

# RaGE Pflichtenheft

Jonas Kasper, Bernard Hohmann, Thomas Fischer, Christian Jung, Jonas Linßen

26. Mai 2018

# Inhaltsverzeichnis

<b>1 Zielbestimmung</b>	<b>4</b>
1.1 Musskriterien . . . . .	4
1.2 Kannkriterien . . . . .	4
1.3 Abgrenzungskriterien . . . . .	4
<b>2 Produkteinsatz</b>	<b>5</b>
2.1 Anwendungsbereiche . . . . .	5
2.2 Zielgruppen . . . . .	5
2.3 Betriebsbedingungen . . . . .	5
<b>3 Produktumgebung</b>	<b>5</b>
3.1 Software . . . . .	5
3.2 Hardware . . . . .	5
<b>4 Funktionale Anforderungen</b>	<b>6</b>
/F10/ Zufällige Generierung von Graphen . . . . .	6
/F11/ Voreinstellungen . . . . .	6
/F12/ Einstellungen für Graphen . . . . .	6
/F13/ Einstellungen für Hypergraphen . . . . .	6
/F14/ Generierung von Graphen mit unterschiedlichen Einstellungen . . . . .	7
/F20/ Anwendung von Heuristiken . . . . .	8
/F21/ Liste von Heuristiken . . . . .	8
/F22/* Ergänzung neuer Heuristiken . . . . .	8
/F30/ Modifikation von Graphen . . . . .	9
/F31/ Modifikation der einfachen Graphen . . . . .	9
/F32/ Modifikation der Hypergraphen . . . . .	9
/F33/ Manuelle Erstellung von Graphen und Hypergraphen . . . . .	9
/F40/ Ausgabe über eine graphischer Benutzeroberfläche . . . . .	10
/F41/ Graphgenerierung . . . . .	10
/F42/ Preview . . . . .	10
/F43/ Detailansicht . . . . .	11
/F44/ Grapheditor . . . . .	11
/F50/ Heuristiken für das Total Coloring Conjecture . . . . .	12
/F60/ Heuristiken für das Erdős Faber Lovasz Conjecture . . . . .	12
/F70/ Speichern und Laden . . . . .	13
/F71/ Graphen . . . . .	13
/F72/ Heuristiken . . . . .	13
<b>5 Nichtfunktionale Anforderungen</b>	<b>14</b>
/NF10/ Zuverlässigkeit . . . . .	14
/NF20/ Benutzbarkeit . . . . .	14
/NF30/ Effizienz . . . . .	14
/NF40/ Erweiterbarkeit . . . . .	14
<b>6 Testfälle</b>	<b>15</b>
/T10/ Generierung und Modifikation . . . . .	15
/T20/ Heuristiken . . . . .	15
/T30/ Darstellung von Graphen . . . . .	15
/T40/ Datenkonsistenzen . . . . .	15
<b>7 Produktdaten</b>	<b>16</b>
/D10/ Graphen . . . . .	16
/D20/ Heuristiken . . . . .	16

<b>8</b>	<b>Graphische Benutzeroberfläche</b>	<b>17</b>
8.1	Graphgenerierung . . . . .	17
8.2	Preview . . . . .	18
8.3	Detailansicht . . . . .	21
8.4	Grapheditor . . . . .	22
<b>9</b>	<b>Systemmodelle</b>	<b>23</b>
9.1	Szenarien . . . . .	23
9.2	Anwendungsfälle . . . . .	23
9.3	Durchführbarkeitsanalyse . . . . .	24
9.3.1	Technische Durchführbarkeit . . . . .	24
9.3.2	Personelle Durchführbarkeit . . . . .	24
9.3.3	Alternativen . . . . .	24
9.3.4	Risiken . . . . .	25
<b>10</b>	<b>Glossar</b>	<b>25</b>

# 1 Zielbestimmung

Das Produkt ermöglicht dem Lehrstuhl IPD Böhm die automatische Generierung ungerichteter, einfacher Graphen sowie einfacher Hypergraphen und die automatisierte Evaluation verschiedener Heuristiken zur Lösung von bisher ungelösten Problemen der Informatik.

## 1.1 Musskriterien

- (M1) Es können einfache, ungerichtete Graphen und einfache Hypergraphen nach durch den Nutzer spezifizierten Kriterien zufällig generiert werden.
- (M2) Auf die generierten Graphen können durch den Nutzer auszuwählende Heuristiken angewandt werden.
- (M3) Die Graphen und Ergebnisse der Heuristiken werden in einer graphischen Oberfläche angezeigt.
- (M4) Die Graphen können durch den Nutzer modifiziert werden, d.h. es können Knoten und Kanten hinzugefügt und gelöscht werden. Insbesondere ist es möglich Graphen von Hand zu zeichnen.
- (M5) Für die folgenden offenen Probleme sind Heuristiken implementiert:
  - i) Total Coloring Conjecture
  - ii) Erdős Faber Lovasz Conjecture

## 1.2 Kannkriterien

- (K1) Die folgenden Eigenschaften von Graphen können erkannt werden:
  - i) Zusammenhang
  - ii) Baum
  - iii) Bipartition
  - iv) Planarität
  - v) Intervallgraph / Chordalität

Die zufällige Generierung der Graphen lässt sich insbesondere auf eine Auswahl dieser Eigenschaften einschränken bzw. einige der genannten Eigenschaften können dabei ausgeschlossen werden.
- (K2) Weitere Heuristiken können als Plugins in das Programm eingebunden werden. Dies ermöglicht die Anwendung auf weitere ungelöste Probleme der Informatik.
- (K3) Die folgenden Sprachen werden unterstützt
  - i) Deutsch
  - ii) Englisch

## 1.3 Abgrenzungskriterien

- (A1) Die Heuristiken sind nicht an sich parallelisierbar.
- (A2) Es können keine gerichteten (Multi-)Graphen und keine allgemeinen Hypergraphen generiert werden.
- (A3) Bei der Generierung erfolgt keine Erkennung von Duplikaten. Die Erkennung von Graphenisomorphie ist **nicht** möglich.
- (A4) Das Programm kann weder für das Total Coloring Conjecture, noch für das Erdős Faber Lovasz Conjecture einen Beweis finden.

## **2 Produkteinsatz**

Das Produkt dient der Auswertung von Heuristiken für ungelöste Probleme der Graphentheorie auf möglichst vielen generierten Graphen.

### **2.1 Anwendungsbereiche**

Das Programm ist für die Forschung in der Mathematik und Informatik, genauer der Graphentheorie, vorgesehen.

### **2.2 Zielgruppen**

Das Produkt wird für die Nutzung im Lehrstuhl IPD Böhm am KIT entwickelt. Eine Anwendung in anderen Forschungseinrichtungen, Universitäten und Hochschulen sowie durch Privatpersonen mit Interesse an offenen Problemen der Graphentheorie ist möglich und erwünscht.

### **2.3 Betriebsbedingungen**

Das Programm läuft auf Privat- und Betriebsrechnern (z.B. in einer Büroumgebung).

## **3 Produktumgebung**

### **3.1 Software**

Die folgende Software ist für die Lauffähigkeit des Programms hinreichend:

- Windows 10
- Java 8 Runtime Environment

Durch die Plattformunabhängigkeit der JRE ist auch eine Anwendung auf anderen Betriebssystemen möglich.

### **3.2 Hardware**

Das Programm läuft auf Rechnern mit durchschnittlicher Rechenleistung. Es benötigt mindestens:

- 4GB Arbeitsspeicher
- 4GB Festplattenspeicher
- Maus und ggf. Tastatur

## 4 Funktionale Anforderungen

Die durch \* markierten Einträge sind Kannkriterien und nur aus Vollständigkeitsgründen im Folgenden aufgeführt.

### /F10/ Zufällige Generierung von Graphen

#### /F11/ Voreinstellungen

Die folgenden Voreinstellungen können getroffen werden:

- i) Es kann ausgewählt werden, ob einfache ungerichtete Graphen (SGs) oder einfache Hypergraphen (SH) generiert werden sollen
- ii) Die Anzahl der zu generierenden (Hyper-) Graphen kann eingestellt werden. Es werden zwischen 1 und 100 000 Graphen unterstützt.

#### /F12/ Einstellungen für Graphen

Hat der Nutzer die Generierung von SGs gewählt, so kann er die folgenden Einstellungen vornehmen:

- i) Die Anzahl von Knoten  $n$  der zu generierenden Graphen kann festgelegt werden. Es gilt  $1 \leq n \leq 1000$ .
- ii) Der minimale und maximale Knotengrad  $\delta_{min}$  und  $\Delta_{max}$  der zu generierenden Graphen kann festgelegt werden. Für einen beliebigen generierten Graphen  $G$  gilt dann

$$0 \leq \delta_{min} \leq \delta(G) \leq \Delta(G) \leq \Delta_{max} \leq n - 1.$$

- iii)\* Die folgenden Grapheigenschaften können (falls gewünscht) bei der Generierung ausgeschlossen oder forciert werden:

- Zusammenhang
- Baum
- Bipartition
- Planarität
- Intervallgraph / Chordalität

#### /F13/ Einstellungen für Hypergraphen

Hat der Nutzer die Generierung von SHs gewählt, so kann er die folgenden Einstellungen vornehmen:

- i) Die Anzahl von Knoten  $n$  der zu generierenden Graphen kann festgelegt werden. Es gilt  $1 \leq n \leq 1000$ .
- ii) Der minimale und maximale Knotengrad  $\delta_{min}$  und  $\Delta_{max}$  der zu generierenden Graphen kann festgelegt werden. Für einen beliebigen generierten Graphen  $G$  gilt dann

$$0 \leq \delta_{min} \leq \delta(G) \leq \Delta(G) \leq \Delta_{max} \leq n - 1.$$

- iii)\* Es kann (falls gewünscht) die minimale und maximale Kardinalität  $k_{min}$  und  $k_{max}$  der Hyperkanten festgelegt werden. Für einen beliebigen generierten Hypergraph  $G$  und eine Hyperkante  $e \in E(G)$  gilt dann

$$2 \leq k_{min} \leq \#e \leq k_{max} \leq n.$$

Insbesondere lässt sich die Generierung über  $k_{min} = k_{max} =: k$  auch auf  $k$ -uniforme Hypergraphen beschränken.

#### **/F14/ Generierung von Graphen mit unterschiedlichen Einstellungen**

Der Nutzer hat die Möglichkeit mehrere voneinander unabhängige Datenmengen zu generieren. Dies ermöglicht den direkten Vergleich zwischen Graphen, die mit unterschiedlichen Einstellungen generiert wurden. Zudem ist der Nutzer so in der Lage an mehreren offenen Problemen zeitgleich zu arbeiten.

## **/F20/ Anwendung von Heuristiken**

### **/F21/ Liste von Heuristiken**

Es kann eine Liste von Heuristiken bestimmt werden, die auf jeden der generierten Graphen angewandt werden. Diese obliegt den folgenden Einschränkungen:

- i) Die möglichen Heuristiken sind abhängig von dem generierten Graphentyp.
- ii) Dieselbe Heuristik kann mit unterschiedlichen Einstellungen mehrfach in der Liste vorkommen, nicht jedoch zweimal mit derselben Einstellung.
- iii) Die Bestätigung der Änderungen an der Liste bewirkt eine Neuberechnung aller neuen oder veränderten Heuristiken auf allen generierten Graphen, nicht jedoch die Neuberechnung von bereits angewandten und unverändert gebliebenen Heuristiken.

### **/F22/\* Ergänzung neuer Heuristiken**

Es können neue Heuristiken als Plugins hinzugefügt werden. Die Erkennung der Plugins geschieht zum Programmstart und ist während der Laufzeit nicht mehr möglich.



## /F30/ Modifikation von Graphen

### /F31/ Modifikation der einfachen Graphen

Das Programm bietet die Möglichkeit die Graphen in der folgenden Art und Weise zu modifizieren:

- i) Es können Knoten hinzugefügt werden.
- ii) Kanten zwischen zwei bestehenden und noch nicht adjazenten Knoten können hinzugefügt werden.
- iii) Jeder Knoten  $v$  kann gelöscht werden. Alle zu  $v$  inzidenten Kanten werden automatisch ebenfalls gelöscht.
- iv) Der Nutzer kann Kanten löschen.
- v) Knoten können kontrahiert werden. Auftretende Mehrfachkanten werden automatisch zu einfachen Kanten reduziert. Auftretende Schleifen werden gelöscht.
- vi) Jeder Knoten  $v$  kann dupliziert werden, d.h. es wird ein neuer Knoten  $v$  mit derselben Nachbarschaft wie  $v$  hinzugefügt.
- vii) Die Reihenfolge der Knoten kann verändert werden.

Eine beliebige Auswahl von  $n$  Knoten kann auf die folgenden Graphen ergänzt werden:

- i) **Leerer Graph**  $E_n$ : Alle Kanten zwischen den ausgewählten Knoten werden gelöscht.
- ii) **Pfad**  $P_n$ : In Reihenfolge der Auswahl werden zwischen aufeinanderfolgenden und nichtadjazenten Knoten Kanten hinzugefügt. Der erste und der letzte Knoten werden nicht explizit durch eine Kante verbunden. Wenn sie jedoch bereits adjazent sind, wird die bestehende Kante nicht gelöscht.
- iii) **Kreis**  $C_n$ : Die Knoten werden wie beim Pfad in der Reihenfolge ihrer Auswahl verbunden mit dem Unterschied, dass auch der erste und letzte Knoten explizit verbunden werden.
- iv) **Clique**  $K_n$ : Je zwei Knoten aus der Auswahl, die nicht bereits adjazent sind, werden durch eine neue Kante verbunden.

Es wird sichergestellt, dass der resultierende Graph auch wieder ein einfacher ungerichteter Graph ist.

### /F32/ Modifikation der Hypergraphen

Hypergraphen können wie folgt modifiziert werden:

- i) Es können Knoten hinzugefügt werden
- ii) Durch Auswahl von mehr als einem Knoten kann eine Hyperkante hinzugefügt werden, wenn der Hypergraph keine Hyperkante besitzt, die mehr als einen der ausgewählten Knoten enthält.
- iii) Jeder Knoten kann gelöscht werden. Dabei werden alle Hyperkanten, die diesen Knoten enthalten gelöscht.
- iv) Jede Hyperkante kann gelöscht werden.
- v) Die Reihenfolge der Knoten kann verändert werden.

Es wird sichergestellt, dass der resultierende Hypergraph auch wieder einfach ist.

### /F33/ Manuelle Erstellung von Graphen und Hypergraphen

Die in /F31/ und /F32/ erwähnten Modifikationen implizieren die Möglichkeit Graphen von Hand im Programm einzutragen. Der Typ des Graphen wird durch die Auswahl in /F11/ festgelegt.

## /F40/ Ausgabe über eine graphischer Benutzeroberfläche

Die graphische Benutzeroberfläche gliedert sich im wesentlichen in die folgenden Bereiche:

### /F41/ Graphgenerierung

In diesem Fenster können alle Einstellungen aus /F10/ für die zu generierenden Graphen vorgenommen werden. Nach Bestätigung durch den Nutzer werden die Graphen unter Berücksichtigung dieser Einstellungen generiert.

### /F42/ Preview

Alle generierten Graphen werden in einer Übersicht dargestellt. Es ist ersichtlich, wie weit die Generierung fortgeschritten ist. Graphen, die durch Modifikation (siehe /F30/) auseinander hervorgegangen sind, werden als Gruppe angezeigt.

#### /F42.1/ Auswahl und Anwenden der Heuristiken

Der Nutzer kann Algorithmen auswählen, Einstellungen an ihnen vornehmen und sie in der Liste der auf die Graphen anzuwendenden Heuristiken speichern. Heuristiken können auch wieder aus der Liste entfernt werden.

Wie in /F21/ angegeben führt eine Veränderung der List zu einer Berechnung der neu hinzugekommenen oder veränderten Heuristiken auf allen generierten Graphen. Ferner werden die Heuristiken der Liste automatisch auf jeden neu generierten Graph angewandt. Es ist ersichtlich, ob auf einen Graphen noch nicht alle Heuristiken angewandt wurden bzw. ob im Moment eine Heuristik auf ihm berechnet wird.

#### /F42.2/ Filtern der angezeigten Graphen

Die Reihenfolge der Graphen in der Preview können

- i) nach Reihenfolge der Generierung
- ii) nach Geschwindigkeit einer auszuwählenden Heuristik der Liste
- iii) nach der Gesamtgeschwindigkeit aller Heuristiken der Liste

jeweils aufsteigend oder absteigend sortiert werden. Gruppierte Graphen, die z.B. durch Modifikation auseinander hervorgegangen sind, werden nach dem Durchschnitt des Sortierungsparameters einsortiert.

Es kann für jede Heuristik ausgewählt werden, ob

- i) alle Graphen
- ii) nur die Graphen, für die die Heuristik erfolgreich terminiert ist
- iii) nur die Graphen, für die die Heuristik nicht erfolgreich war

angezeigt werden sollen. Ferner kann die Anzahl der angezeigten Graphen durch den Nutzer beschränkt werden.

Dieser Filterungsprozess hat rein visuelle Auswirkungen und bewirkt **nicht** den Verlust der aussortierten Graphen.

### /F43/ Detailansicht

Die Detailansicht bietet dem Nutzer die Möglichkeit, sich die Arbeitsweise und Ergebnisse der Heuristiken auf einem von ihm ausgewählten Graph genauer anzuschauen. Sowohl Graphen als auch Hypergraphen können in den folgenden Arten präsentiert werden:

- i) In der graphischen Repräsentation sind alle Knoten im Uhrzeigersinn gemäß ihrer Reihenfolge im Kreis angeordnet. Die Kanten sind einfache Liniensegmente von Knoten zu Knoten, während die Hyperkanten als konvexe Polygone mit den zugehörigen Knoten als Eckpunkten dargestellt werden.
- ii) In einer tabellarischen Repräsentation

Der Nutzer kann aus der in /F42/ erstellten Liste eine beliebige Heuristik auswählen, deren Ergebnis dann in geeigneter Weise (z.B. durch Färben der Kanten) angezeigt wird. Ferner besteht die Möglichkeit die Heuristik neu auszuführen, um ihre Arbeitsweise wahlweise live oder schrittweise zu begutachten. Ein Eingriff in die Arbeitsweise der Heuristik ist jedoch **nicht** möglich.

Der Nutzer kann zudem lokal weitere Heuristiken auf den Graph anwenden, die nicht in der globalen Liste aus /F42/ sind. Das ermöglicht die genauere Untersuchung eines einzelnen Graphen.

### /F44/ Grapheditor

Der Grapheditor bietet dem Nutzer alle in /F30/ spezifizierten Möglichkeiten um Graphen zu modifizieren oder zu erstellen.

## /F50/ Heuristiken für das Total Coloring Conjecture

Die folgenden Heuristiken sind implementiert:

- i) **Greedy**: Zunächst werden die Knoten möglichst gleichmäßig koloriert. Anschließend werden Knoten für Knoten alle Kanten koloriert.
- ii) **Greedy + One**: Wie Greedy, jedoch mit der folgenden Einschränkung: Gibt es eine unkolorierte Kante im Graph, die genau eine freie Farbe hat, so wird diese vorrangig koloriert.
- iii) **Greedy + Few**: Wie Greedy mit der Eigenschaft, dass statt Knoten für Knoten stets die Kanten mit der minimalen Anzahl von freien Farben koloriert werden.
- iv) **Greedy + Set**: Wie Greedy mit der folgenden Policy: Es wird unter allen Mengen von Kanten, bei denen alle Kanten inzident zu einem gemeinsamen Knoten sind, eine mit minimaler Flexibilität koloriert.
- v) **Greedy + Connected**: Wie Greedy + Set mit der Verallgemeinerung, dass die Menge von Kanten nicht inzident zu einem gemeinsamen Knoten, sondern lediglich zusammenhängend sein muss. Die maximale Kardinalität einer solchen Teilmenge ist durch den Nutzer einstellbar.
- vi)\* **Mixed Greedy**: Knoten und Kanten sind gleichwertig und werden nicht separat koloriert.
- vii)\* **Line Greedy**: Wie die Greedy Familie, nur Vertauschen der Rolle von Knoten und Kanten.

Die Heuristiken berechnen eine möglichst gleichmäßige Farbverteilung. Dabei ist die Gewichtung des Einflusses von Knoten und Kanten für die freien Farben durch den Nutzer einstellbar.

Es wird abgebrochen, sobald die Bedingungen des Total Coloring Conjectures verletzt sind, oder wenn eine korrekte Totalfärbung berechnet wurde.

Die Heuristiken sind deterministisch. Das bedeutet, dass eine Heuristik mehrfach angewandt auf denselben Graph mit denselben Einstellungen das gleiche Ergebnis liefert.

## /F60/ Heuristiken für das Erdős Faber Lovasz Conjecture

Die folgenden Heuristiken sind implementiert:

- i) **Greedy**: Es werden Knoten für Knoten alle Hyperkanten koloriert.
- ii) **Greedy + One**: Wie Greedy, jedoch mit der folgenden Einschränkung: Gibt es eine unkolorierte Hyperkante im Hypergraph, die genau eine freie Farbe hat, so wird diese vorrangig koloriert.
- iii) **Greedy + Few**: Wie Greedy mit der Eigenschaft, dass statt Knoten für Knoten stets die Hyperkanten mit der minimalen Anzahl von freien Farben koloriert werden.
- iv) **Greedy + Set**: Wie Greedy mit der folgenden Policy: Es wird unter allen Mengen von Hyperkanten, bei denen alle Kanten inzident zu einem gemeinsamen Knoten sind, eine mit minimaler Flexibilität koloriert.
- v) **Greedy + Connected**: Wie Greedy + Set mit der Verallgemeinerung, dass die Menge von Hyperkanten nicht inzident zu einem gemeinsamen Knoten, sondern lediglich zusammenhängend sein muss. Die maximale Kardinalität einer solchen Teilmenge ist durch den Nutzer einstellbar.

Die Heuristiken berechnen eine möglichst gleichmäßige Farbverteilung. Es wird abgebrochen, sobald die Bedingungen des Erdős Faber Lovasz Conjectures verletzt sind, oder wenn eine korrekte Färbung berechnet wurde.

Die Heuristiken sind deterministisch. Das bedeutet, dass eine Heuristik mehrfach angewandt auf denselben Graph mit denselben Einstellungen das gleiche Ergebnis liefert.

## **/F70/ Speichern und Laden**

### **/F71/ Graphen**

Der Nutzer kann eine Auswahl oder alle generierten Graphen speichern. Für jeden Graph werden die Ergebnisse aller darauf berechneten Heuristiken zusammen mit den Einstellungen der Heuristik gespeichert.

Gespeicherte Graphen können wieder geladen werden. Sie werden als von möglicherweise bereits generierten Graphen unabhängige Datenmenge in das Programm eingefügt. Ergebnisse von auf ihnen berechneten Heuristiken werden übernommen. Ist eine Heuristik eigentlich nicht auf den Graph anzuwenden, so wird ihr Ergebnis in der lokalen Liste von Heuristiken des Graphen gespeichert.

### **/F72/ Heuristiken**

Die Liste der eingestellten Heuristiken kann gespeichert werden. Die Heuristiken können auch wieder geladen werden, wenn sie auf den in /F11/ eingestellten Graphentyp anwendbar sind.

## 5 Nichtfunktionale Anforderungen

### /NF10/ Zuverlässigkeit

Eine Benutzung gemäß der funktionalen Anforderungen führt zu keinen Komplikationen. Die Generierung von Graphen erfolgt nach den eingestellten Kriterien. Die implementierten Heuristiken verhalten sich wie in /F50/ und /F60/ spezifiziert.

### /NF20/ Benutzbarkeit

Die Benutzung des Programms erfolgt über die graphische Benutzeroberfläche. Sie ist intuitiv gestaltet, sodass für die Benutzung keine Schulung nötig ist. Die Eingabe von ungültigen Daten wird dem Nutzer mitgeteilt und führt insbesondere nicht zu einem Absturz.

### /NF30/ Effizienz

Die Generierung der Graphen erfolgt abhängig von den gewählten Einstellungen in angemessener Zeit.

Über die Effizienz der Heuristiken selbst wird **keine** Angabe gemacht. Die Laufzeit der Heuristiken wird jedoch durch andere Funktionen des Programms nicht übermäßig beeinflusst. Parallelisierung erlaubt Generierung und Anwendung von Heuristiken gleichzeitig.

### /NF40/ Erweiterbarkeit

Das Programm bietet mit der in /F22/ erwähnten Pluginarchitektur und insgesamt modularen Programmarchitektur die Erweiterung um neue Heuristiken.

## 6 Testfälle

### /T10/ Generierung und Modifikation

- Die Einstellungen bei der Graphengenerierung (z.B. Graphentyp, minimaler + maximaler Knotengrad etc.) werden bei der Generierung berücksichtigt
- Nach Bestätigung durch den User werden die Graphen generiert
- Graphen werden vollständig importiert
- Abbrechen der Generierung
- Eingabe des Graphen per Hand

### /T20/ Heuristiken

- Heuristiken in der Liste entsprechen den durch den Nutzer ausgewählten Heuristiken
- Anwendung der Heuristiken auf die Graphen gemäß Nutzerauswahl
- Fehlverhalten der Heuristik

### /T30/ Darstellung von Graphen

- Filtern und Sortieren der Graphen nach durch den Nutzer gegebenen Kriterien
- Ausgabe der Graphen in gewünschter Darstellungsform
- Darstellung eines Graphen in jeweiliger Darstellungsform ist eindeutig
- Modifikation von Graphen führt zu Gruppierung

### /T40/ Datenkonsistenzen

- Für einfache ungerichtete Graphen gilt:

$$\#V(G) \cdot \delta_{min} \leq \#V(G) \cdot \delta(G) \leq \#E(G) \cdot 2 \leq \#V(G) \cdot \Delta(G) \leq \#V(G) \cdot \Delta_{max}$$

- Falls  $\delta_{min} = \Delta_{max} = n - 1$ , so wird nur genau ein Graph generiert
- Generierte Graphen sind entweder einfache ungerichtete Graphen oder einfache Hypergraphen
- Die erhobenen Statistiken sind korrekt

## 7 Produktdaten

### /D10/ Graphen

Für die Graphenstruktur werden die folgenden Daten gespeichert:

- Anzahl der Knoten
- Liste der (Hyper-) Kanten
- Minimale, maximaler und durchschnittlicher Knotengrad
- Die in /F12/\* erkannten Grapheigenschaften

Für jede auf den Graph angewandte Heuristik werden die folgenden Daten gespeichert:

- Name der Heuristik
- Einstellungen der Heuristik
- Ergebnis der Heuristik, d.h. z.B. die Färbung zur Zeit des Abbruchs
- Laufzeit der Heuristik

### /D20/ Heuristiken

Für Heuristiken werden der Name sowie die Einstellungen gespeichert.



## 8 Graphische Benutzeroberfläche

Es folgen frühe Vorschläge für die Benutzeroberfläche. Alle angegebenen Werte sind exemplarisch, insbesondere wurden dargestellte Graphen nicht generiert.

### 8.1 Graphgenerierung

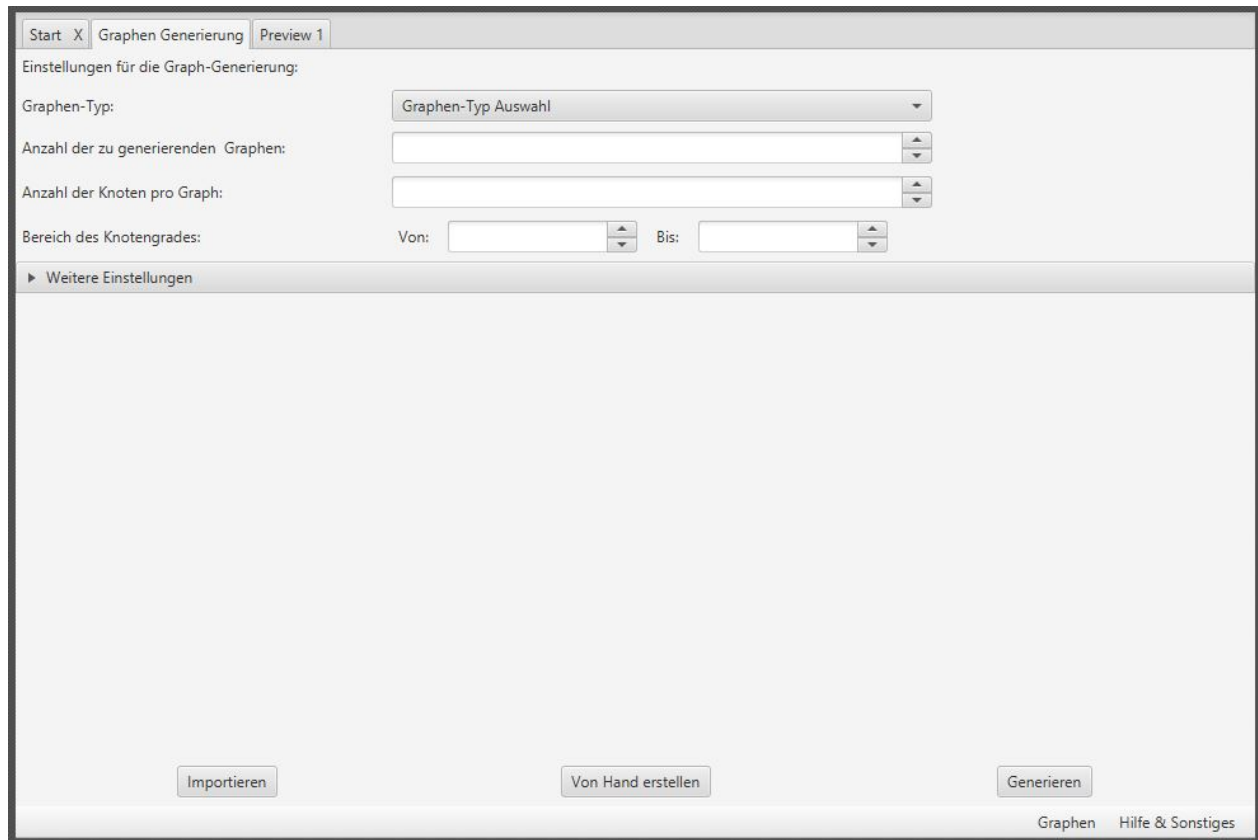


Abbildung 1: Das Fenster für die Graphgenerierung

- (1) Auswahl des zu generierenden Graphentyps
- (2) Allgemeine Einstellungen
- (3) Weitere Einstellungen wie in /F12/iii)\* oder /F13/iii)\* erwähnt
- (4) Laden von bereits generierten Graphen
- (5) Öffnet den Grapheneditor um Graphen per Hand einzugeben
- (6) Generiert Graphen gemäß den getroffenen Einstellungen

## 8.2 Preview

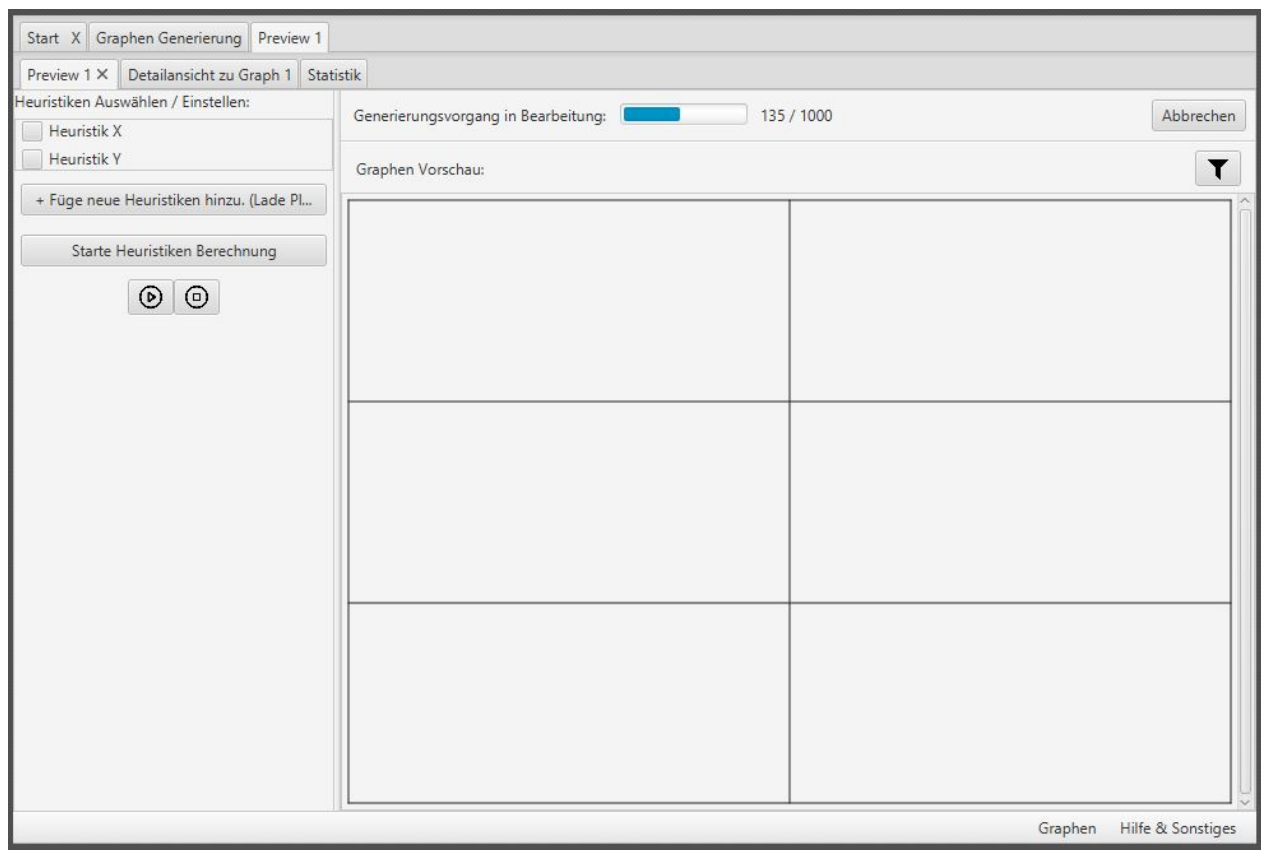


Abbildung 2: Das Preview-Fenster

- (1) Liste der Heuristiken, die auf die generierten Graphen angewandt werden
- (2) Ermöglicht das Hinzufügen von Heuristiken zu der Liste
- (3) Bestätigt Liste und führt alle neu hinzugekommenen Heuristiken aus
- (4) Liste aller bereits generierten Graphen
- (5) Filter- und Darstellungsoptionen für die Liste der Graphen

The image shows a software dialog box with two main sections. The left section, titled "Heuristiken:", contains two radio buttons labeled "Heuristik X" and "Heuristik Y". The right section, titled "Einstellungen:", contains four settings: "Anzahl der Farben:", "Knotengewicht:", "Kantengewicht:", and "Zuerst zu Färben:", each with a corresponding input field or dropdown menu. Below these settings is a button labeled "► Weitere Einstellungen". At the bottom of the dialog are three buttons: "Zur Liste Hinzufügen", "Abbrechen", and "OK".

Abbildung 3: Hinzufügen neuer Heuristiken

- (1) Mögliche Heuristiken, die zur Auswahl stehen
- (2) Heuristikspezifische Einstellungen
- (3) Fügt Heuristik mit getroffenen Einstellungen zur Liste hinzu
- (4) Verwirft die getroffenen Änderungen und kehrt zurück zur Preview
- (5) Sichert die getroffenen Änderungen und kehrt zurück zur Preview

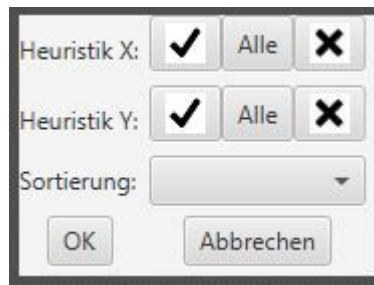


Abbildung 4: Filteroptionen

- (1) Auswahlmöglichkeit pro Heuristik
- (2) Sortierungsoption

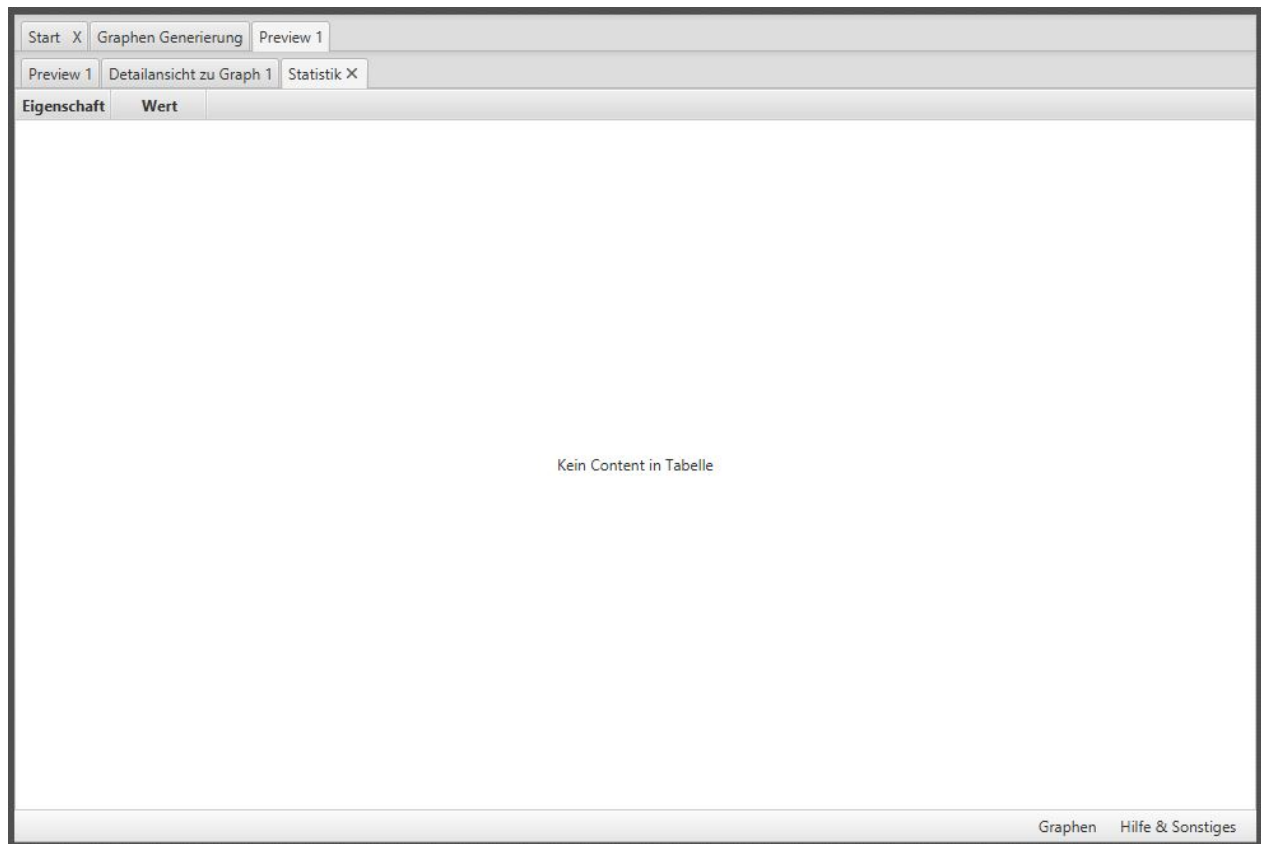


Abbildung 5: Statistiken zur Laufzeit und Erfolgsrate der Statistiken

### 8.3 Detailansicht

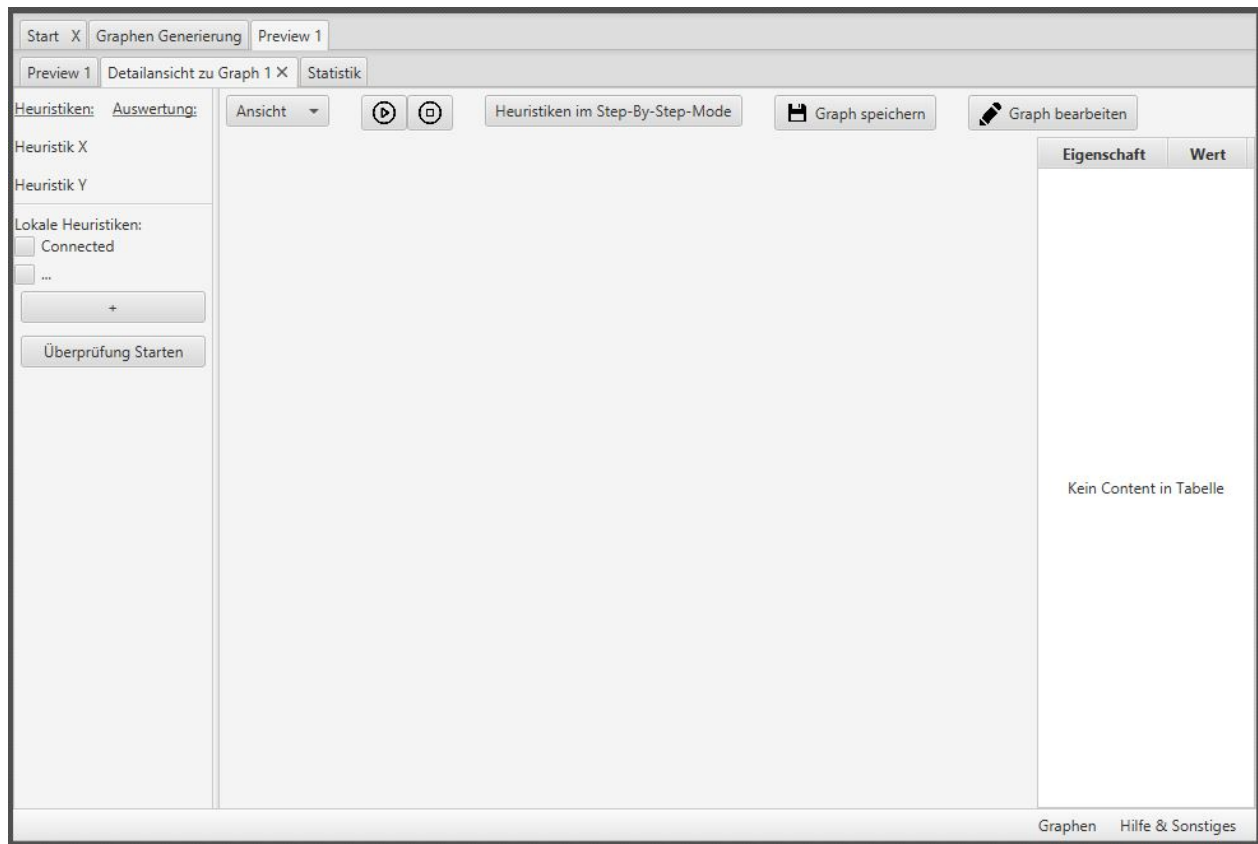


Abbildung 6: Die Detailansicht für die Graphen

- (1) Liste der Heuristiken, die auf alle Graphen der Preview angewendet wurden
- (2) Lokale Liste von Heuristiken, die nur auf den ausgewählten Graph angewandt werden
- (3) Auswahl der Darstellung (graphisch oder tabellarisch)
- (4) Ermöglicht Beobachtung der Arbeitsweise der ausgewählten Heuristik
- (5) Speichern des Graphen
- (6) Öffnet den Grapheditor zur Modifikation des Graphen
- (7) Informationen über den Graph (z.B.  $\delta(G)$ ,  $\Delta(G)$ ,  $\#E(G)$ )
- (8) Darstellung des Graphen mit Ergebnis der ausgewählten Heuristik

## 8.4 Grapheditor

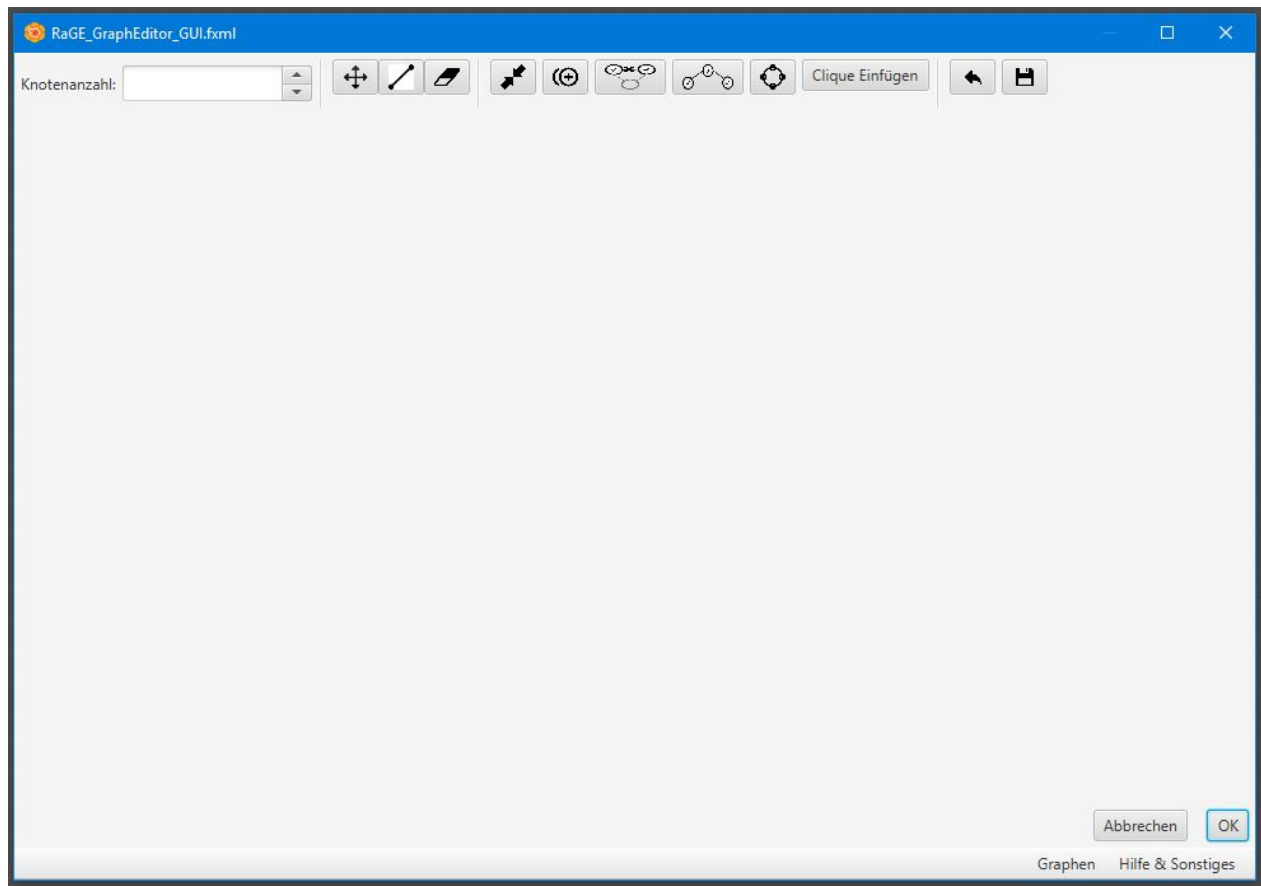


Abbildung 7: Die Detailansicht für die Graphen

- (1) Optionen zur Modifikation des Graphen
- (2) Letzte Änderung rückgängig machen
- (3) Verwerfen / Bestätigen der Modifikation

## 9 Systemmodelle

### 9.1 Szenarien

### 9.2 Anwendungsfälle

## 9.3 Durchführbarkeitsanalyse

### 9.3.1 Technische Durchführbarkeit

#### Hardware

Jedes Teammitglied besitzt einen modernen Rechner, der insbesondere den Kriterien der Produktumgebung entspricht.

#### Software

Die erwähnten Rechner haben hauptsächlich Windows 10 als Betriebssystem. Weitere Rechner mit MacOS und Linux sind vorhanden um Plattformunabhängigkeit zu gewährleisten.

Die Entwicklung des Programms erfolgt in Java. Zu diesem Zweck werden geeignete IDEs (z.B. NetBeans, IntelliJ, Eclipse) verwendet. Die graphischen Benutzeroberflächen werden mit Screenbuilder2 erstellt. Das Programm wird mithilfe von JUnit getestet, Versionskontrolle erfolgt mithilfe von Git.

Dieses und folgende Spezifikationsdokumente werden mit LaTeX erstellt, Folien für Präsentationen hingegen mit PowerPoint, Keynote oder LaTeX.

#### Orgware

Zur Organisation des Teams kommen GitHub, Trello und WhatsApp zum Einsatz.

### 9.3.2 Personelle Durchführbarkeit

Das Team erfüllt alle Voraussetzungen des Moduls Praxis der Softwareentwicklung, das den Rahmen für die Entwicklung dieses Projektes bildet. Es wird von fünf Informatikstudenten im dritten oder höherem Semester entwickelt. Damit verbunden sind solide fachliche Kompetenzen im Bereich der Softwareentwicklung. Die oben genannte Entwicklungsumgebung ist allen Teammitgliedern vertraut.

Die Entwicklung erfolgt nach dem Wasserfallmodell und umfasst einen zeitlichen Aufwand von etwa 270 Stunden pro Teammitglied. Diese Zeit ist ausreichend für die Fertigstellung des Programms innerhalb eines Semesters.

### 9.3.3 Alternativen

Es ist möglich eine ausreichend große Anzahl von Graphen per Hand zu generieren und auf ihnen verschiedene Heuristiken nachzurechnen. Diese Option ist jedoch aufgrund ihrer zeitlichen Aufwändigkeit und Fehleranfälligkeit nicht praktikabel.

Eine weitere Option ist die selbstständige Erweiterung von bestehender Software zur Generierung und Kolorierung von Graphen. Die folgende Software ist frei im Internet verfügbar:

- <https://github.com/CSCsw/ColPack>
- <http://graphstream-project.org/>



### 9.3.4 Risiken

Neben den üblichen Problemen

- Fehlkalkulation des Zeitaufwands
- Ausfall von Teammitgliedern (z.B. durch Krankheit)
- Missverständnisse mit dem Auftraggeber
- Technische Schwierigkeiten (z.B. Verlust von Daten)

die zu einer Verzögerung der Fertigstellung des Projektes führen können, gibt es keine nennenswerten Risiken.

## 10 Glossar

TODO