Softwareprojekt 2019

Gruppe 2 – Schnelle Layoutverfahren für das Zeichnen von Graphen in VANTED

Projektbeschreibung

Implementierung zweier Netzwerklayouts für große Netzwerke als Add-on für das Java-Framework VANTED (vanted.org).

Gruppenarbeit

Die Arbeit im Software-Projekt wird in Gruppen von üblicherweise 5-6 Personen stattfinden. Ein wichtiges Lehrziel ist die Arbeit in der Gruppe, was unter anderem regelmässige Kommunikation, Koordination, und gemeinsame Projektplanung beinhaltet. Als Endergebnis soll die Gruppe gemeinsam die Implementierung, Dokumentation und das Testen als Projektbestandteile bearbeitet haben.

Achtung: Die folgenden Informationen sind nur eine grobe Übersicht über Ablauf und Planung, die im ersten Treffen dann näher erläutert wird. Dazu wird es dann eine genauere Beschreibung in einem dedizierten Dokument geben, das in der ersten Woche bereitgestellt wird, und welches für die Bewertung relevant ist.

Ablauf

Es wird während des Semesters regelmässige Treffen der Gruppen mit den Betreuern geben. In diesem Treffen soll der Fortschritt vorgestellt und eventuelle Fragen geklärt werden. Dazu soll jeweils ein Gruppenmitglied ein Protokoll schreiben und ins Repository legen. Code und Dokumentation sollen ebenso im von uns bereitgestellten Uni-Gitlab Repository gespeichert werden.

Es wird drei Milestone-Termine geben, bis zu denen vorgegebene Zwischenziele erreicht werden müssen, für welche Ihre erarbeiteten Ergebnisse dann von Ihnen vorgestellt werden. Am Ende des Semester erfolgt eine Abschlusspräsentation.

Grober Ablauf also: Einführungsveranstaltung, Pfilchtenheft/SDS, Meilensteine, (un)regelmäßige Treffen, Einzelgespräche, Abschlusspräsentation

Generelle Planung und erwartete Ergebnisse

Für einen erfolgreichen Abschluss des Projekts wird erwartet, dass ein Softwareprojekt jeweils innerhalb einer Gruppe geplant und verwaltet wird, die entsprechende Software funktionsfähig implementiert wird, so dass sie die gestellten Anforderungen erfüllt, und dass Sie die Software ordentlich testen und dokumentieren.

Kriterienkatalog: Commits + Issue tracking als Teil-Metrik, Planung und Management der Gruppe, Erfolg der Implementierung, Dokumentation und Präsentation.

Projekt VANTED Erweiterung

Überblick:

VANTED ist ein java-basiertes Open-Source Netzwerkvisualisierungs- und Analysetool mit einem Fokus auf Anwendungen in den Lebenswissenschaften (www.vanted.org).

Die Funktionalität von VANTED kann mithilfe von Addons erweitert werden. Es existiert ein ausführliches Beispiel-Addon, mit dessen Hilfe Sie sich in die Software einarbeiten können.

Ziele: Das Ziel dieses Projekts ist es, ein funktionierendes Addon für zwei Netzwerklayoutverfahren zu implementieren und zu testen. Bei einem davon, dem Multilayerverfahren, handelt es sich lediglich um ein Framework, nicht um einen Algorithmus.

Umsetzung eines Netzwerklayoutverfahrens für ungerichtete Graphen ("Stress-Minimization"). Stress ist ein Ausdruck, der eine bestimmte Art von Zielfunktion zur Optimierung in Layoutverfahren bezeichnet. Grob gesagt geht es um den Unterschied zwischen dem Knotenabstand in der Zeichnung und einem vorher festgelegten Wert. Dieser

festgelegte Wert kann z.B. einfach die graphentheoretische Distanz sein (d.h. umso näher zwei Knoten im Graphen liegen, desto näher sollen sie auch in der Zeichnung liegen).

Hier soll eine einfache iterative Variante als Basisversion implementiert werden.
Dabei werden zunächst ungewichtete kürzeste Wege zwischen allen Knotenpaaren berechnet, dann wird ein initiales Layout bestimmt, und dieses dann iterativ verbessert, bis ein Abbruchkriterium erreicht ist (z.B. dass der Fehler unter einem Schwellwert liegt, oder eine von der Graphgröße abhängige Zahl von Iterationen.
Zusätzlich sollen die Implementierungen auf Korrektheit getestet werden, sowie die Effizienz überprüft und ggf. verbessert werden.

Das Multilevelframework besteht aus drei Teilen, der Aggregation, dem Platzieren, und dem Zeichnen.
Es erzeugt bei der Aggregation ausgehend vom Eingabegraph eine Hierarchie von vereinfachten Versionen des Graphs, z.B. indem jeweils eine Menge von Knoten nach einer fixen Vorschrift aggregiert werden zu einem einzigen Knoten. Eine einfache Vorschrift wäre z.B. dass zwei benachbarte Knoten zusammengelegt werden bis die Knotenanzahl halbiert ist. Eine solche Vorschrift wird dann auf jeder Hierarchiebene durchgeführt, bis der Graph auf einer Ebene unter eine gewisse Größe fällt.
Ausgehend vom kleinsten Graph wird dann auf jeder Ebene mit einem Layoutverfahren ein Layout erzeugt, und dann werden die Knoten der nächsten Ebene mit einer einfachen Heuristik eingefügt für die nächste Iteration. Eine einfache Heuristik wäre z.B., die neuen Knoten zunächst an die Stelle des Knotens zu platzieren, mit dem sie in der Aggregation verschmolzen wurden. Danach startet die nächste Iteration.

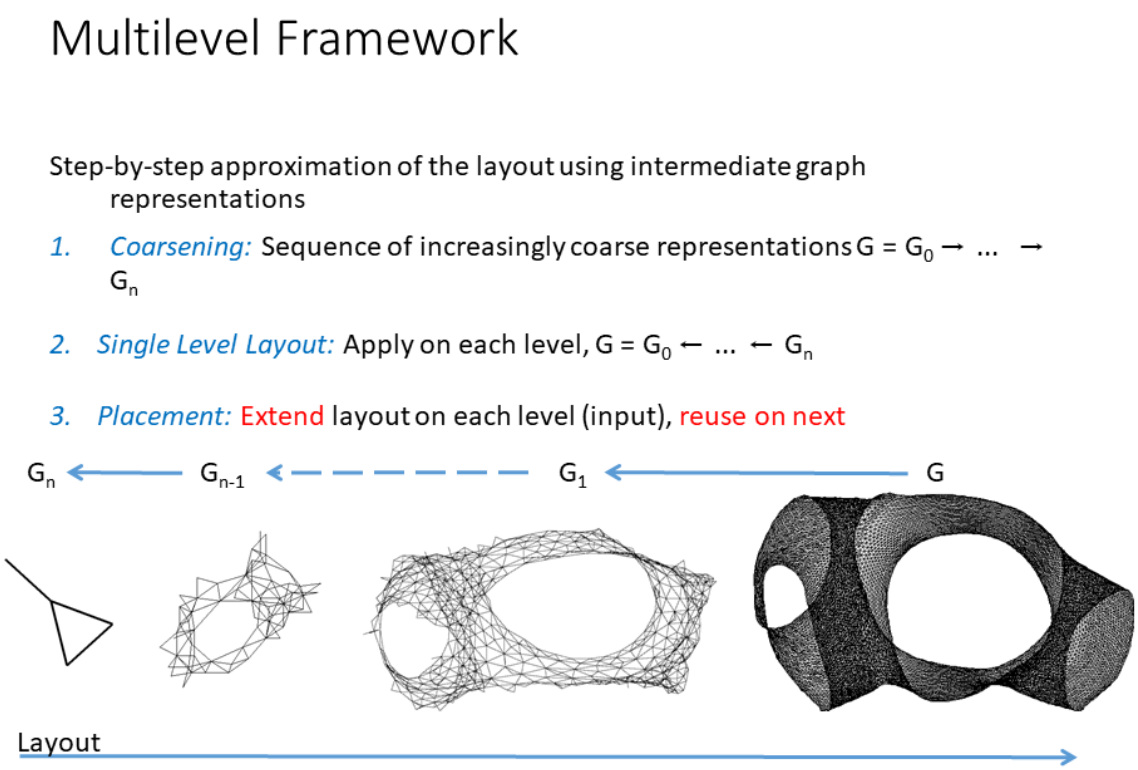
Erwartete Mindestziele:

- Ein Vanted Addon mit einem Layoutverfahren mit Stress Minimization für ungerichtete Graphen
- Einbau des Multilevelframeworks, welches die Auswahl eines Zeichenverfahrens erlaubt das auf jeder Hierarchieebene angewandt werden soll.

- Modularer Aufbau des Frameworks so dass einzelne Schritte auch mit anderen Verfahren durchgeführt werden können
- Klar spezifizierte Tests und Benchmarksets
- Effiziente Implementierung, so dass große Graphen (mehr als tausend Knoten und Kanten) in wenigen Sekunden gerechnet werden können (nicht wesentlich langsamer als die bekannten Implementierungen zum Beispiel in der OGDF Bibliothek)
- Saubere Dokumentation und Kommentierung des Codes

Erste Schritte:

- Pflichtenheft (wir können ein Beispiel bereitstellen falls notwendig)
- Einarbeiten in Vanted-Erweiterungen (es existiert ein Beispiel-Addon, dass die benötigten Bestandteile der Architektur bereits zeigt)
- Einarbeiten in Netzwerklayouts



Das Multilevel-Verfahren benötigt drei Bestandteile:

1. Vereinfachung in einer Hierarchie
2. Layoutverfahren zum Aufruf auf jeder Hierarchieebene
3. Platzierung zum Einfügen von Knoten aus der nächsten Hierarchiebene

Für Stress-basierte Layoutverfahren gibt es eine Vielzahl von Variationen und Verbesserungen.

Literatur:

Bestehende Software als Beispiele, zum Vergleichen, und zur Orientierung – NICHT zum Einbinden!

1. OGDF https://www.ogdf.net
2. GraphViz https://graphviz.gitlab.io/ , https://gitlab.com/graphviz/ (mit einigen Veröffentlichungen und Graph Viewer Verweisen)
3. JGraphT https://jgrapht.org/

Veröffentlichungen und Folien: Siehe Repository