Random Graph Evaluation

Pflichtenheft

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

Inhaltsverzeichnis

1 Zielbestimmung **3**
1.1 Musskriterien . . . 3
1.2 Kannkriterien . . . 3
1.3 Abgrenzungskriterien . . . 3

2 Produkteinsatz **4**
2.1 Anwendungsbereiche . . . 4
2.2 Zielgruppen . . . 4
2.3 Betriebsbedingungen . . . 4

3 Produktumgebung **4**
3.1 Software . . . 4
3.2 Hardware . . . 4

4 Funktionale Anforderungen **5**
/F10/ Zufällige Generierung von Graphen . . . 5
/F11/ Voreinstellungen . . . 5
/F12/ Einstellungen für Graphen . . . 5
/F13/ Einstellungen für Hypergraphen . . . 5
/F14/ Generierung von Graphen mit unterschiedlichen Einstellungen . . . 6
/F20/ Anwendung von Heuristiken . . . 7
/F21/ Liste von Heuristiken . . . 7
/F22/* Ergänzung neuer Heurisitiken . . . 7
/F30/ Modifikation von Graphen . . . 8
/F31/ Modifikation der einfachen Graphen . . . 8
/F32/ Modifikation der Hypergraphen . . . 8
/F33/ Manuelle Erstellung von Graphen und Hypergraphen . . . 8
/F40/ Ausgabe über eine graphischer Benutzeroberfläche . . . 9
/F41/ Graphgenerierung . . . 9
/F42/ Preview . . . 9
/F43/ Detailansicht . . . 10
/F44/ Grapheditor . . . 10
/F50/ Heuristiken für das Total Coloring Conjecture . . . 11
/F60/ Heuristiken für das Erdös Faber Lovasz Conjecture . . . 11
/F70/ Speichern und Laden . . . 12
/F71/ Graphen . . . 12
/F72/ Heuristiken . . . 12

5 Nichtfunktionale Anforderungen **13**
/NF10/ Zuverlässigkeit . . . 13
/NF20/ Benutzbarkeit . . . 13
/NF30/ Effizienz . . . 13
/NF40/ Erweiterbarkeit . . . 13

6 Testfälle **14**
/T10/ Generierung und Modifikation . . . 14
/T20/ Heuristiken . . . 14
/T30/ Darstellung von Graphen . . . 14
/T40/ Datenkonsistenzen . . . 14

7 Produktdaten **15**
/D10/ Graphen . . . 15
/D20/ Heuristiken . . . 15

8 Graphische Benutzeroberfläche **16**
8.1 Graphgenerierung . . . 16
8.2 Preview . . . 17
8.3 Detailansicht . . . 20
8.4 Grapheditor . . . 21

9 Systemmodelle **22**
9.1 Szenarien . . . 22
9.1.1 Generierung von Graphen . . . 22
9.1.2 Graphen von Hand erstellen . . . 23
9.1.3 Importieren von Graphen . . . 24
9.1.4 Detailansicht . . . 25
9.1.5 Graphen von Hand bearbeiten . . . 25
9.1.6 Allgemeiner Workflow . . . 27
9.1.7 Heuristiken auf Graphen anwenden . . . 28
9.1.8 Filtern der Graphen . . . 29
9.1.9 Statistiken anzeigen . . . 30
9.1.10 Graphen speichern . . . 30
9.1.11 Heuristiken speichern . . . 30
9.1.12 Heuristiken laden . . . 31
9.1.13 * Einfügen eines Heuristik-Plugins . . . 31
9.2 Anwendungsfälle . . . 32
9.3 Durchführbarkeitsanalyse . . . 33
9.3.1 Technische Durchführbarkeit . . . 33
9.3.2 Personelle Durchführbarkeit . . . 33
9.3.3 Alternativen . . . 33
9.3.4 Risiken . . . 34

10 Glossar **35**
10.1 Mathematische Definitionen . . . 35
10.2 Terminologie . . . 36
10.3 Akronyme . . . 36

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) Erdös 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 Erdös 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 einfachen ungerichteten Graphen 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 δ_{min} und Δ_{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

/F13/ Einstellungen für Hypergraphen

Hat der Nutzer die Generierung von einfachen Hypergraphen 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 δ_{min} und Δ_{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 Heurisitk 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 Heurisitiken

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/ spezifierten 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 Erdös 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 Erdös 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

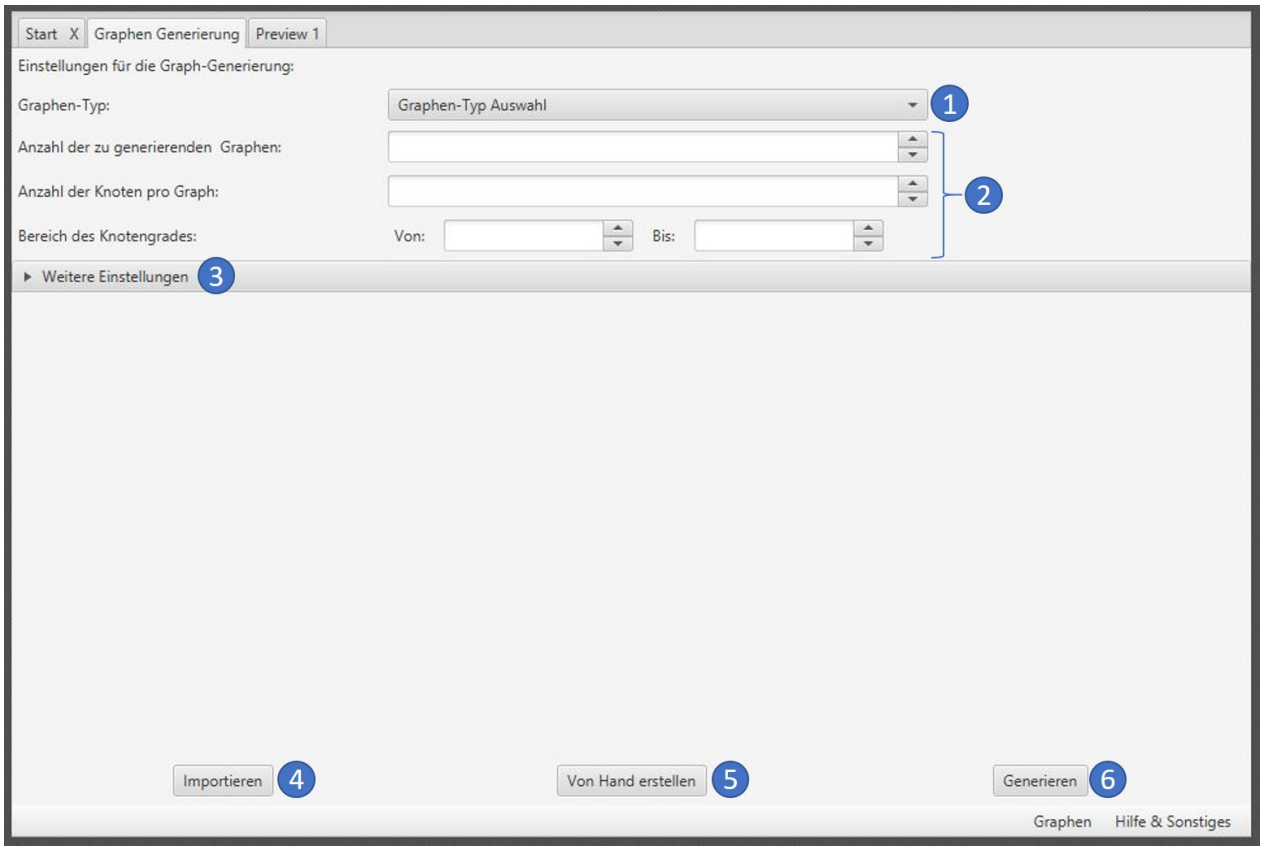


Abbildung 1: Das Fenster für die Graphgenerierung

(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

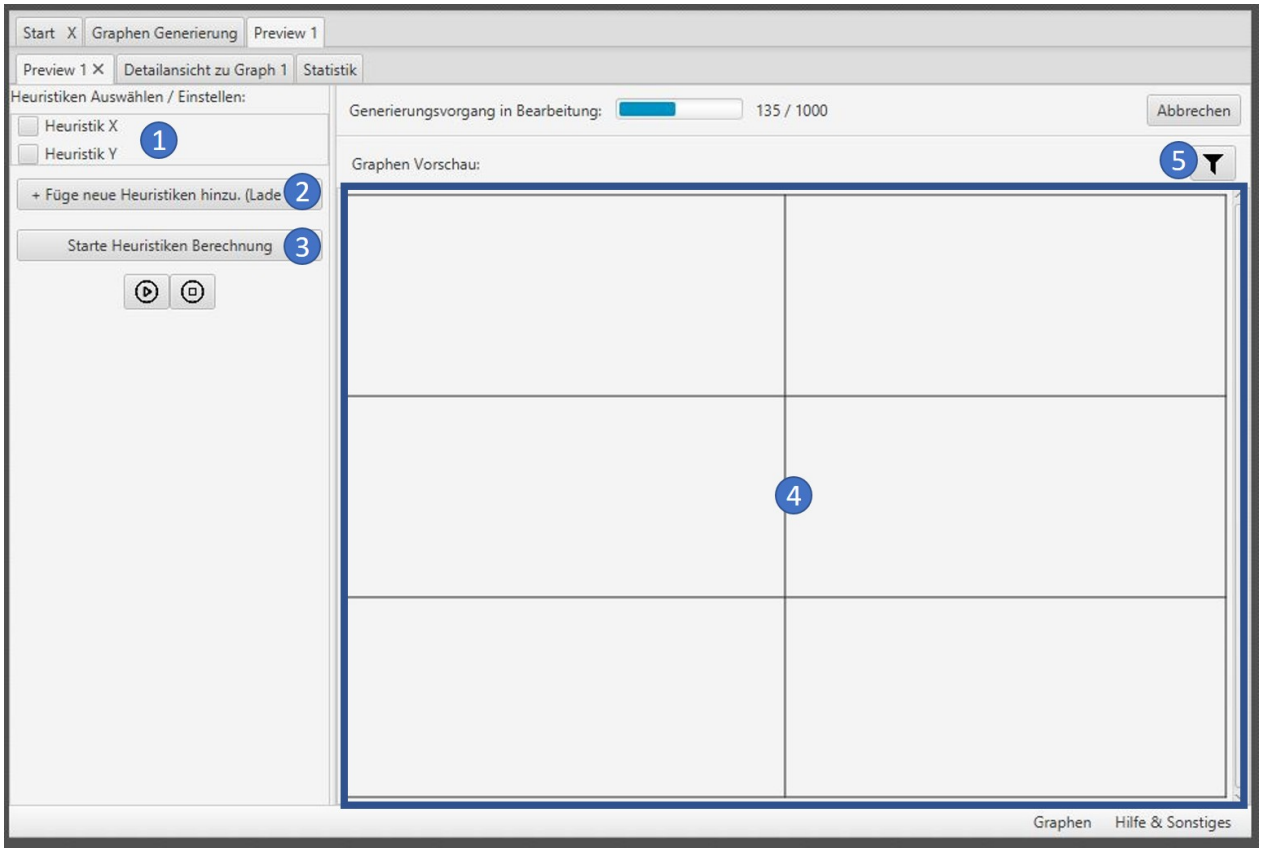


Abbildung 2: Das Preview-Fenster

(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

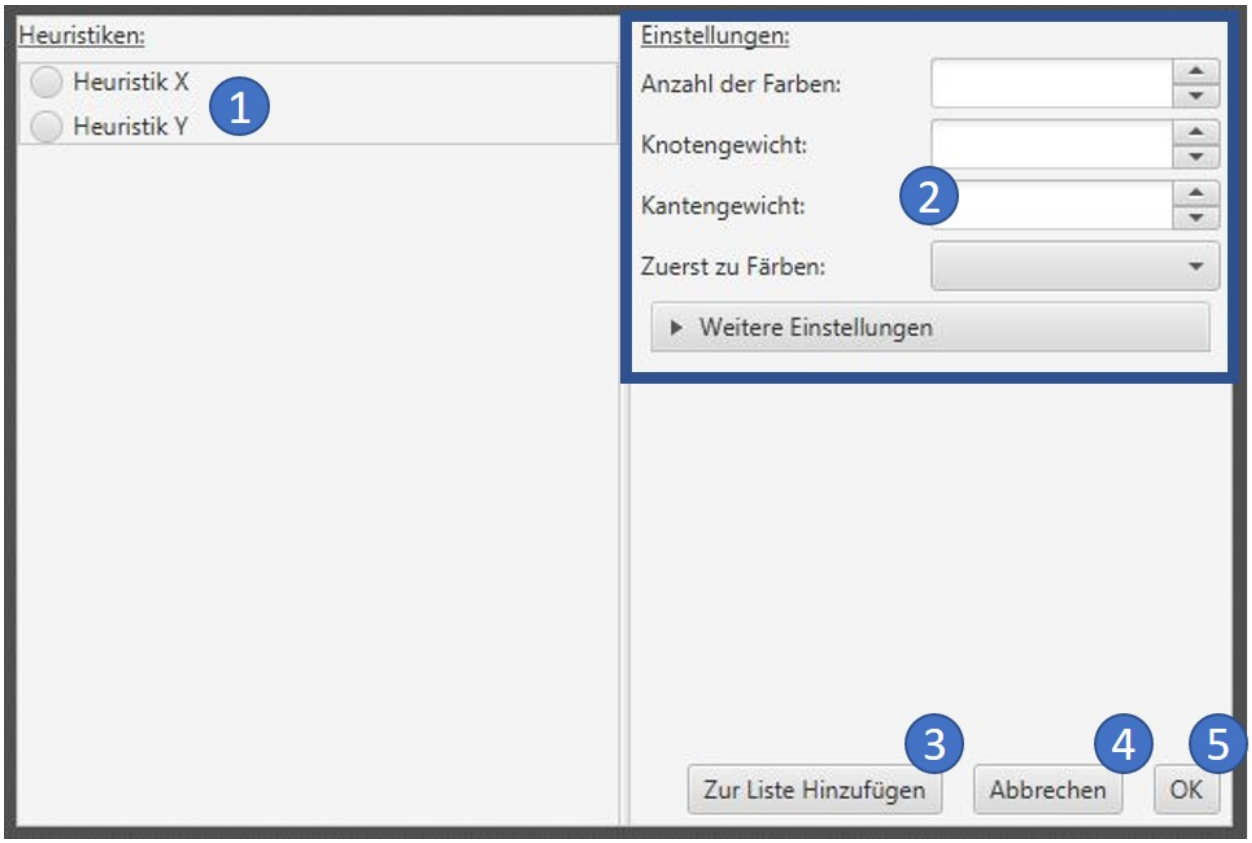


Abbildung 3: Hinzufügen neuer Heuristiken

(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

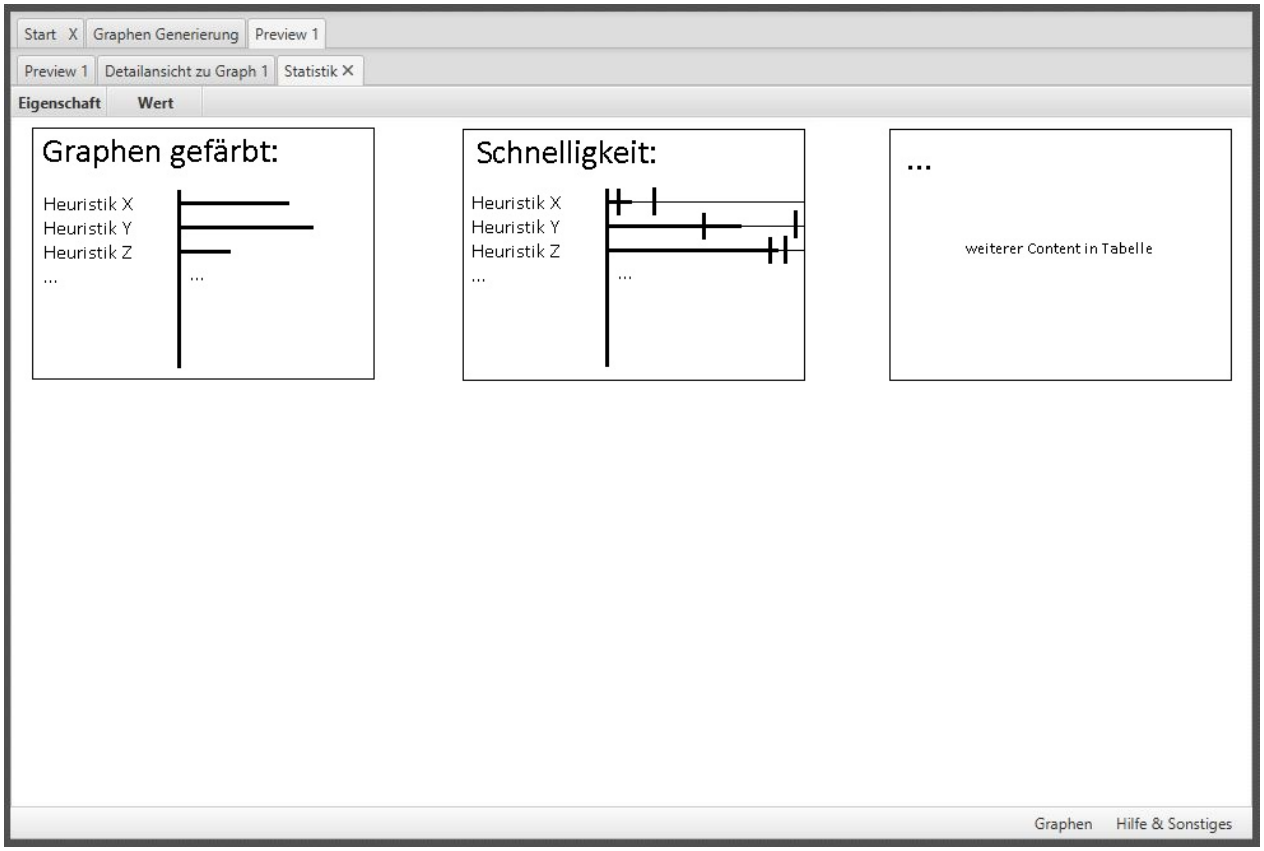


Abbildung 5: Statistiken zur Laufzeit und Erfolgsrate der Statistiken

8.3 Detailansicht

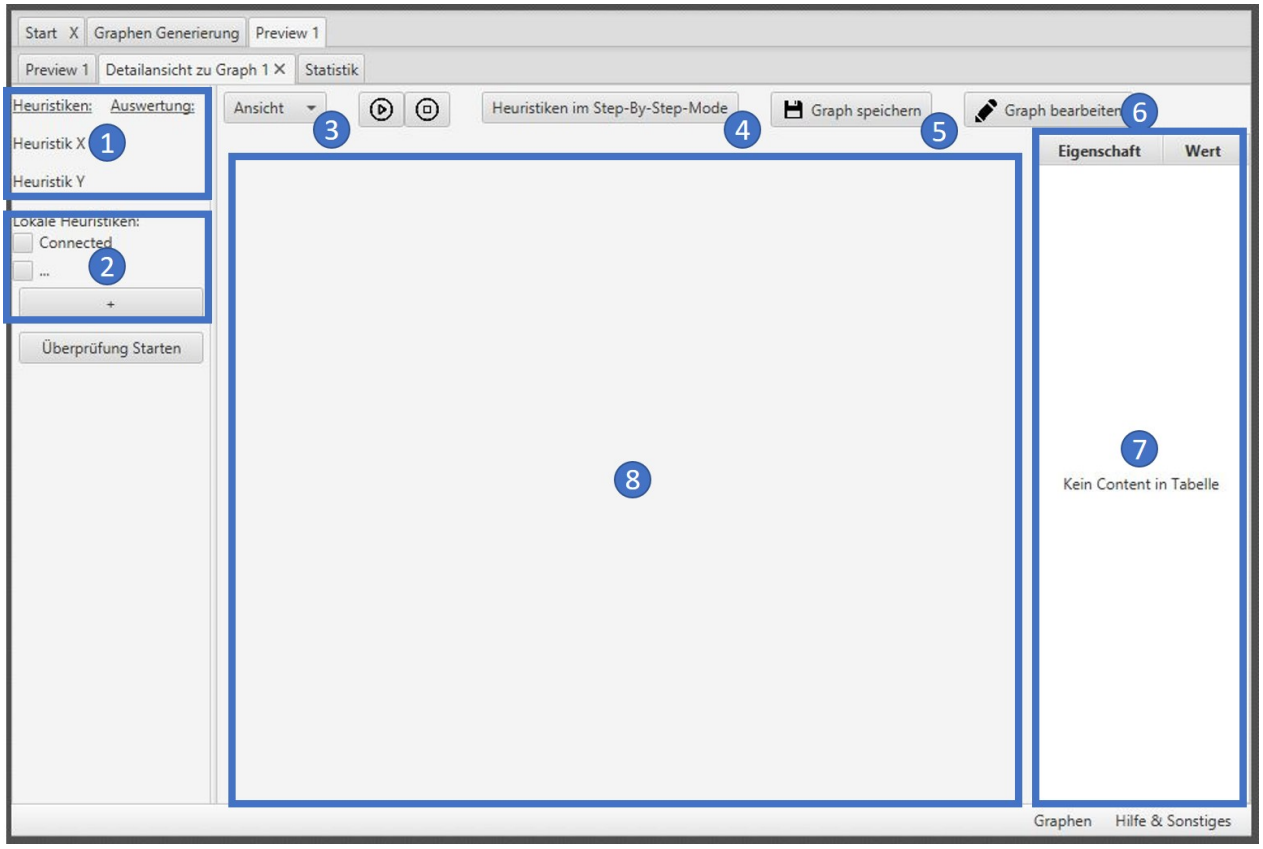


Abbildung 6: Die Detailansicht für die Graphen

(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

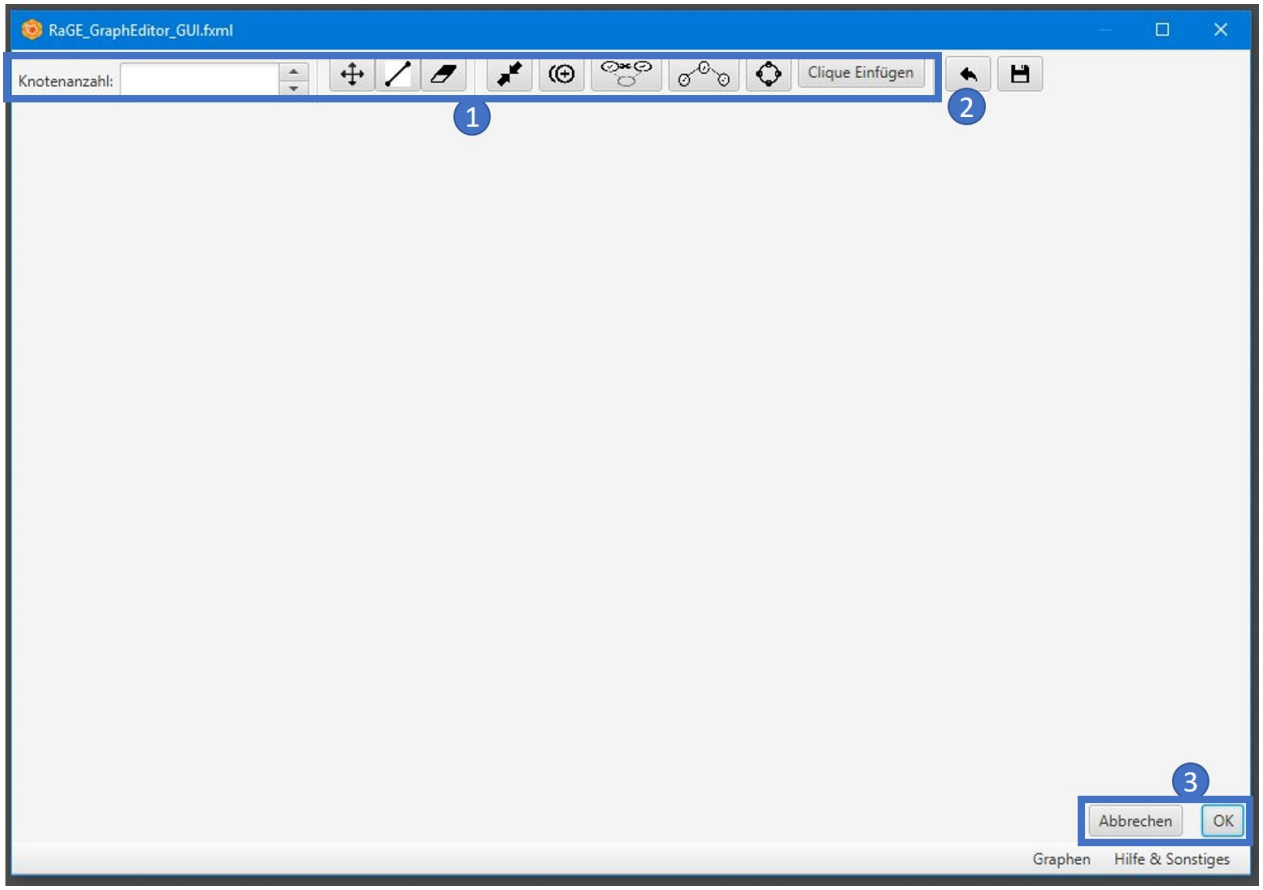


Abbildung 7: Die Detailansicht für die Graphen

(1) Optionen zur Modifikation des Graphen

(2) Letzte Änderung rückgängig machen

(3) Verwerfen / Bestätigen der Modifikation

9 Systemmodelle

9.1 Szenarien

9.1.1 Generierung von Graphen

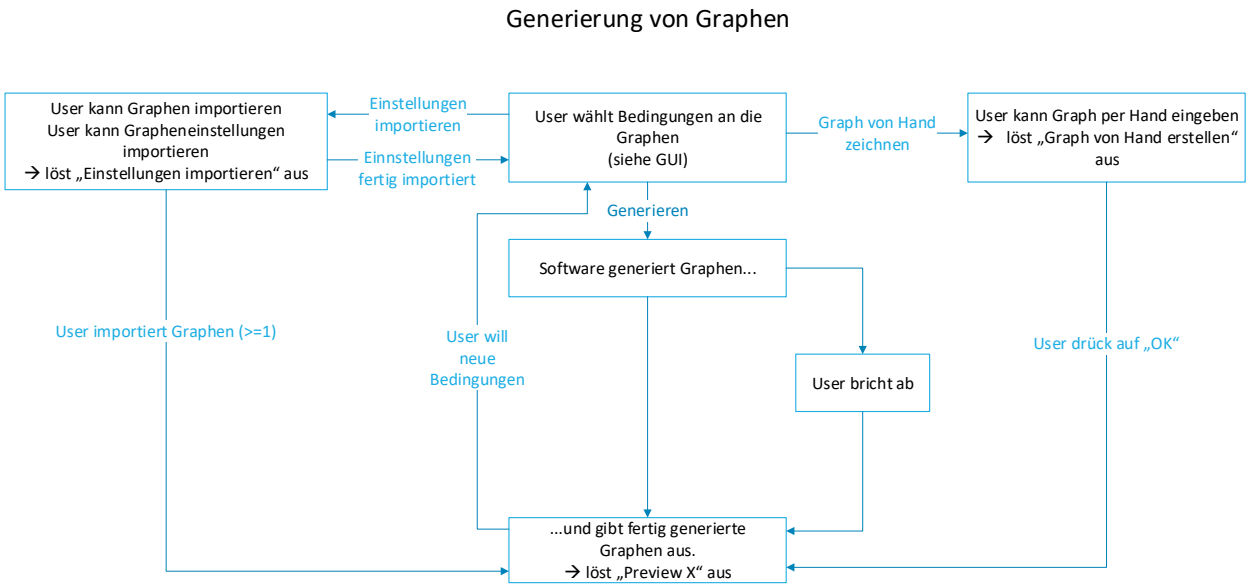
Vorbedingung: keine

Nachbedingung Es wurden Graphen mit durch den Nutzer spezifizierten Eigenschaften generiert

Ablauf:

1) Der Nutzer wählt den gewünschten Graphentyp und die gewünschten Grapheigenschaften aus und bestätigt seine Auswahl.
2) Das Programm generiert anhand der Eigenschaften zufällige Graphen und gibt sie in der Preview graphisch aus. Es können jederzeit neue Graphen generiert werden. Diese werden dann in einer seperaten Preview ausgegeben.

Schema:



9.1.2 Graphen von Hand erstellen

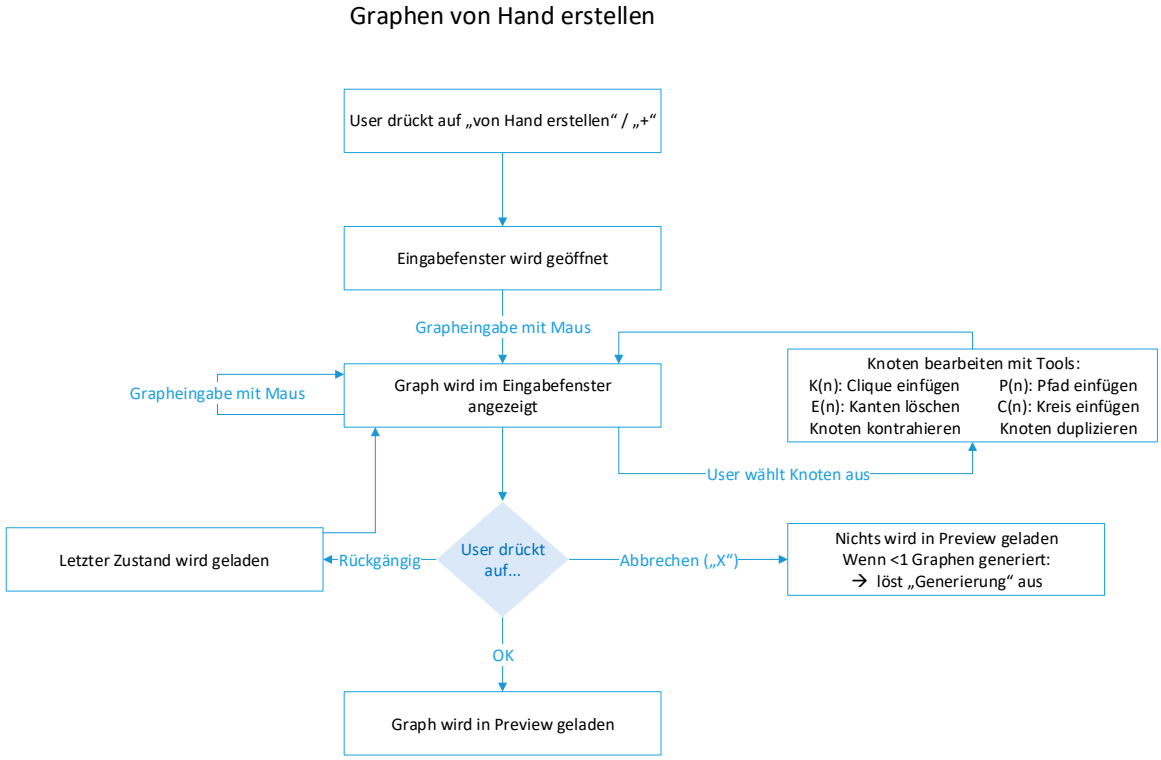
Vorbedingung: keine

Nachbedingung Der erstellte Graph wird in Preview geladen.

Ablauf:

1) Der Nutzer wählt die »Von Hand erstellen« Option aus. Der Grapheditor wird geöffnet.
2) Es kann mithilfe von Tools ein Graph erstellt werden.
3) Sobald der Nutzer »OK« betätigt, wird der Graph hinten an die Graphenliste in der Preview geladen.

Schema:



9.1.3 Importieren von Graphen

Vorbedingung: Es wird eine korrekt kodierte Datei ausgelesen

Nachbedingung (Erfolg): Der Graph / die Graphen werden aus der Datei gelesen und in die Liste der Graphen aufgenommen.

Nachbedingung (Misserfolg): Es erscheint eine aussagekräftige Fehlermeldung.

Ablauf:

1) Der Nutzer wählt den Pfad zu der Datei aus.
2) Das Programm liest die Datei aus.
3) a) Bei Erfolg werden die Graphen in die Liste der Graphen aufgenommen.
   b) Bei Misserfolg wird der Grund des Fehlers angezeigt, falls er ermittelt werden kann.

9.1.4 Detailansicht

Vorbedingung: Es muss einen generierten Graph geben.

Nachbedingung Keine

Ablauf: In der Detailansicht hat der Nutzer die folgenden Möglichkeiten:

1) Lokale Heuristiken hinzufügen
2) Graphen auf der Festplatte speichern
3) Ansicht ändern: Graphisch / Tabellarisch
4) Eine Heuristik im Step-by-Step-Modus ablaufen lassen
5) Einen neuen Graphen aus dem alten erstellen. Nach Auswahl von »Graphen von Hand bearbeiten« öffnet sich der Grapheditor.
6) Zwischen den Graphen in der Preview wechseln, die in der Detailansicht angezeigt werden sollen
7) Informationen über Eigenschaften des Graphen einlesen

Schema: Siehe Schema zu »Graphen von Hand bearbeiten«

9.1.5 Graphen von Hand bearbeiten

Vorbedingung: Es gibt einen Graph.

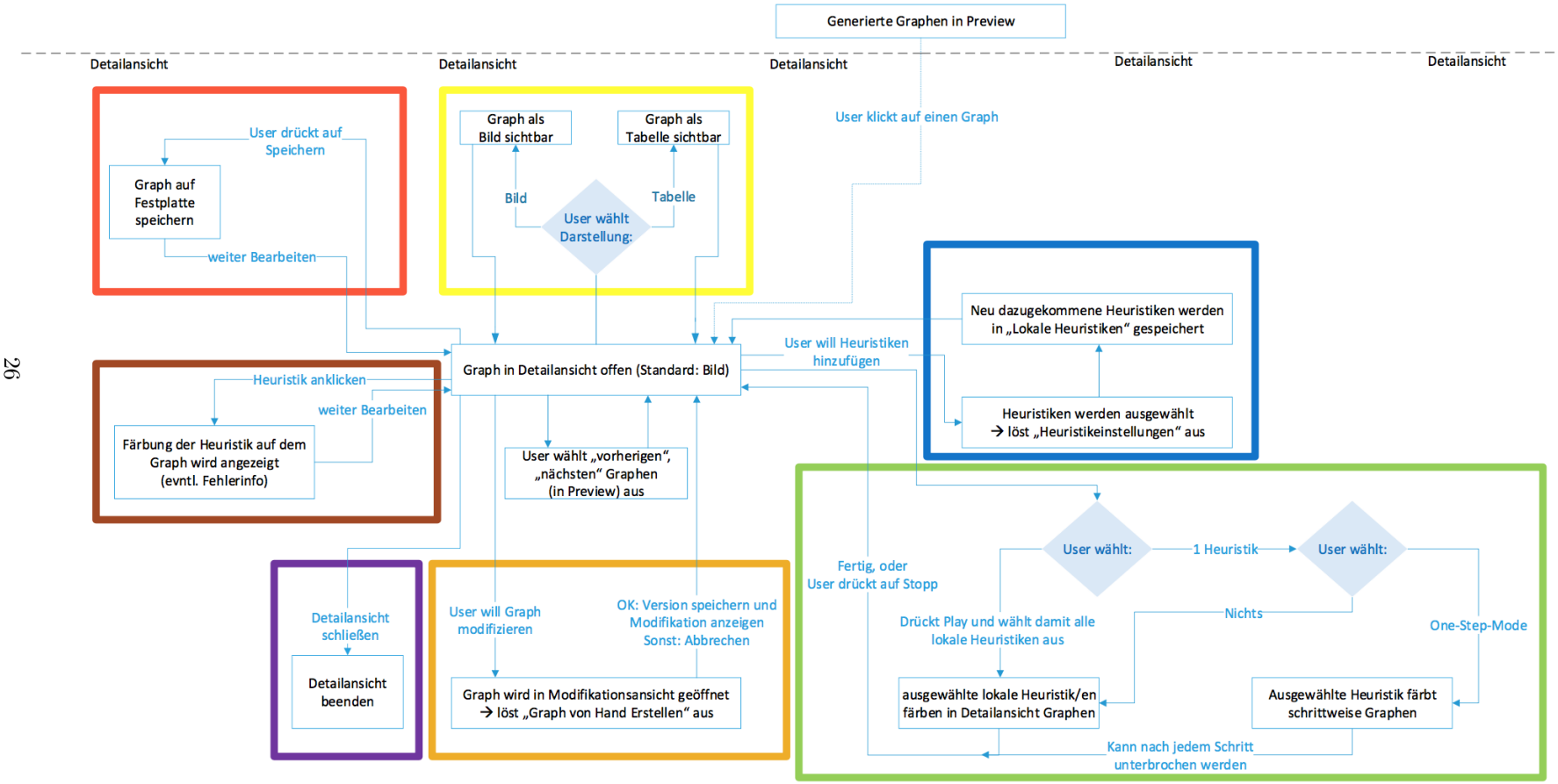
Nachbedingung: Falls Änderungen am Graphen vorgenommen wurden, so wird die geänderte Version direkt neben dem ursprünglichen Graphen in der Preview gespeichert und die Zugehörigkeit mit einer Schraffierung gekennzeichnet.

Ablauf:

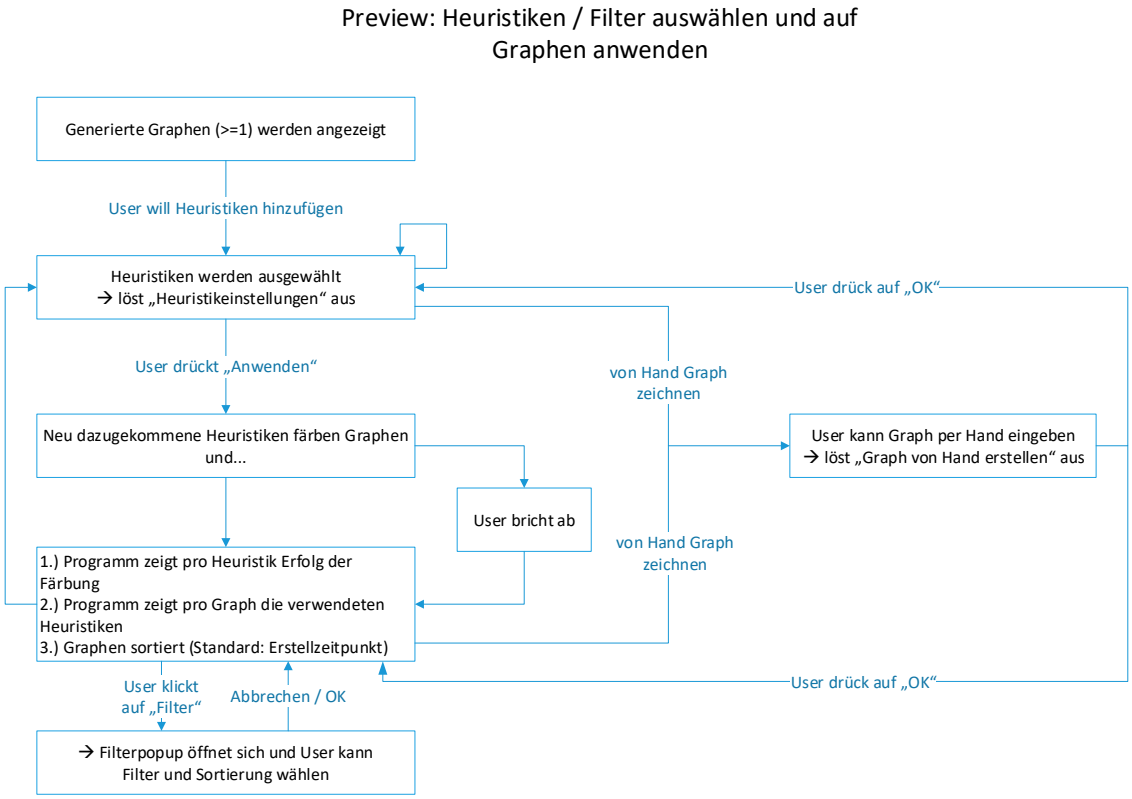
1) Der Nutzer öffnet einen Graphen in der Detailansicht. Er drückt auf »Bearbeiten«.
2) Graph wird im Editor geöffnet
3) Der Nutzer kann mithilfe von Tools den Graphen bearbeiten
4) Sobald der Nutzer auf »OK« drückt, wird der modifizierte Graph in die Detailansicht geladen

Schema: siehe nächste Seite

Detailansicht und Modifikation



9.1.6 Allgemeiner Workflow



9.1.7 Heuristiken auf Graphen anwenden

Vorbedingung: Es gibt mindestens einen generierten oder geladenen Graph, auf den die Heuristik noch nicht angewandt wurde.

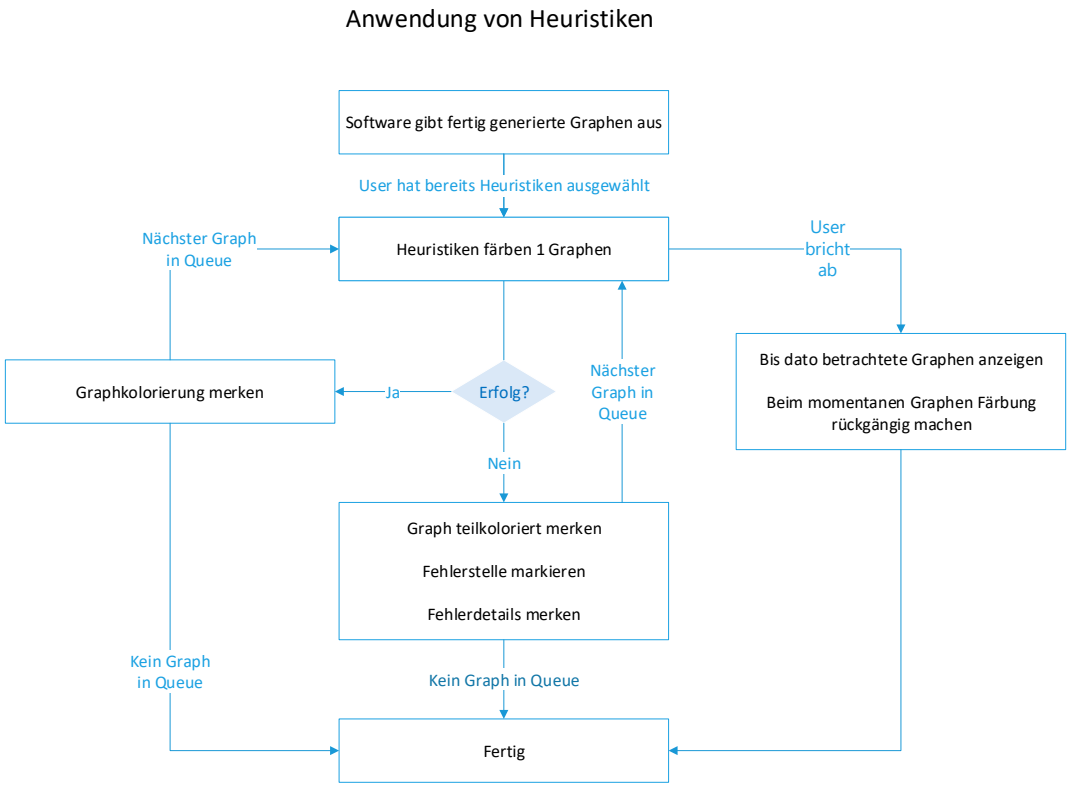
Nachbedingung (Erfolg): Heuristik wurde auf den Graph angewandt. Das Resultat war positiv.

Nachbedingung (Misserfolg): Heuristik wurde auf den Graph angewandt. Das Resultat war negativ.

Ablauf:

1) Der Nutzer wählt die anzuwendenden Heuristiken aus.
2) Das Programm färbt jeden Graph mit jeder Heuristik
3) a) Bei Erfolg speichert das Programm, dass die Heuristik auf diesem Graph erfolgreich war.
   b) Bei Misserfolg speichert das Programm nicht nur, dass die Heuristik auf dem Graph nicht erfolgreich war, sondern auch die Fehlerdetails und den Zustand beim Abbruch.
4) Das Programm sucht den nächsten Graph, der die Vorbedingung erfüllt oder der Nutzer bricht die Anwendung der Heuristiken ab.

Schema:



9.1.8 Filtern der Graphen

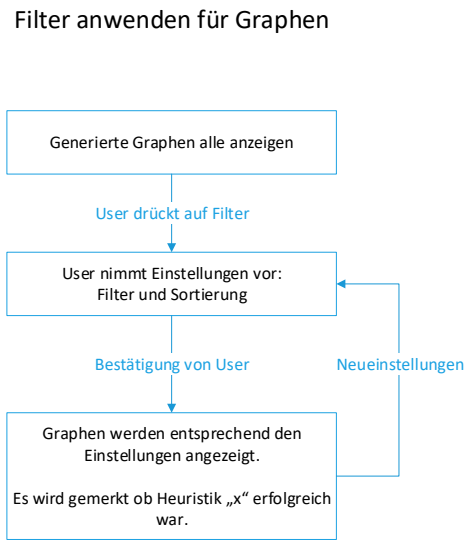
Vorbedingung: Es gibt mindestens einen Graph, auf den mindestens eine Heuristik angewandt wurde.

Nachbedingung: Die Graphen werden gemäß dem eingestellten Filter sortiert angezeigt.

Ablauf:

1) Der Nutzer wählt Filter und Sortierung.
2) Das Programm filtert die Graphen, auf denen die Heuristiken berechnet wurden.
3) Die Graphen werden sortiert.
4) Die bereits gefilterten und sortierten Graphen werden zuerst ausgegeben, die nicht filter-/sortierbaren Graphen werden anschließend angezeigt.
5) Hat eine Heuristik auf einem noch nicht gefilterten/sortierten Graph terminiert, wird dieser falls möglich in die sortierten Graphen eingefügt.

Schema:



9.1.9 Statistiken anzeigen

Vorbedingung: Es gibt mindestens einen Graph, auf den mindestens eine Heuristik angewandt wurde.

Nachbedingung: Es werden verschiedene Statistiken angezeigt, die Informationen über Heuristiken und Graphen beinhalten.

Ablauf:

1) Während der Anwendung der Heurstiken sammelt das Programm Daten über die Graphen, sowie Effizienz und Erfolg der Heuristiken.
2) Durch Nutzerinteraktion wird das Statistik-Fenster geöffnet.
3) Das Programm liest die gesammelten Daten aus und stellt sie graphisch dar.

9.1.10 Graphen speichern

Vorbedingung: Es gibt mindestens einen Graph in der Preview.

Nachbedingung: Graphen werden korrekt in einer Datei auf der Festplatte gespeichert.

Ablauf:

1) Der Nutzer wählt die Optionen »Graphen« → »Graphen speichern«.
2) Anschließend wählt er die zu speichernden Graphen aus und bestätigt.
3) Die Graphen werden in einer Datei kodiert und auf der Festplatte gespeichert.

9.1.11 Heuristiken speichern

Vorbedingung: Es gibt mindestens eine Heuristik in der Liste der Heuristiken.

Nachbedingung: Heuristiken werden korrekt in einer Datei auf der Festplatte gespeichert.

Ablauf:

1) Der Nutzer wählt die Optionen »Hilfe & Sonstiges« → »Heuristiken speichern«.
2) Anschließend wählt er die zu speichernden Heuristiken aus und bestätigt.
3) Die Heuristiken werden in einer Datei kodiert und auf der Festplatte gespeichert.

9.1.12 Heuristiken laden

Vorbedingung: Die einzulesende Datei ist korrekt und es wurde mindestens ein Graph erstellt.

Nachbedingung (Erfolg): Heuristikeinstellungen werden aus der Datei gelesen und in die Heuristikauswahl übernommen.

Nachbedingung (Misserfolg): Es erscheint eine aussagekräftige Fehlermeldung.

Ablauf:

1) Der Nutzer wählt die Optionen »Hilfe & Sonstiges« → »Heuristiken importieren«.
2) a) Falls der Nutzer die Preview offen hat, liest das Programm die Datei aus und importiert die Heuristiken in die globale Liste.
   b) Falls der Nutzer die Detailansicht offen hat, liest das Programm die Datei aus und importiert die Heuristiken in die lokale Liste.
3) Bei Misserfolg wird der Grund des Fehlers angezeigt, falls er ermittelt werden kann.

9.1.13 * Einfügen eines Heuristik-Plugins

Vorbedingung: Das Plugin ist korrekt geschrieben und compiliert.

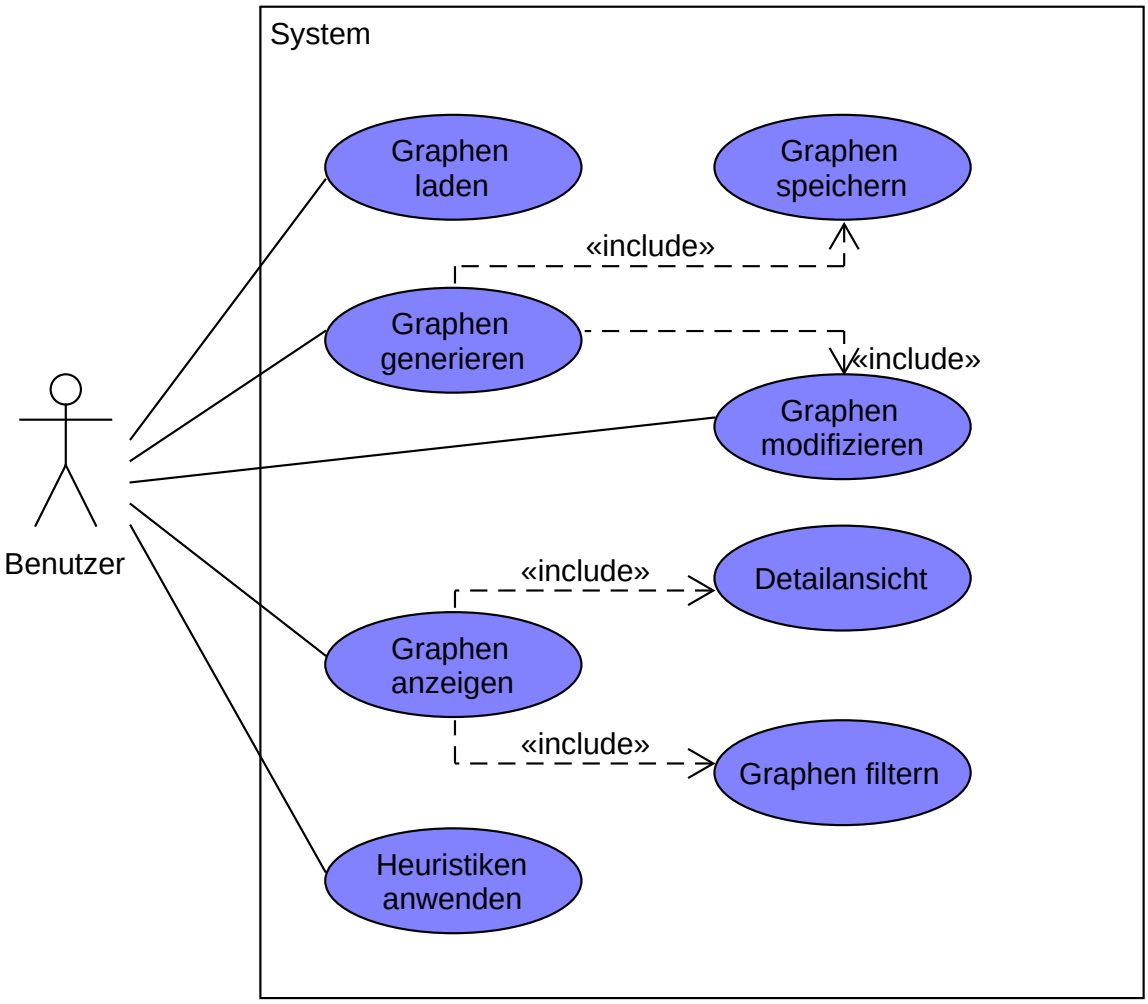
Nachbedingung (Erfolg): Das Plugin wird in das Programm übernommen und seine Heuristiken werden korrekt erkannt.

Nachbedingung (Misserfolg): Es erscheint eine aussagekräftige Fehlermeldung.

Ablauf:

1) Der Nutzer informiert sich unter »Hilfe & Sonstiges« → »Heuristik-Plugins« über das Einfügen von Heuristik-Plugins.
2) Das Plugin wird in dem angegebenen Ordner abgelegt.
3) Beim nächsten Programmstart werden alle Plugins aus dem Ordner geladen.
4) a) Bei Erfolg erscheinen die Heuristiken des hinzugefügten Plugins in den auswählbaren Heuristiken.
   b) Bei Misserfolg wird der Grund des Fehlers angezeigt, falls er ermittelt werden kann.

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

10.1 Mathematische Definitionen

Graph Überbegriff für alle Graphentypen, insbesondere der hier aufgeführten

einfacher ungerichteter Graph
Tupel $G = (V, E)$ mit Knotenmenge V und Kantenmenge $E \subseteq \{e \subseteq V \mid \#e = 2\}$

einfacher Hypergraph
Tupel $G = (V, E)$ mit Knotenmenge V und Hyperkantenmenge $E \subseteq \{e \subseteq V \mid \#e > 1\}$ mit der Eigenschaft, dass $\forall e, f \in E : \#e \cap f \leq 1$

r-uniformer Hypergraph
Hypergraph $G = (V, E)$ mit der Eigenschaft, dass $\forall e \in E : \#e = r$

Adjazenz von Knoten
$x, y \in V$ mit $\exists e \in E : x, y \in E$

Adjazenz von Kanten
$e, f \in E$ mit $e \cap f \neq \emptyset$

Knotenkontraktion
Für einen einfachen ungerichteten Graph G und $x \neq y \in V$ ist $G' = (V', E')$ definiert durch $V' = V \setminus \{x, y\} \cup \{v_{xy}\}$, wobei $v_{x,y} \notin V$, und $E' = E_1 \cup E_2$ mit $E_1 = \{e \in E \mid x, y \notin e\}$ und $E_2 = \{\{v, v_{x,y}\} \mid \{v, x\} \in E \text{ oder } \{v, y\} \in E\}$

Heuristik
Algorithmus, der mit auf Vermutungen basierenden Methoden, versucht zu einer Lösung des Problems zu kommen. Falls eine Lösung gefunden wird, kann diese von der optimalen Lösung abweichen.

Zusammenhängender Graph
$\forall x, y \in V \; \exists n \in \mathbb{N} \; \exists e_1, ..., e_n \in E$, sodass $x \in e_1$, $y \in e_n$ und $\forall i \in \{1, ..., n-1\} : e_i \cap e_{i+1} \neq \emptyset$

Baum
einfacher, ungerichteter, zusammenhängender Graph mit $\#E = \#V - 1$

Bipartiter Graph
einfacher, ungerichteter Graph mit $\exists A, B \subseteq V$, sodass $A \cap B = \emptyset$, $A \cup B = V$ und $\forall e = \{x, y\} \in E :$ Œ $x \in A, y \in B$

Pfad
Für $n \in \mathbb{N}$ ist $P_n = (V, E)$ mit $V = \{v_1, ..., v_n\}$ und $E = \{\{v_i, v_i + 1\} \mid i = 1, ..., n-1\}$

Kreis
Für $n \in \mathbb{N}$ ist $C_n = (V, E)$ mit $V = \{v_1, ..., v_n\}$ und $E = \{\{v_i, v_i + 1\} \mid i = 1, ..., n-1\} \cup \{\{v_n, v_1\}\}$

Clique
Für $n \in \mathbb{N}$ ist $K_n = (V, E)$ mit $V = \{v_1, ..., v_n\}$ und $E = \{e \subseteq V \mid \#e = 2\}$

Knotenkolorierung
$c : V \to \mathbb{N}$ mit $\forall x, y \in V$ mit $\exists e \in E : x, y \in e$ gilt $c(x) \neq c(y)$

Kantenkolorierung
$c : E \to \mathbb{N}$ mit $\forall e, f \in E$ mit $e \cap f \neq \emptyset$ gilt $c(e) \neq c(f)$

Knotenkolorierung
$c : V \cup E \to \mathbb{N}$ mit $c|_V$ ist Knotenkolorierung, $c|_E$ ist Kantenkolorierung und $\forall v \in V, e \in E$ mit $v \in e$ gilt $c(v) \neq c(e)$

Heuristik
: Algorithmus, der mit auf Vermutungen basierenden Methoden, versucht zu einer Lösung des Problems zu kommen. Falls eine Lösung gefunden wird, kann diese von der optimalen Lösung abweichen.

10.2 Terminologie

Generieren
: Erstellen von Datenmengen mit probabilistischen Methoden.

Plugin
: Optionale Softwarekomponente, die eine bestehende Software erweitert. Sie wird oft vom Benutzer installiert und von der Hauptanwendung zur Laufzeit eingebunden. Ein Plugin ist ohne Hauptanwendung nicht lauffähig.

Speichern und Laden
: Das Programm nutzt zunächst nur den flüchtigen Arbeitsspeicher. Sollen Daten über die Laufzeit des Programms hinaus gesichert werden, so müssen diese auf die Festplatte geschrieben werden. Dieser Prozess heißt speichern. Die Umkehrung des Prozesses, d.h. das Wiedereinfügen der Daten in den Arbeitsspeicher, heißt laden.

10.3 Akronyme

KIT Karlsruher Institut für Technologie

IPD Institut für Programmstrukturen und Datenorganisation