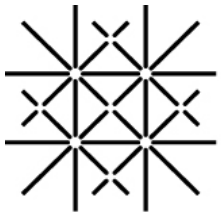
28. Juni 2016

Bachelorarbeit

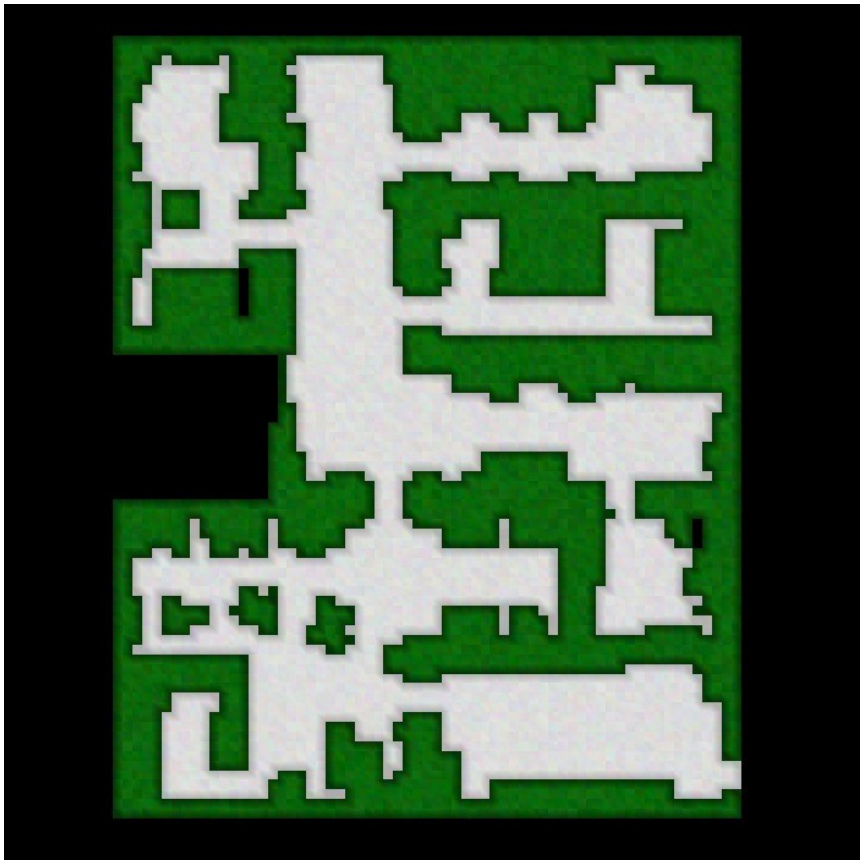
Komprimierte Pfaddatenbanken durch Lauflängenkodierung von optimal permutierten first-move Matrizen

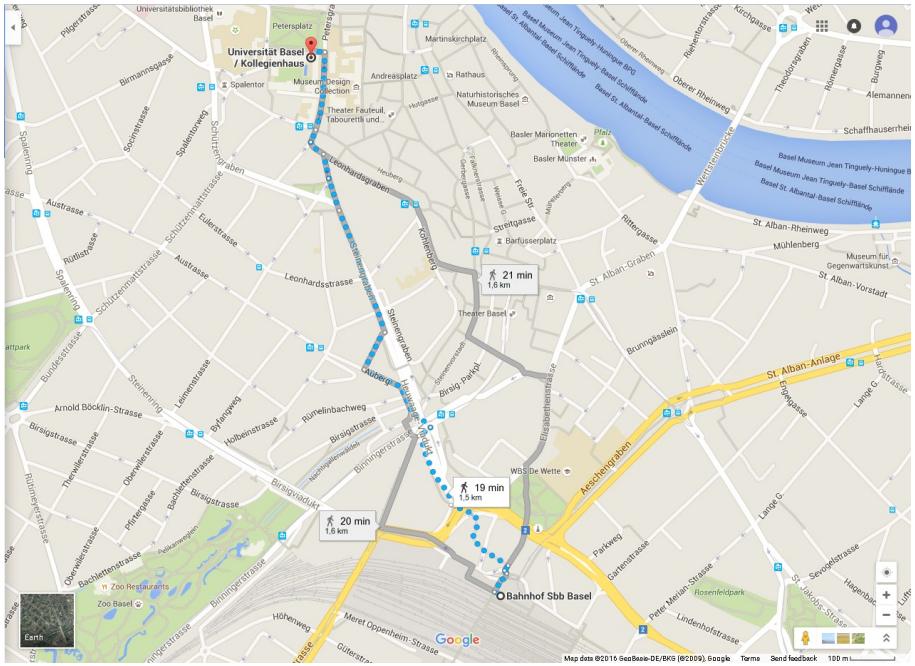
Patrick Zumsteg



University of Basel

Motivation




Universität Basel / Kollegienhaus
Bahnhof Sbb Basel
21 min
1.6 km
19 min
1.5 km
20 min
1.6 km
Google

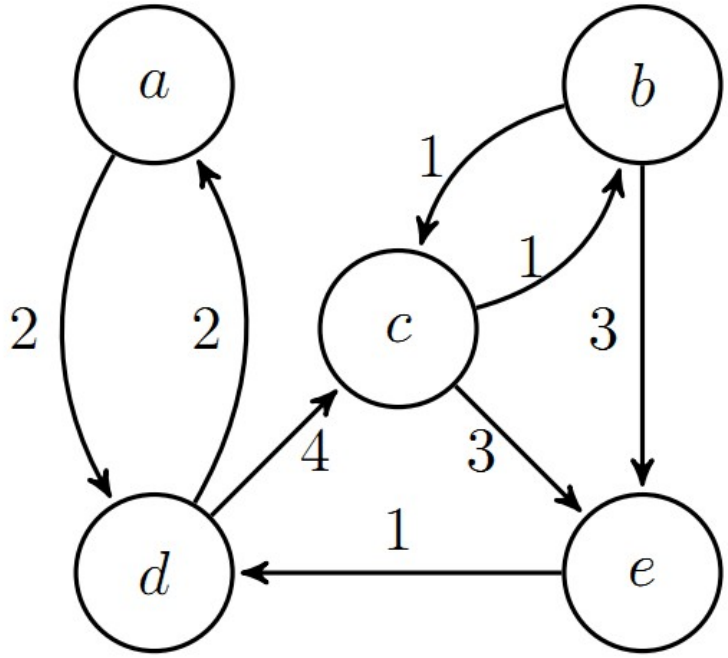
Alternative: CPD

- Optimale Pfade in Datenstruktur abgelegt
- Auslesen statt aufwändiger Berechnung
- Aber: hohe Speicherkosten
  - Kompression
- Optimale Kompression?

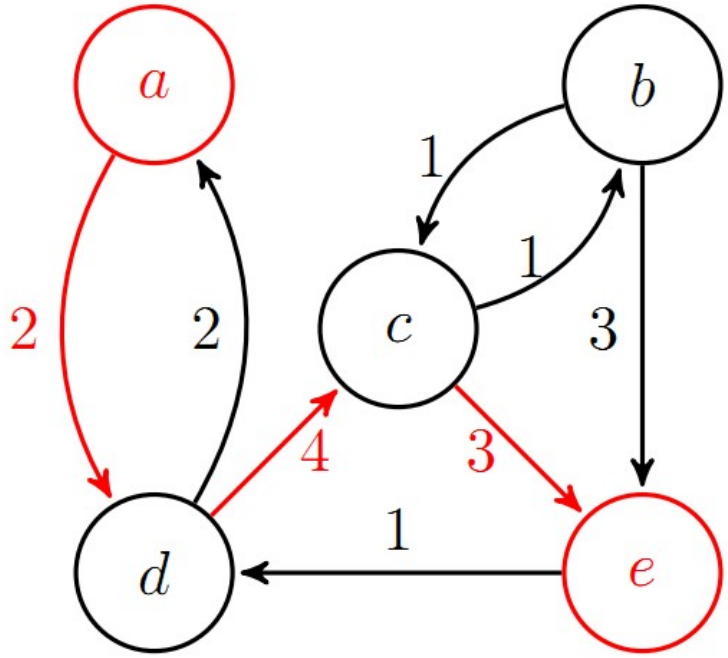
Inhalt

- Hintergrund
  - Graphen, Pfade und first-move Matrizen
- Lauflängenkodierung
- Optimale Spaltenordnung einer first-move Matrix
  - Metriken
- Resultate
- Ausblick

Graphen, Pfade und first moves

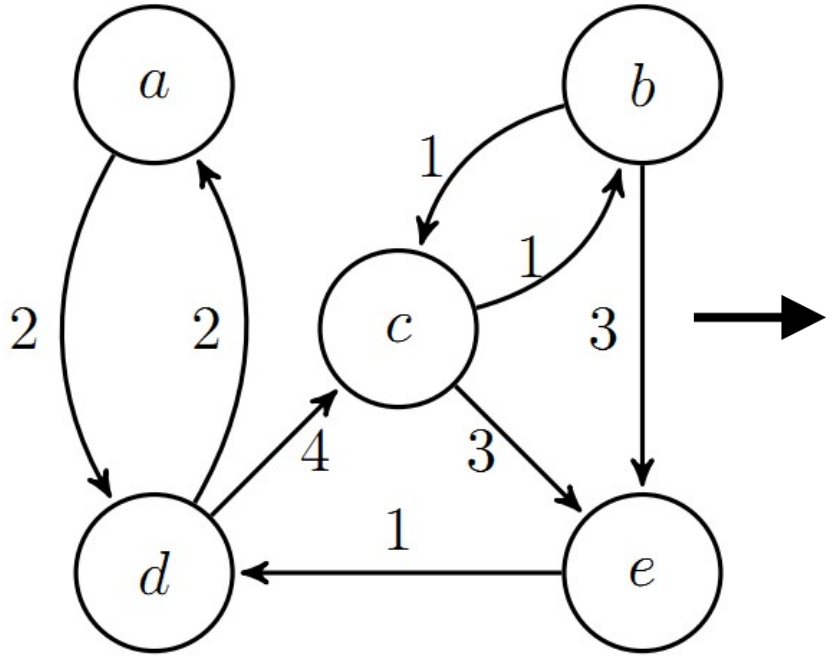


Graphen, Pfade und first moves



- Kürzester Pfad: <a,d>,<d,c>,<c,e>
- First move: <a,d>

First-move Matrix



	a	b	c	d	e
a	–	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$
b	$\langle b, c \rangle$	–	$\langle b, c \rangle$	$\langle b, e \rangle$	$\langle b, e \rangle$
c	$\langle c, e \rangle$	$\langle c, b \rangle$	–	$\langle c, e \rangle$	$\langle c, e \rangle$
d	$\langle d, a \rangle$	$\langle d, c \rangle$	$\langle d, c \rangle$	–	$\langle d, c \rangle$
e	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	–

Lauflängenkodierung

	a	b	c	d	e
a	–	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$
b	$\langle b, c \rangle$	–	$\langle b, c \rangle$	$\langle b, e \rangle$	$\langle b, e \rangle$
c	$\langle c, e \rangle$	$\langle c, b \rangle$	–	$\langle c, e \rangle$	$\langle c, e \rangle$
d	$\langle d, a \rangle$	$\langle d, c \rangle$	$\langle d, c \rangle$	–	$\langle d, c \rangle$
e	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	–

- Ersetze Läufe durch Tupel <Zeichen,Länge>

Lauflängenkodierung

	a	b	c	d	e
a	–	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$
b	$\langle b, c \rangle$	–	$\langle b, c \rangle$	$\langle b, e \rangle$	$\langle b, e \rangle$
c	$\langle c, e \rangle$	$\langle c, b \rangle$	–	$\langle c, e \rangle$	$\langle c, e \rangle$
d	$\langle d, a \rangle$	$\langle d, c \rangle$	$\langle d, c \rangle$	–	$\langle d, c \rangle$
e	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	–

$$\langle \langle b, c \rangle, 3 \rangle, \langle \langle b, e \rangle, 2 \rangle$$

Lauflängenkodierung

	a	b	c	d	e
a	–	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$
b	$\langle b, c \rangle$	–	$\langle b, c \rangle$	$\langle b, e \rangle$	$\langle b, e \rangle$
c	$\langle c, e \rangle$	$\langle c, b \rangle$	–	$\langle c, e \rangle$	$\langle c, e \rangle$
d	$\langle d, a \rangle$	$\langle d, c \rangle$	$\langle d, c \rangle$	–	$\langle d, c \rangle$
e	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	–

a	$\langle \langle a, d \rangle, 5 \rangle$
b	$\langle \langle b, c \rangle, 3 \rangle, \langle \langle b, e \rangle, 2 \rangle$
c	$\langle \langle c, e \rangle, 1 \rangle, \langle \langle c, b \rangle, 2 \rangle, \langle \langle c, e \rangle, 2 \rangle$
d	$\langle \langle d, a \rangle, 1 \rangle, \langle \langle d, c \rangle, 4 \rangle$
e	$\langle \langle e, d \rangle, 5 \rangle$

Vertauschen der Spalten

	c	b	a	d	e
a	$\langle a, d \rangle$	$\langle a, d \rangle$	–	$\langle a, d \rangle$	$\langle a, d \rangle$
b	$\langle b, c \rangle$	–	$\langle b, c \rangle$	$\langle b, e \rangle$	$\langle b, e \rangle$
c	–	$\langle c, b \rangle$	$\langle c, e \rangle$	$\langle c, e \rangle$	$\langle c, e \rangle$
d	$\langle d, c \rangle$	$\langle d, c \rangle$	$\langle d, a \rangle$	–	$\langle d, c \rangle$
e	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	–

- Spalten vertauschbar
- Permutationsvektor für Zugriffe

Vertauschen der Spalten

	c	b	a	d	e
a	$\langle a, d \rangle$	$\langle a, d \rangle$	–	$\langle a, d \rangle$	$\langle a, d \rangle$
b	$\langle b, c \rangle$	–	$\langle b, c \rangle$	$\langle b, e \rangle$	$\langle b, e \rangle$
c	–	$\langle c, b \rangle$	$\langle c, e \rangle$	$\langle c, e \rangle$	$\langle c, e \rangle$
d	$\langle d, c \rangle$	$\langle d, c \rangle$	$\langle d, a \rangle$	–	$\langle d, c \rangle$
e	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	–

→ $\langle \langle c, b \rangle, 2 \rangle, \langle \langle c, e \rangle, 3 \rangle$

→ $\langle \langle d, c \rangle, 2 \rangle, \langle \langle d, a \rangle, 2 \rangle, \langle \langle d, c \rangle, 1 \rangle$

Vertauschen der Spalten

- Nebeneinandersetzen von Spalten hat Kosten (unterbrochene Läufe)
- Ziel: Spaltenordnung, die möglichst wenig Läufe abbricht
- Hamming-Distanz: Zählt, wie viele Läufe abgebrochen werden
  - Optimal für einelementige Matrizen:
    - Neuer Lauf gdw. Elemente der Spalten ungleich

Hamming-Distanz

- Hamming-Distanz beträgt 2

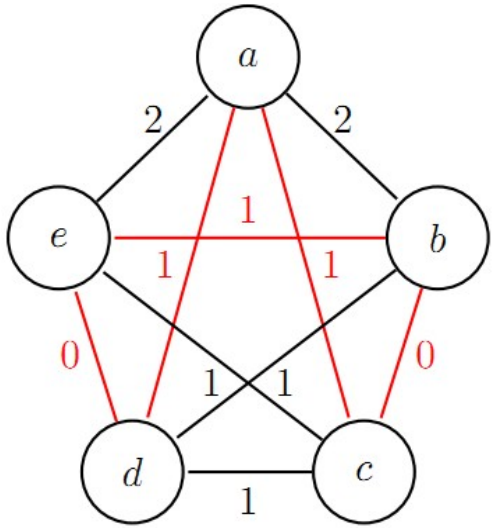
	a	b	c	d	e
a	–	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$
b	$\langle b, c \rangle$	–	$\langle b, c \rangle$	$\langle b, e \rangle$	$\langle b, e \rangle$
c	$\langle c, e \rangle$	$\langle c, b \rangle$	–	$\langle c, e \rangle$	$\langle c, e \rangle$
d	$\langle d, a \rangle$	$\langle d, c \rangle$	$\langle d, c \rangle$	–	$\langle d, c \rangle$
e	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	–

Hamming-Distanz

- Hamming-Distanz beträgt 2

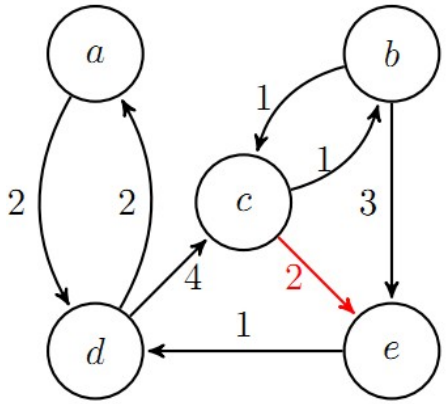
	a	b	c	d	e
a	–	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$
b	$\langle b, c \rangle$	–	$\langle b, c \rangle$	$\langle b, e \rangle$	$\langle b, e \rangle$
c	$\langle c, e \rangle$	$\langle c, b \rangle$	–	$\langle c, e \rangle$	$\langle c, e \rangle$
d	$\langle d, a \rangle$	$\langle d, c \rangle$	$\langle d, c \rangle$	–	$\langle d, c \rangle$
e	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	–

- Hamming-Distanzen als Kantengewichte
- Vollständiger Graph
- Beste Spaltenordnung: kürzeste Tour
  - Travelling Salesman Problem



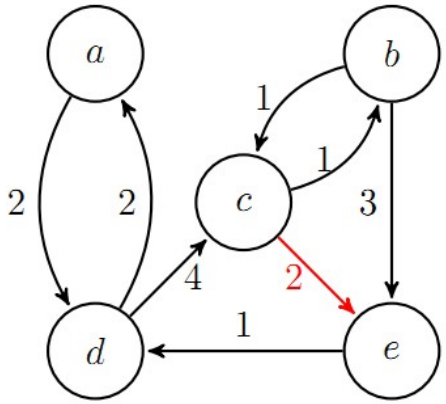
Matrizen mit mehrelementigen Einträgen

- Mehrere kürzeste Pfade/first Moves



Matrizen mit mehrelementigen Einträgen

- Mehrere kürzeste Pfade/first Moves
- Für RLE: Wähle first move, der Lauf nicht abbricht



	a	b	c	d	e
a	–	$\langle a, d\rangle$	$\langle a, d\rangle$	$\langle a, d\rangle$	$\langle a, d\rangle$
b	$\langle b, c\rangle$	–	$\langle b, c\rangle$	$\langle b, c\rangle, \langle b, e\rangle$	$\langle b, c\rangle, \langle b, e\rangle$
c	$\langle c, e\rangle$	$\langle c, b\rangle$	–	$\langle c, e\rangle$	$\langle c, e\rangle$
d	$\langle d, a\rangle$	$\langle d, c\rangle$	$\langle d, c\rangle$	–	$\langle d, c\rangle$
e	$\langle e, d\rangle$	$\langle e, d\rangle$	$\langle e, d\rangle$	$\langle e, d\rangle$	–

Matrizen mit mehrelementigen Einträgen

- Hamming-Distanz nicht mehr optimal
  - Soll abgebrochene Läufe widerspiegeln
- Hamming-Distanz=1, obwohl kein Lauf abgebrochen

  → H.-Distanz überschätzt Kosten
- Unterschätzt nicht:
  - Müsste möglichen Abbruch des Laufes ignorieren
  - Nicht möglich; Ungleiche Menge

    → Hamming-Distanz += 1

	a	b	c	d	e
a	–	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$
b	$\langle b, c \rangle$	–	$\langle b, c \rangle$	$\langle b, c \rangle, \langle b, e \rangle$	$\langle b, c \rangle, \langle b, e \rangle$
c	$\langle c, e \rangle$	$\langle c, b \rangle$	–	$\langle c, e \rangle$	$\langle c, e \rangle$
d	$\langle d, a \rangle$	$\langle d, c \rangle$	$\langle d, c \rangle$	–	$\langle d, c \rangle$
e	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	–

Disjunktheits-Distanz

- Erhöhe Distanz, wenn Elemente disjunkt sind

  → Nur wenn Lauf garantiert abbricht

- Unterschätzt Kosten

- Überschätzt nicht: Müsste Distanz erhöhen, obwohl Lauf möglich
  - Mengen wären dann nicht disjunkt

- Disjunktheits-Distanz = 0

	a	b	c	d	e
a	–	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$
b	$\langle b, c \rangle$	–	$\langle b, c \rangle$	$\langle b, c \rangle, \langle b, e \rangle$	$\langle b, c \rangle, \langle b, e \rangle$
c	$\langle c, e \rangle$	$\langle c, b \rangle$	–	$\langle c, e \rangle$	$\langle c, e \rangle$
d	$\langle d, a \rangle$	$\langle d, c \rangle$	$\langle d, c \rangle$	–	$\langle d, c \rangle$
e	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	–

Gewichtete Disjunktheits-Distanz

- Idee: Mehr Elemente nur in einer Menge
  → Grössere Chance auf Laufabbruch
- Zählt Kanten, die nur in einer Menge vorkommen (symmetrische Differenz)
- Überschätzt Kosten: Eine Menge ist Teilmenge der anderen
  → Erhöht Distanz, obwohl Lauf möglich
- Unterschätzt nicht: Immer, wenn Laufabbruch möglich, ist Distanz > 0

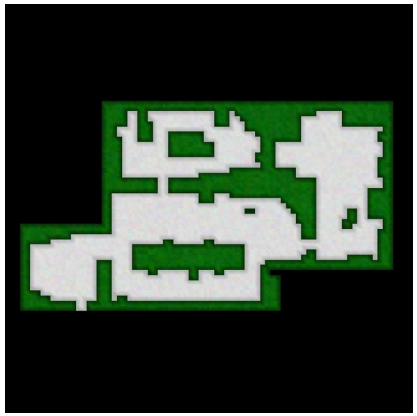
- Gew. Disjunktheits-Distanz = 1

	a	b	c	d	e
a	–	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$	$\langle a, d \rangle$
b	$\langle b, c \rangle$	–	$\langle b, c \rangle$	$\langle b, c \rangle, \langle b, e \rangle$	$\langle b, c \rangle, \langle b, e \rangle$
c	$\langle c, e \rangle$	$\langle c, b \rangle$	–	$\langle c, e \rangle$	$\langle c, e \rangle$
d	$\langle d, a \rangle$	$\langle d, c \rangle$	$\langle d, c \rangle$	–	$\langle d, c \rangle$
e	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	$\langle e, d \rangle$	–

Experimente

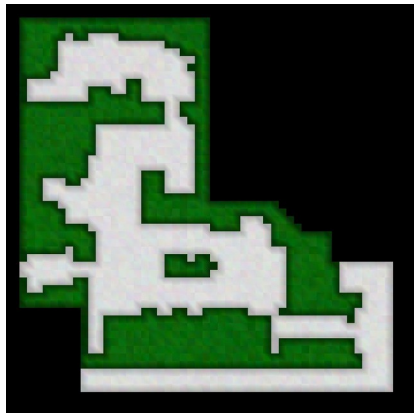
- Vergleich der Kompressionsqualität
- Anwendung auf Karten aus Dragon Age: Origins
- Implementation in C++
- TSP mit Concorde gelöst
- Referenzgrösse der CPD: Berechnung mit Strasser et. al.s Algorithmus

Ergebnisse



		Hamming-Distanz	Disjunktheit	gew. Disjunktheit
Rechenzeit	avg	350s	41s	152s
DistMasse	min	6.58806s	7.54519s	22.4139s
	avg	6.95229s	7.64552s	23.4894s
	max	7.17239s	7.74901s	24.0899s
Speicher	avg	25'740 kB	19'645 kB	22'600 kB
CPD	avg	116.6 kB	111.9 kB	115.2 kB

- 1360 Knoten
- Kompression auf 3.2%
- Referenzgrösse CPD: 137.4 kB



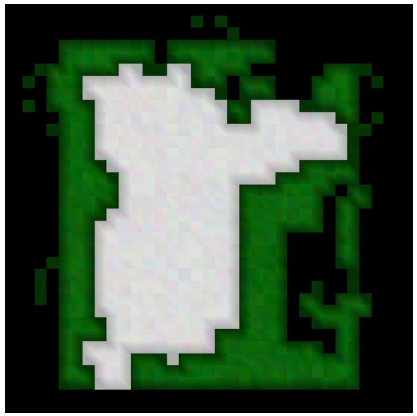
		Hamming-Distanz	Disjunktheit	gew. Disjunktheit
Rechenzeit	avg	7s	35s	30s
DistMasse	min	1.47375s	1.48498s	5.49935s
	avg	1.49710s	1.51262s	5.56296s
	max	1.52164s	1.52682s	5.60192s
Speicher	avg	13'457 kB	16'576 kB	19'809 kB
CPD	avg	58.4 kB	57.8 kB	58.3 kB

- 861 Knoten
- Kompression auf 4.1%
- Referenzgrösse CPD: 65.2 kB

Detailreiche Karten: gute Kompression

Unterschätzendes Distanzmass besser

Ergebnisse



		Hamming-Distanz	Disjunktheit	gew. Disjunktheit
Rechenzeit	avg	3s	1s	2s
DistMasse	min	0.070455s	0.066759s	0.249161s
	avg	0.071324s	0.066970s	0.250014s
	max	0.072058s	0.067297s	0.251333s
Speicher	avg	7'969 kB	4'910 kB	7'336 kB
CPD	avg	26.7 kB	27.6 kB	26.6 kB

- 318 Knoten
- Kompression auf 13.7%
- Referenzgrösse CPD: 30.6 kB
- Detailarme Karte: Überschätzende Distanzmasse besser
- Deutlich schlechtere Kompression

Ergebnisse

- Sehr hohe Laufzeit schon für kleine Karten
  - TSP-Solver benötigt am meisten Rechenzeit
  - Nur für kleine Karten anwendbar

- Besserer Kompressionsgrad als Referenz

- Unkomprimierte first-move Matrix im Arbeitsspeicher
  → Quadratischer Platzverbrauch, aber weniger restriktiv als Laufzeit

Ausblick

- Verringern der Laufzeit mit TSP-Heuristik
  - Trade-off zwischen Laufzeit und Optimalität
- Besseres Distanzmass?
- Reduzierter Speicherverbrauch?