Fachpraktikum Erweiterbare Datenbanksysteme
Wintersemester 2015/16

Aufgabe 1 der Phase 2

Ein SECONDO-basierter Routenplaner mit Berücksichtigung von Steigungen

Ralf Hartmut Güting, Thomas Behr, Holger Helmut Hennings, Fabio Valdés

19.11.2015

Lehrgebiet Datenbanksysteme für neue Anwendungen
Fakultät für Mathematik und Informatik

FernUniversität in Hagen

1 Motivation

Im Rahmen dieser Aufgabe sollen Sie SECONDO derart erweitern, dass mit Hilfe eines Straßennetzes sowie präziser Höhendaten ein für den Anwender optimaler Weg zwischen zwei Punkten bestimmt werden kann. Dabei werden die als Punktwolke vorliegenden Höheninformationen in sogenannte triangulierte irreguläre Netzwerke überführt, bevor sie mit dem aus OpenStreetMap (OSM) importierten und mit bereits existierenden SECONDO-Skripten erzeugten Straßennetzen kombiniert werden können. Nun ist es möglich, unter Berücksichtigung von Benutzereinstellungen optimale Wege zwischen zwei Punkten zu berechnen, die durch die Eingabe von Adressen festgelegt werden. Die gesamte Interaktion mit dem Benutzer erfolgt in einer Erweiterung der graphischen Oberfläche von SECONDO.

Bislang ist es in SECONDO möglich, OSM-Daten einzulesen und aus ihnen ein Straßennetzwerk zu erzeugen. Außerdem bietet das System Datentypen, die einer Linie eine Folge von reellen Funktionen zuordnen. Auch ein Operator zur Bestimmung geographischer Koordinaten nach Eingabe einer postalischen Adresse ist bereits vorhanden.

2 Aufgabenbeschreibung

Im Folgenden beschreiben wir die einzelnen Schritte, die zur Bearbeitung der Aufgabe erforderlich sind. Dabei war es unser Ziel, einzelne Anforderungen möglichst voneinander abzugrenzen, so dass Ihnen die Gliederung in diesem Kapitel möglicherweise bei der Aufgabenverteilung innerhalb Ihrer Gruppe hilft.

2.1 Import von Höhendaten in Punktwolken

Die zu importierenden Höhendaten liegen ursprünglich im Dateiformat LAS oder LAZ vor. Dabei ist das letztgenannte Format lediglich eine komprimierte Version des erstgenannten. Dateien im LAZ-Format können mit Hilfe der frei verfügbaren Software laszip [1] ins LAS-Format entpackt werden. Eine ausführliche Dokumentation zu diesem Dateiformat sowie zahlreiche Beispieldaten zum Herunterladen finden Sie unter [2]. Sie sollen SECONDO um einen Operator **importpointcloud** erweitern, der eine oder mehrere LAS-Dateien einliest (z.B. durch Eingabe des Dateipfads oder des Verzeichnisses) und in einen Strom von Punktwolken umwandelt.

Der Typ Punktwolke (*pointcloud*) soll von Ihnen als Attributdatentyp in SECONDO realisiert werden. Eine solche Punktwolke enthält bis zu 50.000 dreidimensionale Punkte, die jeweils aus drei reellen Zahlen (Längen-, Breiten- und Höhenangabe) bestehen. Die einzelnen *pointcloud*-Objekte sollen dabei disjunkte rechteckige Teilbereiche des ursprünglichen Bereichs aus den LAS-Dateien repräsentieren und intern mit einem FLOB realisiert sein. Sie werden gemeinsam in einer neuen Relation abgelegt, über die Sie einen R-Baum anlegen sollen, damit die einzelnen Teilbereiche schnell angesteuert werden können. Zu diesem Zweck gibt es in SECONDO bereits eine R-Baum-Implementierung in der RTreeAlgebra mit Operatoren zum Erstellen und Abfragen eines R-Baums. Innerhalb der einzelnen Punktwolken sollen die jeweiligen Punkte in einem Gitter verwaltet werden, so dass zu einem Anfragerechteck (z.B. Bounding Box eines Straßenabschnitts) effizient die in Frage kommenden Punktmengen ermittelt werden können. Sie dürfen selbst entscheiden, ob Sie die Größe der Gitterzellen statisch oder dynamisch implementieren, dabei soll in jedem Fall die Anzahl von 2.500 Zellen pro Punktwolke nicht überschritten werden.

2.2 Erweiterung eines Straßennetzes um Höhendaten

Wir beschreiben zunächst, wie ein Straßennetz in SECONDO importiert werden kann. Das Projekt OpenStreetMap [3] enthält detaillierte geographische Daten der gesamten Welt. Datensätze zu einzelnen Bereichen wie Kontinenten, Staaten oder Verwaltungsbezirken lassen sich auf der verwandten Seite Geofabrik [4] kostenfrei herunterladen. OSM-Daten enthalten u.a. präzise Verläufe von Verkehrswegen wie Autobahnen, Fahrrad- und Fußwegen sowie Wohnstraßen und sogar Radfahrstreifen auf Fahrbahnen, so dass sie für unsere Zwecke sehr gut geeignet sind. Bitte wählen Sie stets diejenige Version aus, die auf `.osm.bz2` endet. Nach dem Entpacken kann die Datei über ein komplexes Skript [5] in eine SECONDO-Datenbank eingelesen und in eine Relation `Edges` überführt werden. Vor der Ausführung muss noch der Name der

zu importierenden Datei in Zeile 18 an der entsprechenden Stelle (zwischen den einfachen Anführungszeichen) eingetragen werden.

Anschließend existiert in der Datenbank eine Relation `Edges`, die u.a. ein Attribut `Curve` vom Typ *sline* besitzt, also einen Linienzug, der den Straßenverlauf genau beschreibt. Es ist nun Ihre Aufgabe, ein Attribut namens `Altitude` zu der `Edges`-Relation hinzuzufügen, das zu jeder Straße das passende Höhenprofil enthält. Hierzu gibt es in SECONDO bereits die LineFunctionAlgebra, deren Datentyp *lreal* Abschnitten eines Linienzugs eine lineare reelle Funktion zuordnet. Damit lässt sich das Höhenprofil einer Straße sehr effizient darstellen. Die Schwierigkeit besteht nun darin, die `Edges`-Relation mit Punktwolken zu kombinieren und zu jeder Straße (*sline*) in akzeptabler Zeit ein passendes *lreal* zu erzeugen. Dies soll in einem neuen Operator **lcompose** geschehen. Mit Hilfe des R-Baums über die Punktwolken lassen sich diejenigen finden, deren Bounding Boxen mit der Bounding Box der *sline* überlappen. Innerhalb des neuen Operators sollen nun diejenigen Bestandteile der Punktwolken, die in einem bestimmten Bereich um die Straßen liegen (weniger als 20 Meter entfernt) gefunden werden. Abbildung 1 zeigt beispielhaft die gefilterten Punktwolken in der Nähe der Straßen.

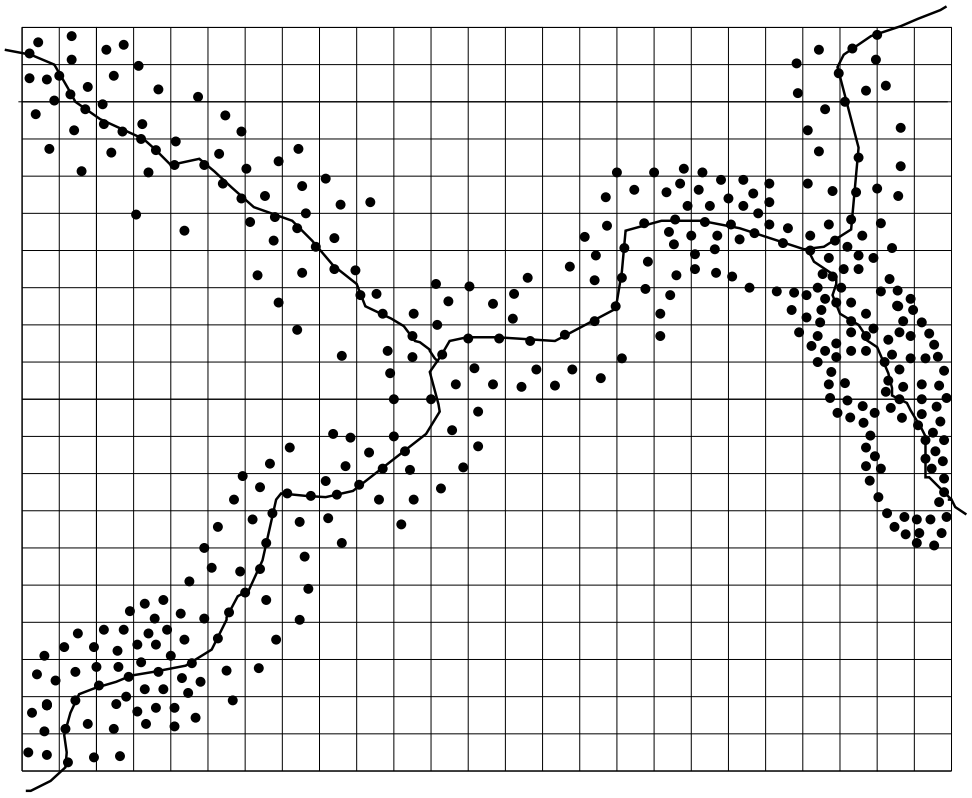

Abbildung 1: Punktwolken beschränkt auf eine Umgebung des Straßennetzes

Nun werden diese Teilpunktwolken in ein unregelmäßiges Dreiecksnetz (triangulated irregular network, TIN) überführt. Ein solches TIN ist i.A. wesentlich speicherplatzeffizienter als eine Darstellung als Punktwolke, wenn bei der Umwandlung eine gewisse Ungenauigkeit in Kauf genommen wird. Als Beispiel für ein TIN ist in Abbildung 2 eine triangulierte Gebirgslandschaft dargestellt. Sie können sich an der bereits in SECONDO implementierten TinAlgebra orientieren, jedoch soll diese Umwandlung innerhalb Ihres Operators im Hauptspeicher geschehen, und das TIN soll nicht als Datenbankobjekt abgespeichert werden. Verwenden Sie im Operator **lcompose** einen zusätzlichen Parameter, mit dem der Nutzer eine gewünschte Genauigkeit für das TIN vorgeben kann (zwischen 0 und 1; bei 0 gibt es nur ein Dreieck pro *sline*, und bei 1 wird jedes Element der Punktwolke exakt durch einen Punkt eines Dreiecks dargestellt). Mit Hilfe des erzeugten TIN sollen Sie nun für jede *sline* aus der Edges-Relation ein *lreal* berechnen, d.h.,

jedem Abschnitt der *sline* eine lineare reelle Funktion zuordnen. Sie können den Genauigkeitsparameter auch, anstatt damit die Genauigkeit des TIN zu kontrollieren, auf die *lreal*-Erstellung anwenden und deren Detailliertheit verringern. So würde bei einem Faktor von 1 jedes Dreieck berücksichtigt, während bei 0 jede *sline* nur eine reelle Funktion erhalten würde.

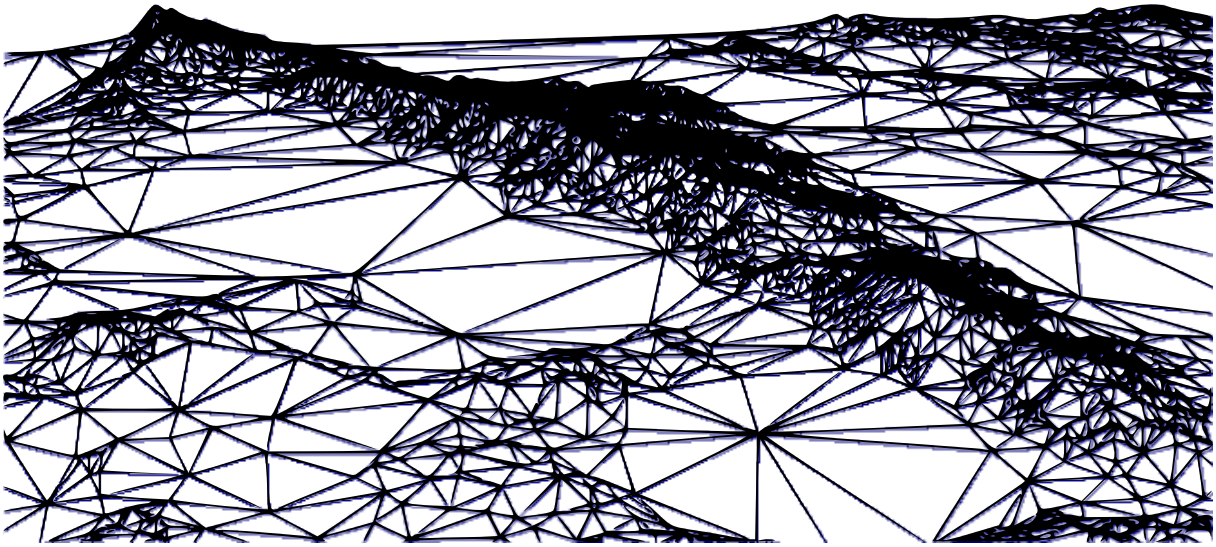

Abbildung 2: Eine Gebirgslandschaft in Form eines TIN (Quelle: [7])

2.3 Bestimmung des optimalen Wegs

Der gewünschte Start- und Endpunkt soll vom Benutzer als postalische Adresse eingegeben werden können. Der existierende Operator **geocode** akzeptiert eine Adresse mit den vier Bestandteilen Straße, Hausnummer, Postleitzahl und Ort. Diese Funktionalität sollen Sie insofern nutzen und erweitern, als bei einer unvollständigen oder mehrdeutigen Eingabe dem Benutzer einige Vorschläge unterbreitet werden. Wenn z.B. keine Postleitzahl angegeben wurde und mehrere Orte mit dem Namen existieren, soll dem Benutzer eine Liste passender Orte mit Postleitzahlen vorgeschlagen werden, aus denen er einen auswählen kann. Bei einer fehlenden Hausnummer wird einfach auf die kleinste verfügbare Hausnummer der Straße zurückgegriffen.

Auf Basis der durch **lcompose** erzeugten Höhenprofile (*lreal*) besteht Ihre Aufgabe nun darin, optimale Wege zwischen zwei Punkten zu berechnen. Der zu implementierende Operator **shortestpath** soll unter Berücksichtigung der Präferenzen des Benutzers, die die Kantengewichte beeinflussen, den A*-Algorithmus [6] umsetzen. So möchten manche Benutzer womöglich auf starke Steigungen verzichten und dafür lieber einen längeren Weg mit sanfteren Anstiegen in Kauf nehmen, während andere gezielt nach solchen Herausforderungen suchen. Es soll ebenfalls möglich sein, den **shortestpath**-Operator ohne eine solche Liste von Präferenzen aufzurufen. In diesem Fall wird der optimale Weg ohne Berücksichtigung des Höhenprofils bestimmt. Beachten Sie bei der Implementierung des A*-Algorithmus, dass die Nutzerpräferenzen keinen allzu großen Einfluss auf die Kantengewichte haben dürfen. Der Algorithmus funktioniert nicht korrekt, wenn durch einen Faktor < 1 ein Kantengewicht geringer ist als die geschätzte (euklidische) Distanz. Zu große Werte dagegen verschlechtern möglicherweise die Effizienz des Algorithmus, so dass dieser seine Vorteile gegenüber dem Algorithmus von Dijkstra einbüßen kann. Sie können und sollen sich bei der Implementierung am bereits vorhandenen Operator **oshortestpatha** aus der OrderedRelation-Algebra orientieren.

2.4 Erweiterung der Secondo GUI

Während für die Operatoren die Aufgabenteile 2.1 und 2.2 keine graphische Darstellung sinnvoll oder erforderlich ist, soll der durch **shortestpath** berechnete Weg im Höse-Viewer durch eine entsprechende Markierung bzw. Einfärbung des Streckenverlaufs angezeigt werden. Für eine komfortable Eingabe der Höhenpräferenzen und der Adressen sollen Sie einen neuen Viewer namens Shortestpath-Viewer implementieren. Die Höhenpräferenzen sind in einer kleinen Tabelle einzugeben, wo der Benutzer für alle Steigungsbereiche (0 bis 5%, 5 bis 10%, usw.) angeben kann, ob er diese ablehnt oder bevorzugt (auf einer Skala von 0 bis 5).

Schließlich soll es dem Benutzer möglich sein, sich in einem separaten Fenster den Höhenverlauf des zuletzt berechneten Wegs anzuzeigen. Dieser kann beispielsweise so aussehen wie in Abbildung 3.

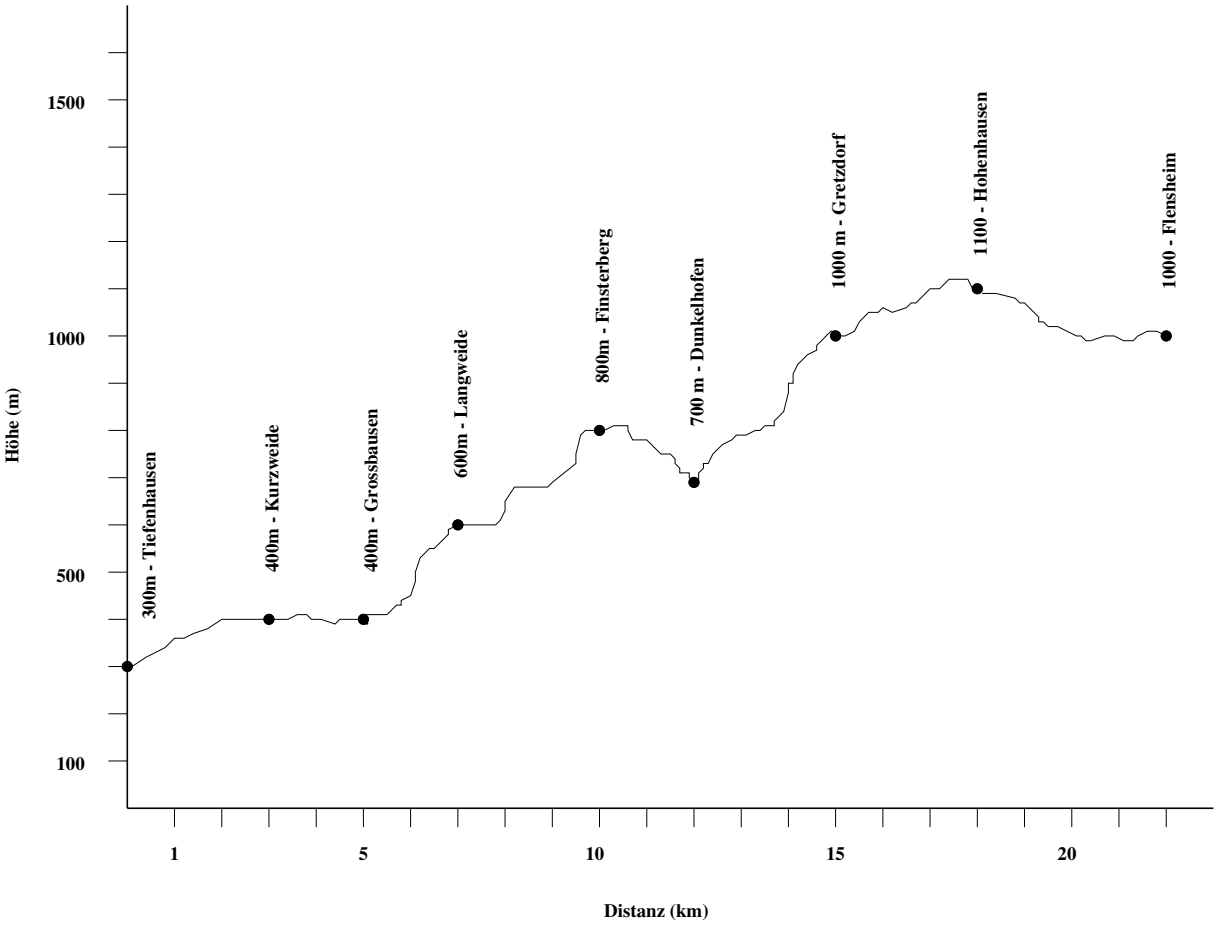

Abbildung 3: Höhenprofil eines berechneten Wegs

3 Übersicht

Es folgt eine Übersicht der zu implementierenden Operatoren.

importpointcloud	: {*string*, *text*}	→ *stream*(*pointcloud*)
lcompose	: *sline* × *stream*(*pointcloud*) × *real*	→ *lreal*
shortestpath	: *orel*(*tuple*(X)) × IDENT^2 × *point*^{2} × pref	→ *stream*(*sline*)

Dabei steht X für eine beliebige Attributmenge und `pref` für einen Datentyp, der die Präferenzen des Benutzers darstellt. Die Schreibweise IDENT^2 bedeutet, dass zwei Namen von Attributen aus dem Tupelstrom angegeben werden müssen, deren Datentyp *sline* und *lreal* ist. Schließlich sind zwei Punkte (*point*^{2}) sowie eine Liste von Nutzerpräferenzen anzugeben.

Nachfolgend zeigen wir beispielhaft, wie diese Operatoren verwendet werden könnten. Die Anfrage

```
let Pointclouds = importpointcloud("elevationdata/") consume
```

liest beispielsweise alle Dateien im `elevationdata`-Verzeichnis (das in diesem Fall direkt unterhalb des `bin`-Verzeichnisses liegen muss) ein, die im Format LAS vorliegen. Ergebnis ist eine Relation mit genau einem Attribut vom Typ *pointcloud*. Dieses kann durch den Befehl

```
let Pointclouds_RTree = Pointclouds creatertree[Pointcloud]
```

indiziert werden. Nach dem Import der Punktwolken und nach Erzeugung der Relation `Edges` kann der Operator **lcompose** in der Anfrage

```
let Edges2 = Edges feed extend[Altitude: .Curve lcompose[Pointclouds_RTree
                        Pointclouds windowintersects[bbox(.Curve)], 0.75] ]
```

dazu verwendet werden, ein Attribut `LineFunction` vom Typ *lreal* zu berechnen. Schließlich ist

```
query Edges2 shortestpath[Curve, LineFunction, start, end, pref]
      transformstream consume
```

eine mögliche Anwendung der Routenberechnung von Startpunkt `start` nach Endpunkt `end` mit den Nutzerpräferenzen `pref`.

Literatur

[1] `http://www.laszip.org/`.

[2] `http://www.asprs.org/Committee-General/LASer-LAS-File-Format-Exchange-Activities.html`.

[3] `http://www.openstreetmap.org`.

[4] `http://download.geofabrik.de/`.

[5] `secondo/bin/Scripts/OrderedRelationGraphFromFullOSMImport.SEC`.

[6] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics*, 4(2):100–107, 1968.

[7] Michigan State University. `https://www.msu.edu/~ashton/classes/866/notes/lect01/tin_01.gif`.