

# Data Ordering Problem with Inversion

Manuel Braun    Jendrik Seipp

Institut für Informatik  
Albert-Ludwigs-Universität Freiburg

Informatiktage 2009

## Unser Team



- Manuel Braun, Jendrik Seipp
- B.Sc. Informatik, Uni Freiburg, 5. Semester

- Erklärung des DOPI
- Lösungen
  - Optimale Lösung
  - 2 Heuristiken
- Vergleich der Heuristiken
- Grafische Oberfläche

# Data Ordering Problem with Inversion (DOPI)

## Definition

### Datenbus

	←	k	-	→					
↑		0		0		1		1	
n		0		1		0		1	
		1		0		1		1	
↓		0		0		1		0	

- Übertragung auf einem Datenbus
- $n$  Wörter der Länge  $k$
- Minimiere Anzahl der Transitionen

# Data Ordering Problem with Inversion (DOPI)

## Definition

### Datenbus

	←	k	-	→	INV
↑	0	0	1	1	0/1
n	0	1	0	1	0/1
	1	0	1	1	0/1
↓	0	0	1	0	0/1

Hintergrund:

- Transitionen kosten Strom
- Umordnen verringert Anzahl der Transitionen

## Umordnen

0	0	1	1
0	1	0	1
1	0	1	1
0	0	1	0

## Umordnen

0	0	1	1
0	1	0	1
1	0	1	1
0	0	1	0

2

## Umordnen

0	0	1	1	
0	1	0	1	2
1	0	1	1	3
0	0	1	0	

## Umordnen

0	0	1	1	
0	1	0	1	2
1	0	1	1	3
0	0	1	0	2

## Umordnen

0	0	1	1	
0	1	0	1	2
1	0	1	1	3
0	0	1	0	2
$\Sigma$ 7				

## Umordnen

0	0	1	1			
0	1	0	1	↓	↖	2
1	0	1	1			3
0	0	1	0	↓	↑	2
						$\Sigma$ 7

## Umgeordnet

0	0	1	1
0	0	1	0
1	0	1	1
0	1	0	1

## Umgeordnet

0	0	1	1	
0	0	1	0	1
1	0	1	1	
0	1	0	1	

## Umgeordnet

0	0	1	1	
0	0	1	0	1
1	0	1	1	2
0	1	0	1	

## Umgeordnet

0	0	1	1	
0	0	1	0	1
1	0	1	1	2
0	1	0	1	3

## Umgeordnet

0	0	1	1	
0	0	1	0	1
1	0	1	1	2
0	1	0	1	3
<hr/>				
$\Sigma$ 6				

## Invertieren

					INV
	0	0	1	1	0
	0	0	1	0	0
	1	0	1	1	0
	0	1	0	1	0

- Invertiert:  $0 \rightarrow 1, 1 \rightarrow 0$
- Transitionen auf dem Invertierungskanal zählen mit

## Invertieren

				INV
0	0	1	1	0
0	0	1	0	0
1	0	1	1	0
1	0	1	0	1

- Invertiert:  $0 \rightarrow 1, 1 \rightarrow 0$
- Transitionen auf dem Invertierungskanal zählen mit

## Invertieren

				INV	
0	0	1	1	0	
0	0	1	0	0	1
1	0	1	1	0	2
1	0	1	0	1	2
					$\Sigma$ 5

- Invertiert:  $0 \rightarrow 1, 1 \rightarrow 0$
- Transitionen auf dem Invertierungskanal zählen mit

- Naiv: probiere für jede Anordnung der Wörter alle Invertierungen
- kleine Verbesserung:
  - Ignoriere inverse Anordnungen
    - $1,2,3,4 = 4,3,2,1$
  - Ignoriere inverse Invertierungen
    - $1,2,3,\bar{4} = \bar{1}, \bar{2}, \bar{3}, 4$

- Naiv: probiere für jede Anordnung der Wörter alle Invertierungen
- kleine Verbesserung:
  - Ignoriere inverse Anordnungen
    - $1,2,3,4 = 4,3,2,1$
  - Ignoriere inverse Invertierungen
    - $1,2,3,\bar{4} = \bar{1}, \bar{2}, \bar{3}, 4$

## Inverse Invertierungen

	INV		INV
$\frac{1 \ 0 \ 1 \ 1}{1 \ 0 \ 1 \ 0}$	0	$\frac{0 \ 1 \ 0 \ 0}{0 \ 1 \ 0 \ 1}$	1
$\Sigma \ 2$		$\Sigma \ 2$	0

- Heuristik: schnelle, approximierte Lösung

INV

Unsortiert: 

0	0	1	1
0	1	0	1
1	0	1	1
0	0	1	0

Sortiert: \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

## Algorithmus

- Nehme erstes Wort

- Heuristik: schnelle, approximierte Lösung

Unsortiert:	<hr/> <hr/> <hr/> <hr/> <hr/>	Sortiert:	<hr/> <hr/> <hr/> <hr/> <hr/>	INV
	0 1 0 1		0 0 1 1	0
	1 0 1 1			
	0 0 1 0			

## Algorithmus

- Nehme erstes Wort
- Nehme ähnlichstes Wort

- Heuristik: schnelle, approximierte Lösung

Unsortiert:	<hr/> <hr/> <hr/> <hr/>	Sortiert:	<hr/> <hr/> <hr/> <hr/>	INV
	0 1 0 1		0 0 1 1	0
	0 0 1 0		1 0 1 1	0

## Algorithmus

- Nehme erstes Wort
- Nehme ähnlichstes Wort
- Hänge Wort vorne oder hinten an

- Heuristik: schnelle, approximierte Lösung

Unsortiert:	<hr/>	Sortiert:	<hr/>	INV
	0 1 0 1		0 0 1 0	0
	<hr/>		0 0 1 1	0
	<hr/>		1 0 1 1	0
	<hr/>		<hr/>	

## Algorithmus

- Nehme erstes Wort
- Nehme ähnlichstes Wort
- Hänge Wort vorne oder hinten an
- Invertiere wenn nötig

- Heuristik: schnelle, approximierte Lösung

Unsortiert:	_____	Sortiert:	<u>0 0 1 0</u>	INV
	_____		<u>0 0 1 1</u>	0
	_____		<u>1 0 1 1</u>	0
	_____		<u>1 0 1 0</u>	1

## Algorithmus

- Nehme erstes Wort
- Nehme ähnlichstes Wort
- Hänge Wort vorne oder hinten an
- Invertiere wenn nötig

- Ruft die Basisheuristik  $n$ -mal auf
- Jedes Wort ist einmal Startwort
- Wähle Sortierung mit kleinster Transitionszahl

- Ruft die Basisheuristik n-mal auf
- Jedes Wort ist einmal Startwort
- Wähle Sortierung mit kleinster Transitionszahl

## n Sortierungen

2x				INV	INV				INV	INV				INV
0	0	1	0	0	1	0	1	1	0	0	1	0	1	0
0	0	1	1	0	0	0	1	1	0	1	0	1	1	0
1	0	1	1	0	0	0	1	0	0	0	0	1	1	0
0	1	0	1	1	0	1	0	1	1	0	0	1	0	0
$\Sigma$		4			$\Sigma$		4			$\Sigma$		5		

# Heuristik sortByDistValues

- Sortiert die Wörter vor
- Wendet Basisheuristik auf vorsortierte Wörter an

## Transitionen

	0011	0101	1011	0010
0011	-	2	1	1
0101	2	-	2	2
1011	1	2	-	2
0010	1	2	2	-

$\Sigma$	4	6	5	5
----------	---	---	---	---

## Vorsortierung

Wort	Distanz
0011	4
0101	6
1011	5
0010	5

→

Vorsortierte Wörter	Distanz
0101	6
1011	5
0010	5
0011	4

- SortNTimes:
  - Sequentielle und parallele Version
  - Parallele Version viel schneller
- Ausweitung der Parallelität
  - *Beide Heuristiken*: Berechne Distanzen parallel
  - *Basisheuristik*: Finde beste Nachbarn parallel
  - *SortByDistValues*: Sortiere parallel

- 2500 zufällige Wortlisten
- $4 \leq n \leq 100$ ,  $4 \leq k \leq 200$

## Transitionen

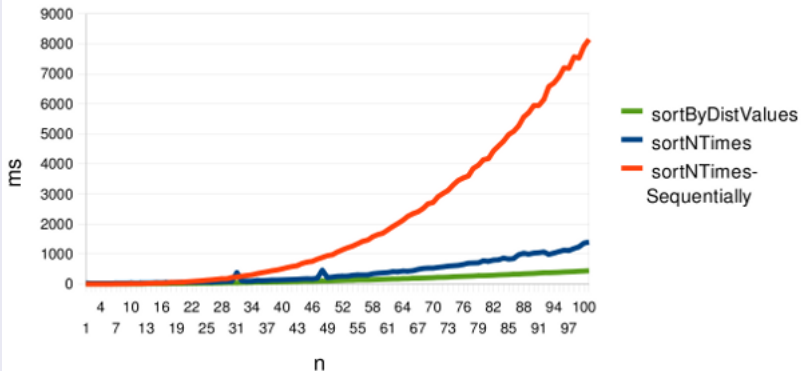
	Absolut	Relativ
unsortiert	6.742.540	100 %
sortNTimes	5.382.094	79,8 %
sortByDistValues	5.396.744	80,0 %

### Landau Notation

- Distanzen:  $O(n^2 \cdot k)$
- Basisheuristik:  $O(n^2)$
- sortNTimes:  $O(n^2 \cdot k) + O(n^3)$
- sortByDistValues:  $O(n^2 \cdot k) + O(n^2)$

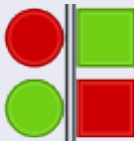
## Empirisch

### Laufzeiten



- Python
- GTK
- Tests (Laufzeit, Transitionen)

## Legende



- Rot: 0-Bit
- Grün: 1-Bit
- Boxen: normale Bits
- Kreise: Inversionsbits

- Sehr gute Lösungen mit `sortByDistValues` und `sortNTimes`
- `sortNTimes` fast immer besser, aber langsamer
- `sortByDistValues` immer schneller, manchmal sogar weniger Transitionen