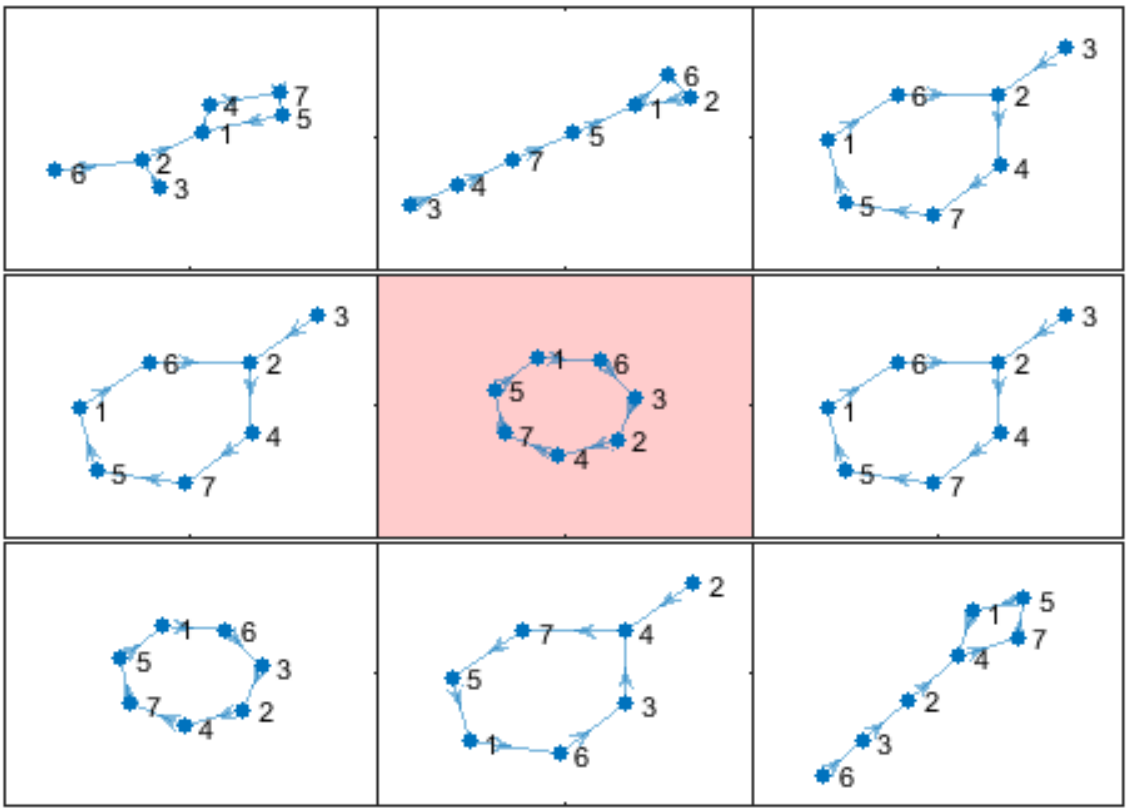

Abbildung 8: *Letzte Population.*

2.4.2 Elitärer Algorithmus

Damit ein Algorithmus *elitär* ist, darf sich das beste Chromosom einer Population nach einem Iterationsschritt nicht verschlechtert haben. Es gilt

$$F_b^{t+1} \geq F_b^t \quad \forall t \geq 0, \tag{15}$$

wobei F_b^t die Bewertung des besten Chromosomes nach Iterationsschritt t ist. Dies ist einfach zu erreichen, indem das beste Chromosom einer Population (*Elite-Chromosom*) vor der Mutation und Rekombination zwischengespeichert wird, um danach wieder hinzugefügt zu werden. Die bisher beste Lösung wird somit nie verworfen oder verändert, das *Elite-Chromosom* bleibt erhalten.

Zu Problemen kommt es jedoch bei der in Abschnitt 2.4.1 eingeführten Stagnationskontrolle durch Umkodierung. Wird das Elite-Chromosom umkodiert, behält es zwar die meisten seiner guten Eigenschaften, jedoch ändert sich womöglich die Fitness des Chromosomes, da meistens nicht alle Gene übernommen werden können und einige zufällig neu bestimmt werden. Eine Kreuzung von primalen und dualen Chromosomen ist, so wie sie hier implementiert wurde, nicht möglich, da die Gene in der primalen und dualen Kodierung unterschiedliche Eigenschaften repräsentieren. Daher kann dieses Elite-Chromosom nicht der neuen Population hinzugefügt werden, falls es nicht umko-

diert wird. Dieses Problem lässt sich beheben, indem zwei Populationen parallel generiert und iteriert werden, eine primal und eine dual kodiert. Kommt es nicht zur Stagnation, übernimmt jede Population in jedem Iterationsschritt das eigene Elite-Chromosom der vorherigen Generation, wie soeben beschrieben. Muss jedoch umkodiert werden, übernimmt die erste Population das *Elite-Chromosom* von der zweiten und umgekehrt. Auf diese Weise müssen die beiden *Elite-Chromosome* nie umkodiert werden, da sie die beste Lösung für jeweils *primale* und *duale* Kodierung repräsentieren.

3 Optimierung der Parameter

Das folgende Kurzbeispiel wurden aus Übersichtsgründen nur mit einer einzigen Population, also nicht *elitär*, durchgeführt. Dies ist bei kleinen Digraphen von bis zu 50 Knoten auch kein Problem, darüber hinaus sollten die in Abschnitt 2.4.2 eingeführten Strategien angewendet werden.

Der Suchraum des Problems aus Beispiel 1.1 hat nach Gleichung (1) und Gleichung (2) einen Umfang von insgesamt $\lambda = \lambda_p + \lambda_d = 48 + 36 = 84$ verschiedenen Chromosomen. Es wurden neun Chromosome pro Generation erzeugt und nach acht Generationen wurde ein Hamiltonzyklus gefunden, daher sind insgesamt $8 \cdot 9 = 72$ Chromosome betrachtet worden. Dies scheint nicht besonders beeindruckend, da die Anzahl betrachteter Chromosomen nahezu so groß ist wie die Anzahl an möglichen Chromosomen. Größere Digraphen, wie der in Abbildung 9 dargestellte mit 20 Knoten und 102 Kanten, benötigen jedoch im Allgemeinen wesentlich weniger Chromosomen zur Berechnung, als zum Suchraum gehören. Die erzeugte Startpopulation ist in Abbildung 10 zu sehen. Nach 80 Generationen ermittelte der Algorithmus einen Hamiltonzyklus durch das System, erkennbar in Abbildung 11. Da die Populationsgröße $P = 49$ beträgt und 80 Generationen erzeugt wurden, sind insgesamt $49 \cdot 80 = 3920$ Chromosome untersucht worden. In diesem Beispiel beträgt die Größe des Suchraumes $\lambda \approx 5.8842 \cdot 10^{13}$.

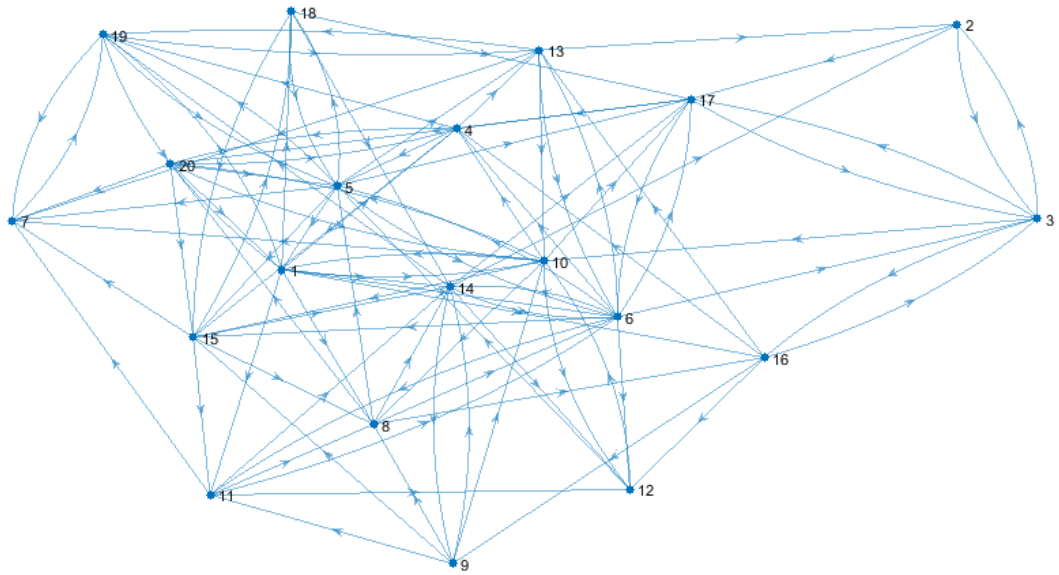

Abbildung 9: *Digraph mit 20 Knoten und 102 gerichteten Kanten.*

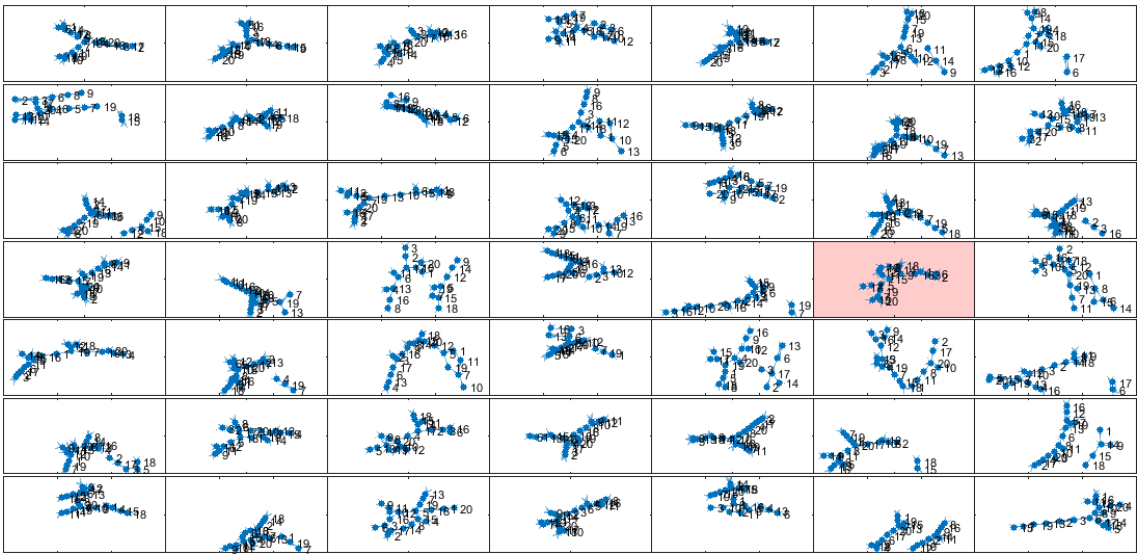

Abbildung 10: *Zufällig generierte Startpopulation zum Digraphen aus Abbildung 9 mit $P = 49$ Chromosomen. Das* **rot** *hinterlegte Chromosom hat die aktuell beste Bewertung.*

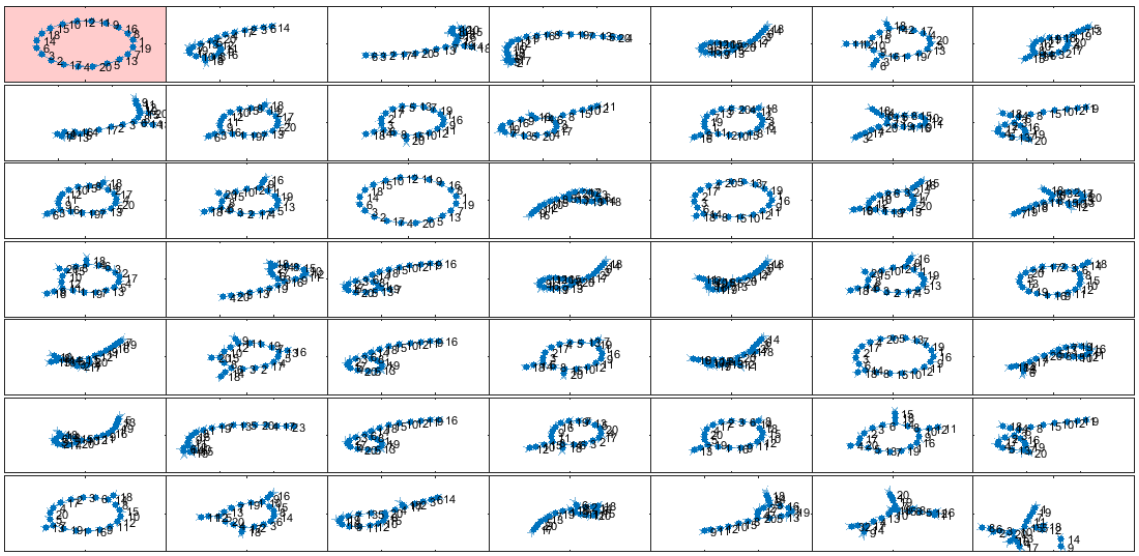

Abbildung 11: *80. Generation der Population zum Digraphen aus Abbildung 9, bei der eine Lösung ermittelt wurde. Das* **rot** *hinterlegte Chromosom kodiert einen Hamiltonzyklus.*

3.1 Parameterdefinition

Den hier vorgestellten genetischen Algorithmus für unterschiedlich große Digraphen und insbesondere für beliebige Topologien gleichermaßen zu optimieren ist nicht das Ziel. Die zu untersuchende Digraphentopologie sollte vorab ermittelt werden und die in Abschnitt 3.2 vorgestellte Erzeugung der Digraphen zur Durchführung von Testreihen muss entsprechend angepasst werden. Der Algorithmus kann durch die Definition von sechs verschiedenen Parametern, dargestellt in Tabelle 2, gesteuert werden.

Parameter	Kurzbeschreibung	Anwendung
α	Steuerparameter für die Fitnessfunktion	$F = [(D \cdot Z)/(\sqrt{\zeta} \cdot n^2)]^\alpha$
ϕ	Steuerparameter für die Populationsgröße	$P = \phi \cdot m$
δ	Steuerparameter für die Iterationsanzahl	$t_{max} = \delta \cdot P^2$
μ_M	Erwartungswert der Anzahl mutierter Gene	$p_M = \mu_M / n$
p_K	Kreuzungswahrscheinlichkeit	p_K
μ_W	Erwartungswert der Anzahl getauschter Gene	$p_W = \mu_W / n$

Tabelle 2: *Steuerparameter des Genetischen Algorithmus*

In den folgenden Diagrammen wird die Anzahl erzeugter Chromosomen, das heißt *Lösungsanwärtern*, dargestellt, da die Rechenkomplexität zur Erzeugung eines solchen, wie bereits in Abschnitt 2.4 beschrieben, sehr konstant und linear abhängig von der Anzahl an Knoten ist. Das Ziel ist es eine Kombination von Parametern zu finden, für die der Algorithmus, im Hinblick auf die in Tabelle 3 dargestellten Kriterien, möglichst optimal funktioniert.

Kriterium	Kurzbeschreibung	Optimierungsziel
μ	Durchschnittliche Anzahl erzeugter Chromosome	Geschwindigkeit
σ	Standardabweichung der Anzahl von Chromosomen	Streuung
ψ	Prozentsatz nicht erfolgreicher Testläufe	Sicherheit

Tabelle 3: *Auswertungskriterien für die Optimierung*

Bei der Variation der Parameter zur Ermittlung einer guten Parameterkombination muss jedoch bedacht werden, dass diese nicht voneinander isoliert optimiert werden können. Ein einfaches Beispiel hierfür sind die Steuerparameter ϕ und δ, die beide Einfluss auf die maximale Iterationsdauer t_{max} haben. Für die jeweiligen Optimierungsziele wird die ermittelte Lösung vorgestellt.

3.2 Auswertung

Um Testreihen auf unterschiedlichen Digraphen durchzuführen, müssen diese zunächst zufällig erzeugt werden. Zu Beginn muss die Anzahl der Knoten n und die ***Existenzwahrscheinlichkeit*** jeder Kante p_E des Test-Digraphen festgelegt werden. Darauf folgend wird durch eine zufällige Permutation der Knoten 1 bis n eine Rundreise generiert und die zugehörigen Kanten werden in die Adjazenzmatrix A eingetragen. Mit Hilfe der ***Existenzwahrscheinlichkeit*** p_E wird nun für jedes $a_{ij} \in A$, sodass $i \neq j$ gilt, zufallsbasiert ermittelt, ob die Kante existiert und der passende Wert wird in A eingetragen. Auf diese Weise ist sichergestellt, dass der betrachtete Digraph ***hamiltonsch*** ist, da die zu Beginn erzeugte Rundreise einen ***Hamiltonzyklus*** durch den Digraphen darstellt. Die genutzten Digraphen wurden mit einer Wahrscheinlichkeit von $p_E \in [\frac{1}{n}, \frac{1}{\sqrt{n}}]$ erzeugt.

In den folgenden Testreihen wird der ***elitäre*** Algorithmus, eingeführt in Abschnitt 2.4.2, angewandt. Da der genetische Algorithmus ***stochastisch*** ist, muss eine ganze Reihe an Testläufen durchgeführt werden, um eine Parameterkombination bewerten zu können. Grundlage einer Bewertung ist eine ***Monte-Carlo-Simulation*** über θ Testläufe bei identischen Digraphen, wobei die Anzahl erzeugter Chromosome jedes Testlaufes in 21 Kategorien eingeteilt wird. Die ersten 20 Kategorien (blau) ordnen, in 5%-Schritten ansteigend, die Testläufe anhand des prozentualen Anteils der erzeugten Chromosome zu der maximalen Anzahl an Chromosomen a_{max} ein. Schafft ein Testlauf es nicht innerhalb der maximalen Anzahl an Chromosome eine Lösung zu ermitteln, wird es in die 21. Kategorie (rot) eingeordnet. Ebenfalls dargestellt ist der prozentuale Anteil dieser 21. Kategorie ψ, der Erwartungswert μ und die Standardabweichung σ. Testläufe der 21. Kategorie gehen hier mit a_{max} in die Berechnung ein. Das Ergebnis einer solchen Monte-Carlo-Simulation ist eine Schätzung der Verteilungsfunktion, exemplarisch dargestellt in Abbildung 12. Die Parameterreihenfolge stimmt mit der Reihenfolge aus Tabelle 2 überein.

Um diese Parameter nun zu optimieren, wurde zunächst ein Optimierungsziel aus Tabelle 3 gewählt. Es wurden parallelisiert auf kleinen Digraphen mit $n = 20$ Knoten Monte-Carlo-Simulationen über $\theta = 1.000$ Testläufen durchgeführt. Die zu optimierenden Parameter wurden alle mit einem Startwert von 1 initialisiert und mit Hilfe eines simplen *Hill-Climbing-Algorithmus* wurden diese initialen Parametersätze für den gewählten Parameter in 1000 Schritten minimiert, indem die Parameter in jedem Schritt minimal variiert wurden.[8] Daraufhin wurden die ermittelten Parametersätze als initiale Parametersätze für die gleiche Untersuchung mit größeren Digraphen verwendet, die

[8] Zu Beginn des *Hill-Climbing-Algorithmus* ist eine Variation eines Parameters von bis zu ± 0.1 möglich (zufällig ermittelt). In jedem Iterationsschritt wird diese maximale Veränderung um 1% gesenkt.

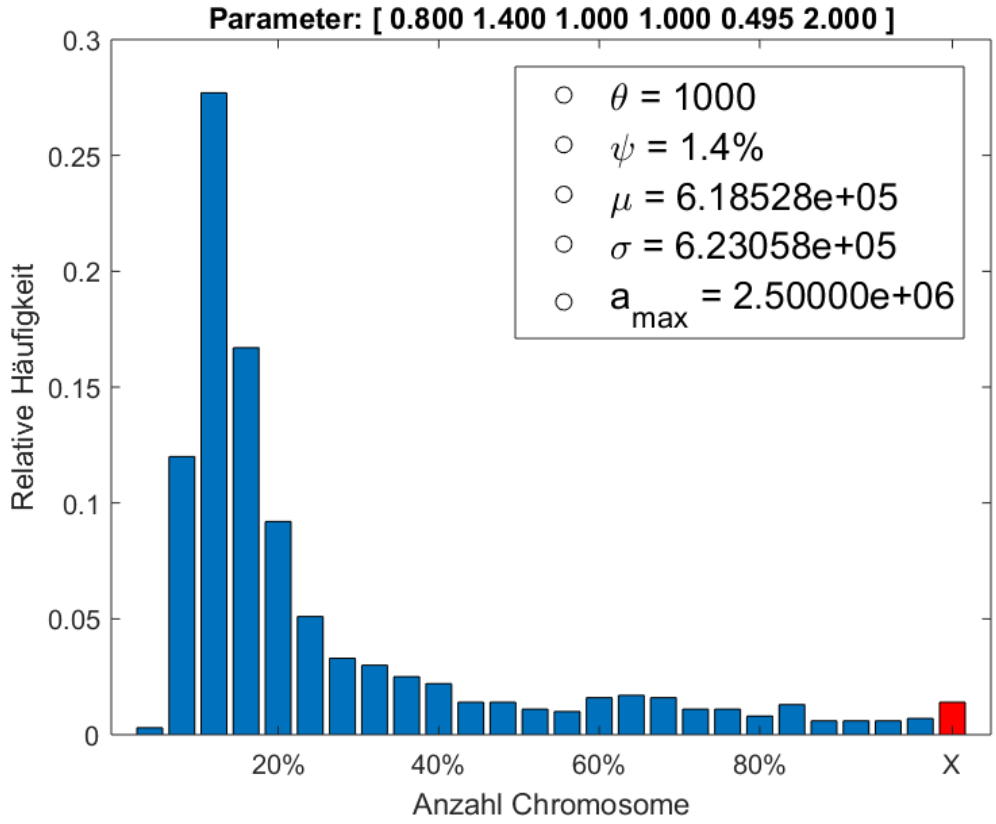

Abbildung 12: *Monte-Carlo-Simulation des genetischen Algorithmus über einen Digraphen mit $n = 40$ Knoten und $m = 249$ gerichteten Kanten*

Ergebnisse dieser Berechnungen wurden dann wieder für Simulationen bei wiederum größeren Digraphen verwendet und so weiter. Hierbei ist zu beachten, dass die Anzahl an durchgeführten Testläufen und die Anzahl an Schritten des Hill-Climbing-Algorithmus sukzessive verringert werden sollte, da die Berechnungen für große Digraphen aufwändiger werden.

Dieses *Herantasten* an größere Digraphen ist insofern sinnvoll, als dass bei kleineren Digraphen weniger Rechenaufwand nötig ist und daher umfangreichere und somit auch aussagekräftigere Monte-Carlo-Simulationen gemacht werden können. Es müssen nicht so viele aufwändige Berechnungen mit größeren Digraphen durchgeführt werden, da die initialen Parametersätze schon voroptimiert sind. Des Weiteren lässt sich durch die parallelisierte Berechnung auf unterschiedlichen, zufällig erzeugten Topologien ermitteln, ob ein verallgemeinerungsfähiges Ergebnis ermittelt wurde. Da die Auswertung einer einzigen Parameterkombination durch die Monte-Carlo-Simulation sehr aufwändig ist und andere Optimierungsverfahren meist wesentlich mehr Auswertungen benötigen, wurde hier mit einem einfachen Hill-Climbing-Algorithmus gearbeitet. In Tabelle 4 sind die Ergebnisse der Optimierung dargestellt. Während der Optimierung fiel auf, dass die ersten beiden Nachkommastellen stabil konvergieren, es danach jedoch zu Schwankungen kommt. Mit einer größeren Anzahl an Testläufen pro *Monte-Carlo-Simulation* könnten

	Geschwindigkeit	Streuung	Sicherheit
α	1.08	0.98	1.16
ϕ	1.36	1.45	1.22
δ	0.94	1.20	1.29
μ_M	1.05	1.07	1.19
p_K	0.56	0.51	0.61
μ_W	1.95	2.03	2.07

Tabelle 4: *Die ermittelten optimalen Parameter für die verschiedenen Optimierungsziele.*

hier möglicherweise noch bessere Ergebnisse ermittelt werden. Die unterschiedlichen Optimierungsziele weisen gemeinsame Tendenzen auf, jedoch verhält sich der Algorithmus unter den verschiedenen Parametersätzen wie gewünscht anders.

Durch die Art der Erzeugung von Testdaten sind die hier berechneten Parametersätze dafür optimiert, gleichmäßig verteilte Digraphen zu lösen. Für strukturell schwierige oder generell anders verteilte Digraphen müssten erneut Parameter ermittelt werden.

Der Algorithmus wurde in ***c++*** programmiert und das ***Hill-Climbing-Verfahren*** wurde in einem ***bash-Script*** mit wenigen Befehlen umgesetzt.

4 Ausblick und Fazit

Neben der Optimierung der Parameter, wie in Kapitel 3 beschrieben, könnte der Algorithmus durch die Anwendung anderer Selektions-, Mutations- bzw. Kreuzungsstrategien verbessert werden. Die angewandten Methoden haben sich in Testreihen bewährt, jedoch konnte aufgrund des hohen Rechenaufwands keine tiefgreifende Analyse aller möglichen Strategien durchgeführt werden. Des Weiteren muss bei der Wahl einer alternativen Strategie bedacht werden, dass sich die optimalen Parameter verändern und unter Umständen sogar neue hinzukommen. Diese müssten also erneut ermittelt werden. Einige mögliche alternative Strategien sind der folgenden Auflistung zu entnehmen.

Rouletteradselektion mit n Zeigern Anstatt jedes neue Chromosom einzeln anhand der Wahrscheinlichkeiten durch *Rouletteradselektion* mit nur einem Zeiger zu ermitteln, könnte mit n äquidistanten Zeigern direkt die gesamte nächste Generation erzeugt werden. Der Vorteil hierbei ist, dass nur eine einzige Zufallszahl ermittelt werden muss, jedoch wären die Chromosome der ermittelten Generation nicht mehr zufällig angeordnet. Dies müsste bei der Kreuzung mit bedacht werden.

Wettkampfselektion Hier würde jedes neue Chromosom durch den Wettkampf zweier Chromosomen der vorherigen Generation ermittelt werden. Ist bei der Auswahl zum Wettkampf sichergestellt, dass jedes Chromosom mindestens einmal kämpft, ist dieser Algorithmus *elitär* und die künstliche Injektion des besten Chromosomes der vorherigen Generation, wie in Abschnitt 2.4.2, wäre nicht nötig (Ausnahme: bei Umkodierung).

k-Punkt-Mutation Es wird ein Parameter k festgelegt, der besagt wie viele Gene mutiert werden sollen. Bei der Mutation werden zufällig k Gene des Chromosomes ermittelt und ihnen werden zufällig neue Allele zugeteilt. Der Vorteil hierbei ist, dass sichergestellt ist, dass bei jeder Mutation auch tatsächlich eine Veränderung stattfindet, jedoch führt dieses Verfahren bei $k > 1$ meist dazu, dass die Chromosome zu stark verändert werden und das Verfahren langsamer konvergiert.

k-Punkt-Kreuzung Es wird ein Parameter k festgelegt, der besagt, an wie vielen Stellen das Chromosom geteilt werden soll. Bei der Kreuzung werden zufällig k Gene als Schnittstellen ermittelt, diese Schnittstellen geben an, welche Teile des Chromosomes getauscht werden. Der Abschnitt vom ersten Gen zur ersten Schnittstelle

wird nicht getauscht, der darauf folgende zweite Abschnitt wird getauscht, der dritte wiederum nicht und so weiter. Diese Art der Kreuzung wird auch ***Kreuzung nach dem Exponentialschema*** genannt.

Alle verwendeten Operatoren sind stochastische und somit heuristische Funktionen. Eine Erweiterung des Algorithmus durch Veränderungen der Chromosome auf deterministischer Basis könnte von Vorteil sein. Abbildung 13 zeigt exemplarisch einen Zyklus $(1-7-6-5-4-1)$ innerhalb eines primalen Chromosomes, mit einem anschließenden Pfad $(3-2-1)$. Existiert die Kante $(4-3)$, dargestellt in grün, könnte sie die vorherige Kante $(4-1)$, dargestellt in rot, ersetzen und das Chromosom würde die Lösung des Problems kodieren. Ist dies nicht möglich, würde durch die Kante $(4-2)$ zumindest die Bewertung verbessert werden. Manipulationen dieser Art können leicht

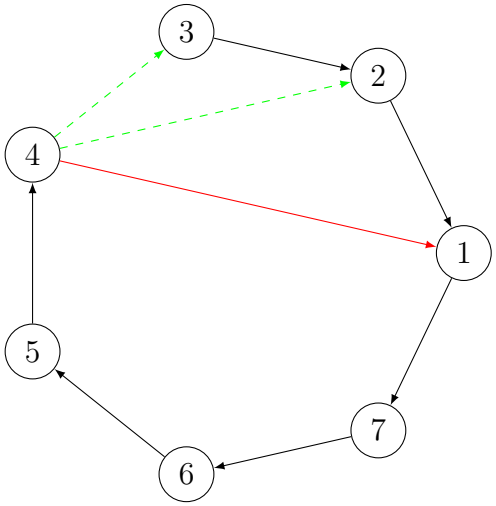

Abbildung 13: *Ein Zyklus mit Pfad eines primalen Chromosomes.*

in den Markierungsalgorithmus zur Ermittlung der Fitness eingebunden werden, jedoch sollten diese Veränderungen an Chromosomen nicht in jedem Iterationsschritt und nicht an jedem Chromosom durchgeführt werden. Eine zu starke Beeinflussung des Verfahrens führt zu vorzeitiger Konvergenz durch Verkleinerung des Suchraumes (des *Genpools*), da viele vorher unterschiedliche Chromosome zu Äquivalenten *umoperiert* werden würden. Dies trifft auch auf das Verfahren, dargestellt in Abbildung 14, zu. Dort ist ein Chromosom mit zwei separaten Zyklen dargestellt, welche durch die dargestellte Veränderung zu einem einzigen Zyklus kombiniert werden könnten. Das Verfahren überprüft für jede vorhandene Kante, die von einem Knoten des einen Zyklus auf einen Knoten des anderen Zyklus zeigt (vgl. Kante $(5a-4b)$), ob die jeweilige Kante $(3b-1a)$ zur Bildung eines gemeinsamen Zyklus existiert. Existiert ein solches Kantenpaar, können die alten Kanten (rot) durch die neuen Kanten (grün) ersetzt werden und die Bewertung des Chromosomes verbessert sich.

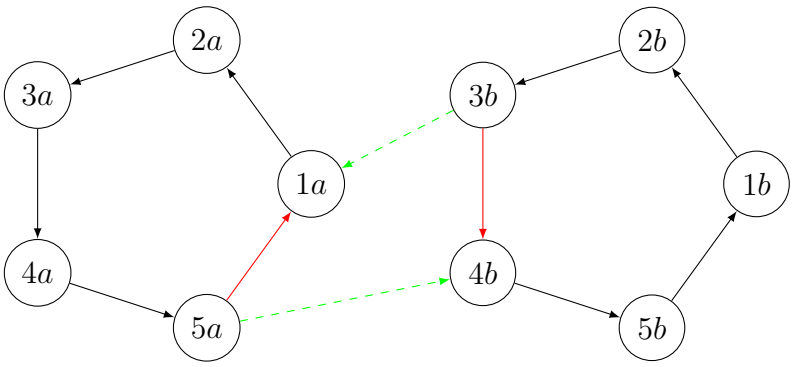

Abbildung 14: *Zwei Zyklen eines primalen Chromosomes*

Um die Güte des genetischen Algorithmus zu bewerten, muss er mit gängigen Algorithmen verglichen werden. Hierfür wurden bereits einige Vergleiche anhand von Tests über die reine Laufzeit durchgeführt, bei denen der genetische Algorithmus ab einer Knotenanzahl von ca. $n = 50$ besser abgeschnitten hat. Die getesteten Algorithmen waren zum einen ein Tiefensuche-Algorithmus, der in der Maple-Bibliothek[9] eingebunden ist, und zum anderen einige Algorithmen, die für die Lösung von Rundreiseproblemen, sogenannten *asymmetrischen Travelling-Salesman-Problemen*, entwickelt wurden.

Ein großer Vorteil des Genetischen Algorithmus ist die Parallelisierbarkeit. Die Bewertung und die Mutation einzelner Chromosome können gesondert voneinander durchgeführt werden und auch die Kreuzung lässt sich einfach parallelisieren, da nur nebeneinander liegende (indexweise aufeinanderfolgende) Chromosome gekreuzt werden. In jedem Zeitschritt müssen zur Selektion alle Informationen zusammengetragen und die neue Population erzeugt werden. Des Weiteren können unterschiedliche Populationen komplett parallel (ausgenommen zum Zeitpunkt der Umkodierung) berechnet werden. Wie bereits in der Einleitung erwähnt, ist ein weiterer großer Vorteil dieses Algorithmus, dass er direkt auf Digraphen anwendbar ist und keine Umwandlung in einen Graphen, bei der sich die Knotenanzahl verdreifacht, notwendig ist (vgl. Abbildung 1). Wichtig ist hierbei, dass der hier vorgestellte genetische Algorithmus für die Lösung von ***asymmetrischen*** Problemen, also *Digraphen* ausgelegt ist. Die Ermittlung eines ***Hamiltonkreislaufs*** in ungerichteten Graphen ist mit diesem Algorithmus zwar auch möglich, alternative Algorithmen, wie beispielsweise beschrieben in [2], werden jedoch empfohlen.

[9] Maple (*mathematical manipulation language*) ist ein *Computeralgebrasystem* für viele Teilgebiete der Mathematik von ***Maplesoft*** aus Kanada.

Literaturverzeichnis

[1] Bermond JC, Thomassen C: *Cycles in digraphs - a survey*, J. Graphs Theory 5 (1981) 1-43.

[2] Chalaturnyk A: *A Fast Algorithm For Finding Hamilton Cycles*, Masters thesis Computer Science, University of Manitoba, 2008.

[3] Gerdes I, Klawonn F, Kruse R: *Genetische Algorithmen und Optimierung. In: Evolutionäre Algorithmen. Computational Intelligence.* Vieweg+Teubner Verlag, 2004.

[4] Krauß V.: *Gene, Zufall, Selektion*, Springer Verlag, 2014.

[5] Kühn D, Osthus D: *A survey on Hamilton cycles in directed graphs*, European Journal of Combinatorics 33 (2012) 750 - 766.

Anhang

Im Folgenden wird der Markierungsalgorithmus aus Abschnitt 2.2.1 grafisch dargestellt und genauer erläutert. Die Markierungen ***Länge*** *(L)* und ***Status*** *(S)* eines Knotens i werden in den Abbildungen A.1 bis A.5 wie folgt dargestellt:

Die Markierungen ***Nachfolger*** und ***Vorgänger*** werden aus Übersichtsgründen vernachlässigt und gehen aus den gerichteten Kanten und den Erläuterungen im folgenden Text hervor. In jedem Zwischenschritt werden die durchlaufenen Kanten der Reihenfolge nach mit römischen Ziffern durchnummeriert, wird eine Kante entgegen der Pfeilrichtung entlanggelaufen, ist die Ziffer rot dargestellt.

Abbildung A.1 zeigt ein unmarkiertes primales Chromosom. Die Variablen ζ, Z, D und alle ***Längen*** *(L)* und ***Status*** *(S)* werden mit Null initialisiert. Zu Beginn beinhaltet ***Rest*** alle Knoten.

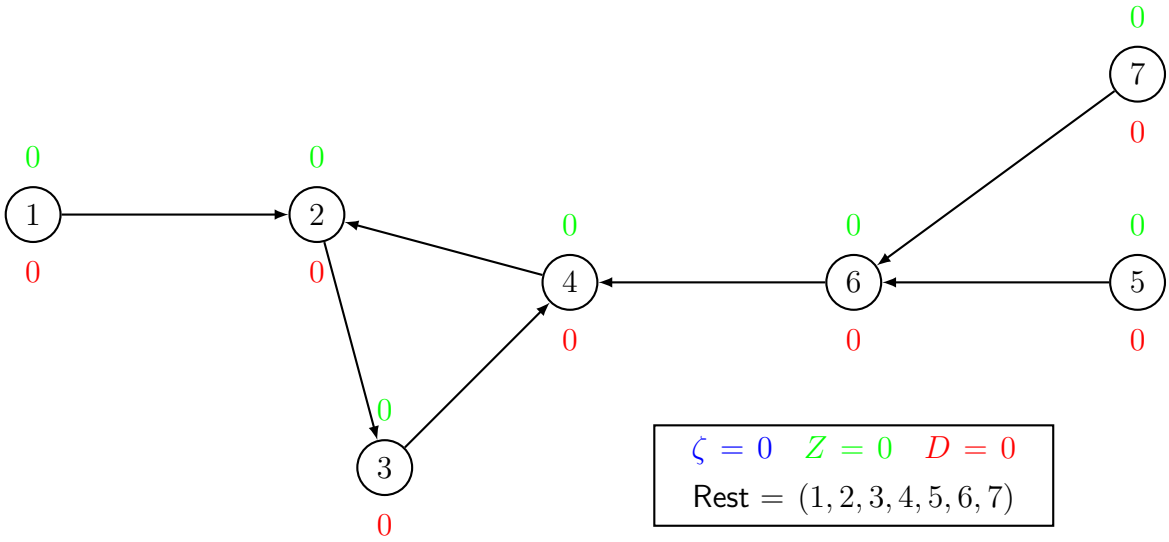

Abbildung A.1: *Unmarkiertes primales Chromosom*

In Abbildung A.2 beginnt der Algorithmus mit dem ersten Element aus ***Rest*** und traversiert durch den Graphen entlang der Pfeilrichtungen. Dieses Verfahren ist eindeutig, da

in einem primalen Chromosom jedem Knoten nur genau eine ausgehende Kante (***Nachfolger***) zugeordnet ist. In jedem dieser Schritte wird der ***Status*** *(S)* der besuchten Knoten auf 1 gesetzt und die ***Länge*** *(L)* auf die Anzahl der bereits besuchten Knoten gesetzt. Des Weiteren wird jeder besuchte Knoten aus ***Rest*** entfernt und der ***Vorgänger***, welcher im Vergleich zum ***Nachfolger*** nicht eindeutig ist, wird markiert. Dieser Prozess terminiert, wenn der Algorithmus auf einen Knoten trifft, dessen ***Status*** *(S)* nicht Null ist. Der ***Status*** *(S)* des betrachteten Knotens 2 ist 1, somit ist ein Zyklus gebildet worden. Hierbei ist zu erkennen, dass im ersten Durchgang zwangsläufig ein Zyklus durchlaufen wird, da die Knoten zu diesem Zeitpunkt nur die Statusmarkierung 0 oder 1 haben können. Nachdem der Zyklus erkannt wurde, wird nun seine Länge (Knotenanzahl = Kantenanzahl) durch

$$\text{LÄNGE}_{\text{ZYKLUS}} = \#\text{BESUCHTE KNOTEN} - \text{LÄNGENMARKIERUNG} \tag{16}$$

bestimmt, wobei die Längenmarkierung des aktuell betrachteten Knotens, in diesem Falle von Knoten 2, gemeint ist. Des Weiteren ist zu beachten, dass dieser Knoten zweimal besucht wurde, somit ergibt sich eine Zykluslänge von 3. Die Anzahl Zyklen ζ erhöht sich somit um 1 und die Anzahl an Knoten in allen Zyklen Z um die Zykluslänge.

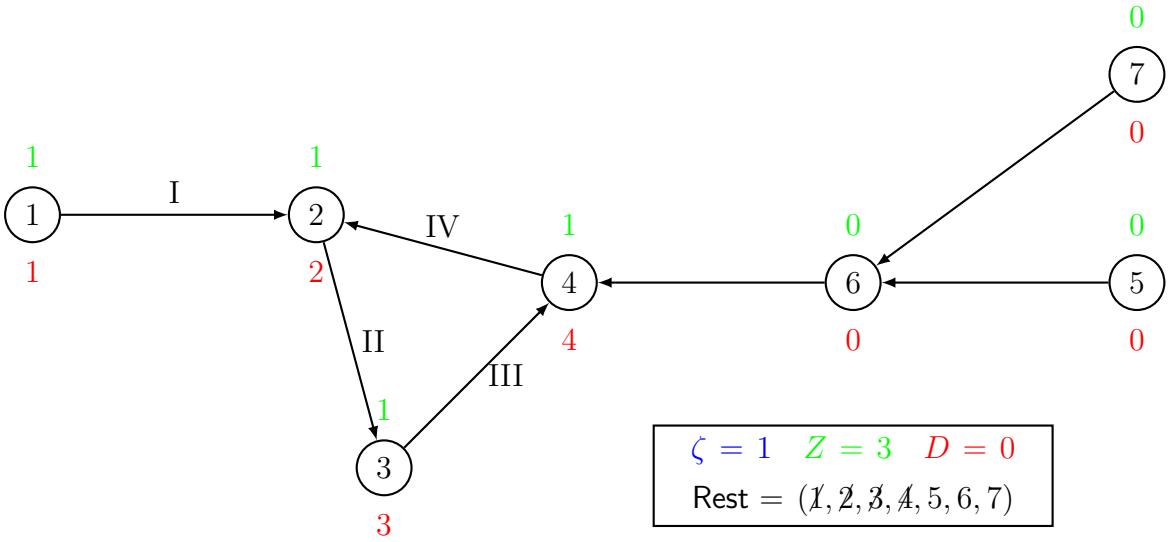

Abbildung A.2: *Initialisierung und Identifikation eines Zyklus*

In der nächsten Abbildung A.3 wird dieser Zyklus erneut durchlaufen, um dort alle ***Längen*** *(L)* auf die Zykluslänge und alle ***Status*** *(S)* auf 2 zu setzen. Hierbei terminiert der Algorithmus, wenn er auf einen Status ungleich 1 trifft. Des Weiteren wird danach der Pfad vor Zyklusbildung durch die Markierung ***Vorgänger*** zurückverfolgt und die Zykluslänge wird als Index verwendet, der nach jedem besuchten Knoten um 1 erhöht wird. Dieser Index ersetzt in jedem Schritt die alte ***Längenmarkierung*** *(L)* der Knoten des Pfades und der ***Status*** *(S)* wird auf 3 gesetzt. Es ist hierbei wichtig, die Markierung ***Vorgänger*** während des Zweitdurchlaufes des Zyklus nicht neu zu setzen, da sonst die Rückverfol-

gung des Pfades nicht mehr möglich ist. Hat ein Knoten keinen ***Vorgänger***, ist der Anfang des Pfades erreicht. Die aktuelle ***Längenmarkierung*** *(L)* wird mit D verglichen und das Maximum wird dort gespeichert.

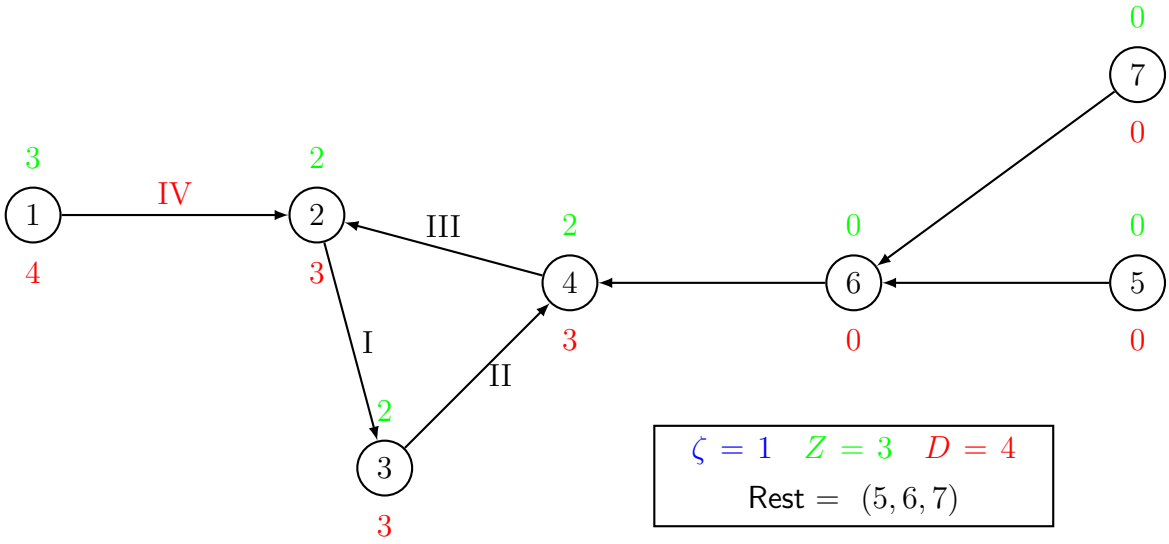

Abbildung A.3: *Zweitdurchlauf des Zyklus und Zurückverfolgung des Pfades*

Da ***Rest*** noch nicht leer ist, beginnt der Algorithmus erneut mit dem ersten Element aus ***Rest*** und traversiert (und markiert) wie beschrieben durch den Graphen, bis er auf einen Knoten mit einer ***Statusmarkierung*** *(S)* ungleich Null trifft, dargestellt in Abbildung A.4. In diesem Fall hat Knoten 4 die Statusmarkierung 2, somit mündet der Pfad in einen bereits erkannten Zyklus. Wie bereits beschrieben wird nun die Zykluslänge, die als ***Längenmarkierung*** *(L)* des betrachteten Knoten 4 zuvor eingespeichert wurde, als Index verwendet, um die ***Längenmarkierung*** *(L)* der Knoten des Pfades bei der Rückverfolgung neu zu bestimmen. Auch hier wird die ***Längenmarkierung*** *(L)* des letzten Knotens wieder genutzt, um D zu aktualisieren. In diesem Falle wird das neue Maximum 5 ermittelt.

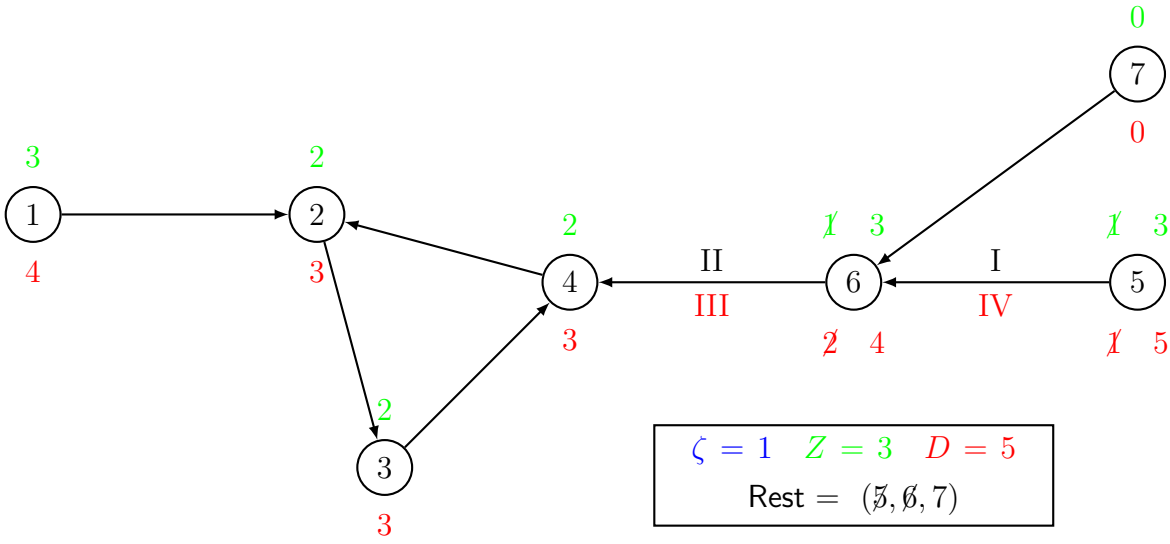

Abbildung A.4: *Initialisierung und Zurückverfolgung eines Pfades, der auf einen Zyklus trifft*

In Abbildung A.5 ist zu sehen, dass sich nur noch Knoten 7 in ***Rest*** befindet, der Algorithmus beginnt hier erneut. An dieser Stelle ist zu erkennen, dass der Algorithmus nach der Initialisierung nicht unterscheidet, ob er auf einen bereits bekannten Zyklus oder einen bereits bekannten Pfad trifft. Er verwendet die dort eingespeicherte ***Längenmarkierung*** *(L)* als Index und verfolgt den Pfad zurück bis zum Anfang. Die Unterscheidung zwischen einem Status von 2 (Zyklusknoten) und einem Status von 3 (Pfadknoten) ist rein aus Gründen der Verständlichkeit getroffen worden, jedoch nicht zwingend notwendig für den Algorithmus.

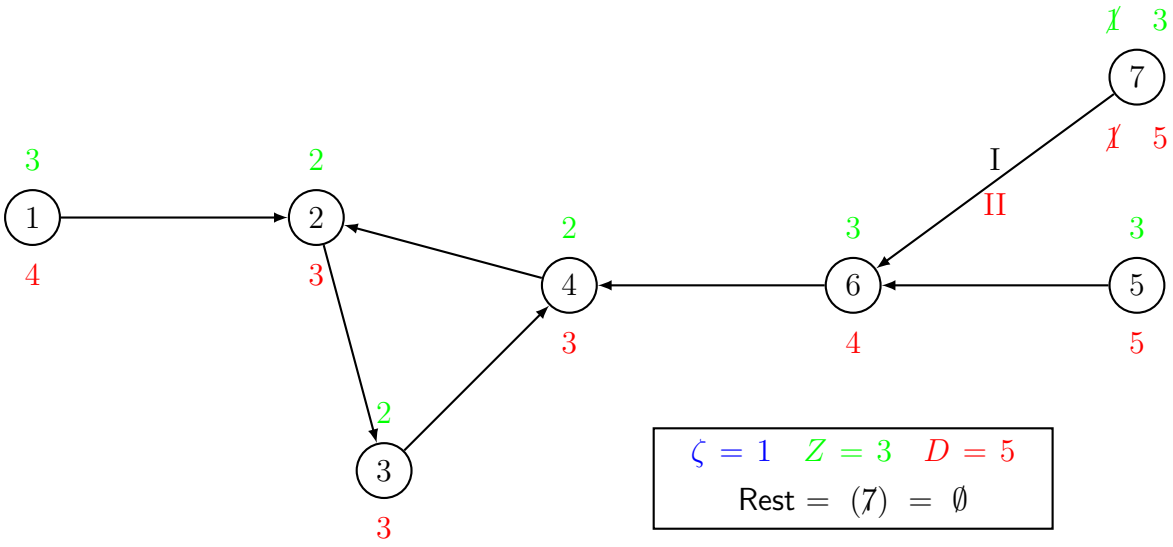

Abbildung A.5: *Initialisierung und Zurückverfolgung eines Pfades, der auf einen Pfad trifft*

Nehmen wir $\alpha = 1$ für den Steuerparameter der Fitnessfunktion an, ergibt sich nach Gleichung (5) für die Fitness dieses Chromosomes

$$F = \left(\frac{F^*}{n^2}\right)^{\alpha} = \left(\frac{D \cdot Z}{\sqrt{\zeta} \cdot n^2}\right)^{\alpha} = \left(\frac{5 \cdot 3}{\sqrt{1} \cdot 7^2}\right)^{1} = \frac{15}{49} \approx 0.3. \tag{17}$$

In den Abbildungen ist gut zu erkennen, dass jede Kante genau zweimal durchlaufen wird, daher ist die Rechenkomplexität dieses Algorithmus linear abhängig von der Anzahl der Kanten. Durch die Art der Kodierung besitzt jedes Chromosom genausoviele Knoten wie Kanten, daher ist er ebenfalls linear abhängig von der Anzahl der Knoten des ursprünglichen Digraphen.

Kontaktdaten

Autoren

Karim Kai Abdelhak, B. Sc.
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
Interaktion 1
33619 Bielefeld
Telefon +49.521.106-7402
kabdelhak@fh-bielefeld.de
Raum D 237

Prof. Dr. Bernhard Bachmann
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
Interaktion 1
33619 Bielefeld
Telefon +49.521.106-7407
Telefax +49.521.106-7176
bbachmann@fh-bielefeld.de
Raum D 230

Prof. Dr. Hermann-Josef Kruse
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
Interaktion 1
33619 Bielefeld
Telefon +49.521.106-7411
Telefax +49.521.106-7176
hkruse@fh-bielefeld.de
Raum D 229

FSP AMMO

Sprecherin

Prof. Dr. rer. nat. Svetozara Petrova
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
FSP Angewandte Mathematische Modellierung und Optimierung
Interaktion 1
33619 Bielefeld
Telefon +49.521.106-7410
Telefax +49.521.106-7190
svetozara.petrova@fh-bielefeld.de
Raum D 226

Stellv. Sprecherin

Dr. rer. nat. Sabrina Proß
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
FSP Angewandte Mathematische Modellierung und Optimierung
Schulstrasse 10
33330 Gütersloh
Telefon +49.5241.21143-21 (70121)
sabrina.pross@fh-bielefeld.de
Raum 303

Veröffentlichungsreihe: AMMO – Berichte aus Forschung und Technologietransfer

Heft 9: D. Nehab, T. Lask, H.-J. Kruse, *Programm zur Entscheidungsunterstützung für eine Klasse von WzP-Kommissionierungsproblemen*. März 2018.

Heft 8: J. Silberberg, T. Lask, B. Bachmann, *Formalismen für gefärbte Petri-Netze und Verfahren zur effizienten Bestimmung von aktiven Modus-Mengen*. Juli 2016.

Heft 7: T.F. Lye, H.-J. Kruse, T. Lask, *Heuristische Lösungsmethoden für eine Klasse von Ware-zum-Mensch-Kommissionierungsproblemen*. März 2016.

Heft 6: T. Kleine-Döpke, H.-J. Kruse, *Lösungsansätze für Konfliktsituationen bei Feuerprozessen in kapazitierten Petri-Netzen*. Juni 2015.

Heft 5: H.-J. Kruse, *Optimumgraphen*. Oktober 2014.

Heft 4: S. Proß, *Diskrete Modellierung und Optimierung praxisrelevanter Prozesse mit Petri-Netzen*. September 2014.

Heft 3: R. Ueckerdt, H.-W. Schmidt, M. Weber, E. Mindlina, *Entwicklung einer Dispatcherfunktion zur Überprüfung von Nominierungsmengen in der Betriebsführung von Erdgasspeichern*. Juli 2014.

Heft 2: R. Walden und V.-M. Roemer, *Methoden der quantitativen rechnergestützten CTG-Analyse*. April 2014.

Heft 1: AMMO-Team, *Informationen über den Forschungs- und Entwicklungsschwerpunkt Angewandte Mathematische Modellierung und Optimierung*. Dezember 2013.

ISSN 2198-4824
Herausgeber:
Vorstand von FSP AMMO
Fachhochschule Bielefeld