Praktikum Algorithmen

Wintersemester 2025

TECHNISCHE UNIVERSITÄT DARMSTADT

Da die Themen des Praktikums im Rahmen des MOTIS[1] Projekts stattfinden, erfordern sie eine selbstständige Einarbeitung in selbiges. Unsere Erfahrungen haben gezeigt, dass Studierende die Komplexität eines modernen, real-world C++ Projekts und den damit verbundenen Arbeitsaufwand häufig unterschätzen.

Führen Sie daher bitte **vor** einer Anmeldung die folgenden Schritte zur Selbsteinschätzung Ihrer Eignung durch.

1. Forken Sie das MOTIS Projekt auf GitHub und klonen Sie Ihren Fork
2. Bauen Sie das Projekt:
   - Linux Developer Setup[2]
   - Windows Developer Setup[3]
   - MacOS Developer Setup[4]
3. Setzen Sie entsprechend der Anleitung[5] einen Motis Server auf
4. Rufen Sie `localhost:8080` mit Ihrem Browser auf und prüfen Sie die Funktionalität mit einigen Testanfragen an das Routingsystem

Uns ist bewusst, dass das erstmalige Aufsetzen des Projekts je nach Vorkenntnissen mit einer steilen Lernkurve verbunden sein kann. Die Selbsteinschätzung dient nicht dazu Studierende von der Teilnahme am Praktikum auszuschließen, sondern soll eine Hilfestellung sein und Sie beim Sicherstellen Ihres Studienerfolges unterstützen. Das selbstständige Einarbeiten in einen unbekannten, umfangreichen Quellcode anhand bereitgestellter Dokumentation ist eine Schlüsselfähigkeit, die Sie im Rahmen dieser Selbsteinschätzung verbessern können.

---

[1] `https://github.com/motis-project/motis`
[2] `https://github.com/motis-project/motis/blob/master/docs/linux-dev-setup.md`
[3] `https://github.com/motis-project/motis/blob/master/docs/windows-dev-setup.md`
[4] `https://github.com/motis-project/motis/blob/master/docs/macos-dev-setup.md`
[5] `https://github.com/motis-project/motis/blob/master/docs/dev-setup-server.md`

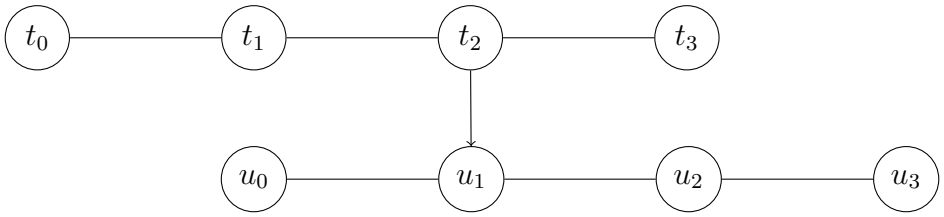
Praktikumsthema

A*-Routing auf dem zeitexpandierten Trip-based Graphen

nigiri ist ein schneller, speichereffizienter Router für den öffentlichen Verkehr. Um Periodizitäten des Fahrplans auszunutzen, werden die Betriebstage der Fahrten in Bitfeldern kodiert.

Das *nigiri* Repository auf GitHub: `https://github.com/motis-project/nigiri`

In diesem Praktikum soll ein Routing-Algorithmus auf dem zeitexpandierten Graphen des Trip-based Algorithmus [1] implementiert werden. Eine Fahrt einer Linie des öffentlichen Verkehrs ist eine Abfolge elementarer Verbindungen, im Weiteren Segmente genannt. Im folgenden Diagram sind zwei Fahrten t und u dargestellt. Die Kreise stellen die Halte der Fahrten dar. Die Verbindungen zwischen den Halten sind die Segmente. Zwischen t_2 und u_1 ist ein möglicher Umstieg durch den Pfeil dargestellt.



Im Graphen des Trip-based Algorithmus sind die Segmente die Knoten und die Umstiege die Kanten. Außerdem besteht eine umstiegsfreie Kante von jedem Segment zu dessen Folgesegment, sofern eines existiert. Im obigen Beispiel findet demzufolge ein Umstieg aus dem zweiten Segment der Fahrt t in das zweite Segment der Fahrt u statt.

In einem Preprocessing-Schritt werden alle für Pareto-optimale Verbindungen notwendigen Umstiege zwischen allen Segmenten des Fahrplans vorberechnet, so dass der Algorithmus zum beantworten einer Routinganfrage nur Fahrten, die zu optimalen Verbindungen führen können, erreicht.

Der Trip-based Algorithmus und dessen relevante Datenstrukturen sowie auch das erwähnte Preprocessing sind im *nigiri* Repository unter folgendem Pfad implementiert.

`include/nigiri/routing/tb`

Durch die Kodierung der Betriebstage in Bitfeldern ist zur Identifizierung einer bestimmten Fahrt der Segment Index und der Day Index, d.h. an welchem Tag des Fahrplans die Fahrt stattfindet, notwendig. Im Praktikum beschränken wir die maximale Reisezeit auf 24 Stunden. Dadurch ist es ausreichend pro Segment genau einen Day Index zu speichern, um nachzuvollziehen, welche Fahrt gemeint ist.

Um ein Routing auf dem Trip-based Graphen durchzuführen sind die folgenden Datenstrukturen notwendig:

- Mapping $\tau(s)$: Segment $\rightarrow$ Ankunftszeit
- Mapping: Segment $\rightarrow$ Day Index
- Predecessor Table (für die Rekonstruktion des Pfades)

Das Routing soll eine Anfrage $(\tau, p_{start}, p_{dest})$ beantworten. Diese besteht aus einem Startzeitpunkt τ, einem Startort p_{start} und einem Ziel p_{dest}.

Das Routing soll den Graphen basierend auf einer Priority Queue durchsuchen. Ein Eintrag der Queue (s, t) setzt sich zusammen aus dem Index des Segments s und der Anzahl der Umstiege t. Die Einträge der Queue sind sortiert nach ihren Kosten entsprechend einer Funktion $f(s, t) = g(s, t) + h(s)$. Hierbei steht die Funktion $g(s, t)$ für die Kosten, die bis zum Segment s entstanden sind und $h(s_i)$ für eine Heuristik, die die weiteren Kosten bis zum Erreichen des Ziels abschätzt. Ein mögliche Kostenfunktion ist

$$f(s, t) = (\tau(s) - \tau) + \alpha t + \lambda(s)$$

wobei $\tau(s)$ die Ankunftszeit des Segment s, τ die angefragte Startzeit, α der Umrechnungsfaktor von Umstiegen in Minuten und $\lambda(s)$ eine untere Schranke für die noch verbleibende Reisezeit vom Segment s bis zum Ziel ist. Dadurch ist eine Suche mit Goal Direction auf dem Trip-based Graphen möglich. Die Funktion λ ist in *nigiri* bereits implementiert und kann von Ihnen genutzt werden. Ihre Implementierung soll außerdem den berechneten Pfad rekonstruieren und ausgeben können.

1. Milestone

Als 1. Milestone soll das oben beschriebene Routing in seinen Grundzügen implementiert werden. Als vorläufige Kostenfunktion genügt $f(s, t) = (\tau(s) - \tau) + \alpha t$, also ohne die Heuristik λ. Die Implementierung muss noch nicht vollständig korrekt sein und Sie dürfen sie im weiteren Verlauf des Praktikums noch erweitern. Die Abgabe erfolgt in Form eines Git Patches per E-Mail an **lab@algo.informatik.tu-darmstadt.de**. Die Deadline für die Abgabe des 1. Milestones ist:

31.01.2026 23:59:59 Uhr (MEZ)

Spätere Abgaben werden nicht berücksichtigt.

Finale Abgabe

Zusätzlich zum 1. Milestone soll die finale Abgabe Folgendes enthalten.

- Interfacing zur Klasse `search.h`, so dass Ihre Implementierung analog zu den beiden anderen bereits implementierten Algorithmen (RAPTOR und Trip-based) genutzt werden kann.
- Die vollständige Kostenfunktion $f(s,t) = (\tau(s) - \tau) + \alpha t + \lambda(s)$, d.h. inklusive der Heuristik λ soll verwendet werden.
- Eine Validierung, in der Sie zeigen, dass mit Ihrer Implementierung durch passend gewichtete Kostenfunktionen die Pareto-Menge von optimalen Verbindungen des Trip-based Algorithmus nachgebildet werden kann.
- Einen Praktikumsbericht, in dem Sie beschreiben, auf welche Probleme Sie während der Implementierung gestoßen sind und wie Sie diese gelöst haben. Erläutern Sie Ihre Ergebnisse aus der der Validierung und führen Sie Messungen der Performance Ihrer Implementierung durch. Der Umfang soll 3-4 Seiten betragen.

Folgende Deadlines sind einzuhalten:

16.03.2026 23:59:59 Uhr (MEZ) Bis zu dieser Deadline muss ein feature-vollständiger Stand als Git Patch per E-Mail geschickt werden – spätere Einreichungen werden nicht berücksichtigt.

17.03.2026 00:00:00 Uhr (MEZ) Ab dieser Deadline darf ein Pull Request an das *nigiri* Repository gestellt werden. Nun können ggf. noch Änderungen vorgenommen werden, um von der CI festgestellte Probleme zu beheben. Das Ziel ist, dass alle Checks der CI erfolgreich durchlaufen.

31.03.2026 23:59:59 Uhr (MESZ) Der Abgabetermin des Praktikumsberichtes per E-Mail und Deadline für letzte Änderungen am PR - spätere Einreichungen werden nicht berücksichtigt.

Anmeldung

Anmeldungen für das Praktikum werden **bis zum 20.10.2025 um 08:00 Uhr (MESZ)** unter der E-Mailadresse **lab@algo.informatik.tu-darmstadt.de** angenommen. Danach ist keine Bewerbung mehr möglich. Bewerbungen unter anderen E-Mailadressen werden ignoriert.

Kick-Off

Der Kick-Off findet am Mittwoch, den 22.10.2025, von 15:00 bis 16:00 Uhr in Raum S2|02 E202 statt.

Sprechstunde

Die Sprechstunde findet jeden Mittwoch von 15:00 bis 16:00 Uhr in Raum S2|02 E123 statt. Dafür ist eine vorherige Anmeldung per E-Mail notwendig. Die Deadline hierfür ist jeweils am Montag derselben Woche. Außerhalb davon sind Nachfragen nur per E-Mail möglich.

Fortschrittsberichte

Während des Semesters ist die Abgabe kurzer Fortschrittsberichte verpflichtend. Die Fälligkeiten entnehmen Sie bitte der Tabelle über den zeitlichen Ablauf unten. Dafür senden Sie jeweils eine E-Mail mit folgendem Inhalt an die E-Mailadresse lab@algo.informatik.tu-darmstadt.de:

- Welche Fortschritte / Erkenntnisse wurden seit dem letzten Bericht erreicht?
- Welche nächsten Schritte sind geplant?
- evtl. Liste mit Fragen - bitte achten Sie darauf, dass die Bewertungskriterien unter anderem selbstständiges Arbeiten berücksichtigen. Hier sollten Sie folglich nur Fragen stellen, deren Beantwortung nicht durch Google bzw. das Lesen von bereitgestelltem Material möglich ist.
- Möchten Sie am Mittwoch die Sprechstunde zur Beantwortung der Fragen oder Besprechen des weiteren Vorgehens besuchen?

Wird wiederholt kein Fortschrittsbericht abgegeben, ist einer weitere Bearbeitung des Praktikums ausgeschlossen.
Sollte es Ihnen nicht möglich gewesen sein am Praktikum zu arbeiten, begründen Sie dies bitte kurz im Fortschrittsbericht, mögliche Gründe können z.B. sein: Erkrankung, Workload aus anderen Veranstaltungen oder Lernen für Klausuren.

Zeitlicher Ablauf

Datum	Event
20.10.2025 08:00 Uhr	Deadline Anmeldung
22.10.2025 15:00 Uhr	Kick-Off in S2|02 E202
03.11.2025	Fortschrittsbericht
17.11.2025	Fortschrittsbericht
01.12.2025	Fortschrittsbericht
15.12.2025	Fortschrittsbericht
12.01.2026	Fortschrittsbericht
31.01.2026	1. Milestone Git Patch
16.02.2026	Fortschrittsbericht
02.03.2026	Fortschrittsbericht
bis 16.03.2026	feature-vollständiger Git Patch
ab 17.03.2026	Pull Request stellen
bis 31.03.2026	Finale Abgabe

Hinweise

- Planen Sie bitte zusätzlichen Aufwand ein, falls die genannten Programmiersprachen oder verwendeten Technologien (z.B. Git, CMake, etc.) neu für Sie sind.
- Als Referenz für die Komplexität des Projektes, das im Praktikum erweitert wird, dient MOTIS.
  Bitte überlegen Sie sich genau, ob Sie mit einer großen real-world Modern-C++ Code Basis klarkommen. Bisherige Erfahrungen haben gezeigt, dass Studierende den Einarbeitungsaufwand und die Komplexität der Programmiersprache unterschätzen.
- Einen Crash-Kurs in die MOTIS-typische C++-Programmierung gibt es hier: `https://www.algo.informatik.tu-darmstadt.de/dl/cpp.pdf`.
- MOTIS C++ Style Guide: `https://github.com/motis-project/motis/blob/master/docs/STYLE.md`

Literatur

[1] Sascha Witt. „Trip-Based Public Transit Routing“. In: *Algorithms - ESA 2015*. Hrsg. von Nikhil Bansal und Irene Finocchi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, S. 1025–1036. ISBN: 978-3-662-48350-3. DOI: https://doi.org/10.1007/978-3-662-48350-3_85.