Computing the Flip Distance of Triangulations
Bachelor-Kolloquium

Fabian Lipp

July 5, 2012
Contents

1 Introduction
   • Motivation of the Problem
   • Definitions

2 Exact Algorithms
   • Breadth-First Search
   • Improvements
   • Effects of the Improvements

3 Heuristics
   • Variants
   • Comparing to other Heuristics

4 Conclusion
Motivation of the Problem

- Let $T_1, T_2$ be binary search trees
- Operations: Left rotation, right rotation

![Diagram of two binary search trees comparing operations](image)
Motivation of the Problem

- Let $T_1, T_2$ be binary search trees
- Operations: Left rotation, right rotation

Task: Find a shortest sequence of rotations that lead from $T_1$ to $T_2$
Motivation of the Problem

- Let $T_1, T_2$ be binary search trees
- Operations: Left rotation, right rotation

Task: Find a shortest sequence of rotations that lead from $T_1$ to $T_2$

There is no polynomial-time algorithm known for this problem, but NP-hardness is not proven
Transformation of the Problem

Represent a binary tree with \( n - 2 \) nodes as a triangulation of a polygon with \( n \) corners
Transformation of the Problem

Represent a binary tree with \( n - 2 \) nodes as a triangulation of a polygon with \( n \) corners

![Binary Tree Diagram]
Transformation of the Problem

Represent a binary tree with $n - 2$ nodes as a triangulation of a polygon with $n$ corners.
Transformation of the Problem

Represent a binary tree with $n - 2$ nodes as a triangulation of a polygon with $n$ corners.
Transformation of the Problem

Represent a binary tree with \( n - 2 \) nodes as a triangulation of a polygon with \( n \) corners
Transformation of the Problem

Represent a binary tree with \( n - 2 \) nodes as a triangulation of a polygon with \( n \) corners.
Represent a binary tree with \( n - 2 \) nodes as a triangulation of a polygon with \( n \) corners.
Transformation of the Problem

Represent a binary tree with \( n - 2 \) nodes as a triangulation of a polygon with \( n \) corners.
Transformation of the Problem

Represent a binary tree with \( n - 2 \) nodes as a triangulation of a polygon with \( n \) corners.
Transformation of the Problem

Represent a binary tree with $n - 2$ nodes as a triangulation of a polygon with $n$ corners.
Equivalent of Rotations

Fabian Lipp
Computing the Flip Distance of Triangulations
Equivalent of Rotations

Diagrams of equivalent rotations.
Equivalent of Rotations
Rotation distance between binary trees is equivalent to flip distance between triangulations.
Flip Distance

Rotation distance between binary trees is equivalent to flip distance between triangulations

Definition (Flip Distance)

The *flip distance* \( \text{fd}(\pi_1, \pi_2) \) is the minimum number of diagonal flips required to transform \( \pi_1 \) to \( \pi_2 \).
Rotation distance between binary trees is equivalent to flip distance between triangulations.

**Definition (Flip Distance)**

The *flip distance* \( \text{fd}(\pi_1, \pi_2) \) is the minimum number of diagonal flips required to transform \( \pi_1 \) to \( \pi_2 \).

Upper bound for two triangulations of the \( n \)-gon:

\[
\text{fd}(\pi_1, \pi_2) \leq 2n - 6
\]
Breadth-First Search

Flip Graph $F_6$:

```
fd(start, target) = 3
```

Runtime: Number of nodes in $F_n + 2$:

$C_n = 1 + 2^n (2^n)$

$C_n \sim 4 \sqrt{n} \pi n^3 \Rightarrow$ Exponential runtime
Breadth-First Search

Flip Graph $\mathcal{F}_6$:
Breadth-First Search

Flip Graph $\mathcal{F}_6$:

Runtime

Number of nodes in $F_n + 2$

$C_n = \frac{n}{2} \cdot (2n - n^2)$

$C_n \sim 4n\sqrt{\pi n^3}$

⇒ Exponential runtime
Breadth-First Search

Flip Graph $F_6$: 

$$\text{flip}(\text{start}, \text{target}) = 3$$

Runtime: Number of nodes in $F_n + 2$

$$C_n = 1 + 2^n (2^n)$$

$$C_n \sim 4n^{3/2} \pi$$

$\Rightarrow$ Exponential runtime
Breadth-First Search

Flip Graph $\mathcal{F}_6$:
Breadth-First Search

Flip Graph $\mathcal{F}_6$: 

$\text{flip}(\text{start}, \text{target}) = 3$

Runtime

$C_n = n + 2(2n^n) 
\sim 4n^{3/2}/\sqrt{\pi} 
\Rightarrow \text{Exponential runtime}$
Breadth-First Search

Flip Graph $\mathcal{F}_6$: 

$$\text{runtime} = \log_2 \left( 2^n + 2 \right) \approx 4n \sqrt{\frac{\pi}{n^3}} \Rightarrow \text{exponential runtime}$$
Breadth-First Search

Flip Graph $\mathcal{F}_6$:

$$fd(start, target) = 3$$
Breadth-First Search

Flip Graph $\mathcal{F}_6$:

$\text{fd}(\text{start}, \text{target}) = 3$

Runtime

Number of nodes in $\mathcal{F}_{n+2}$:

$$C_n = \frac{1}{n+2} \binom{2n}{n}$$
Breadth-First Search

Flip Graph $F_6$: 

$$
\text{fd}(\text{start}, \text{target}) = 3
$$

Runtime 

Number of nodes in $F_{n+2}$: 

$$
C_n = \frac{1}{n+2} \binom{2n}{n}
$$

$$
C_n \sim \frac{4^n}{\sqrt{\pi n^3}}
$$

$\Rightarrow$ Exponential runtime
Simple Improvements

Common Diagonal

Do not flip this diagonal in BFS
Simple Improvements

Common Diagonal
Do not flip this diagonal in BFS

Flip-to-Match Diagonal
Flip only this diagonal in BFS
Two-Sided BFS

- Start BFS from start and target at the same time
- Reduces the number of visited nodes
Two-Sided BFS

Start BFS from start and target at the same time. Reduces the number of visited nodes.
Two-Sided BFS

Start BFS from start and target at the same time

Reduces the number of visited nodes
Two-Sided BFS

Start BFS from start and target at the same time

This reduces the number of visited nodes.
Two-Sided BFS

Start BFS from start and target at the same time

Reduces the number of visited nodes
Two-Sided BFS

Start BFS from start and target at the same time

Reduces the number of visited nodes
Two-Sided BFS

Start BFS from start and target at the same time.

Reduces the number of visited nodes.
Two-Sided BFS

- Start BFS from start and target at the same time
Two-Sided BFS

- Start BFS from start and target at the same time

Reduces the number of visited nodes
Two-Sided BFS

- Start BFS from start and target at the same time

Start BFS from start and target at the same time
Two-Sided BFS

- Start BFS from start and target at the same time

Start BFS from start and target at the same time
Start BFS from start and target at the same time
Two-Sided BFS

Start BFS from start and target at the same time

- Start BFS from start and target at the same time
Two-Sided BFS

Start BFS from start and target at the same time
Start BFS from start and target at the same time
Reduces the number of visited nodes
A* search algorithm

Reduces the number of visited nodes

Needs lower bound for distance from an arbitrary node to target

Easy lower bound: number of diagonals that are not common
A* search algorithm

Reduces the number of visited nodes

Needs lower bound for distance from an arbitrary node to target

Easy lower bound: number of diagonals that are not common
A* search algorithm

- Reduces the number of visited nodes
- Needs lower bound for distance from an arbitrary node to target
- Easy lower bound: number of diagonals that are not common
A* search algorithm

Reduces the number of visited nodes

Needs lower bound for distance from an arbitrary node to target

Easy lower bound: number of diagonals that are not common
A* search algorithm

Reduces the number of visited nodes

Needs lower bound for distance from an arbitrary node to target

Easy lower bound: number of diagonals that are not common
A* search algorithm

Reduces the number of visited nodes

Needs lower bound for distance from an arbitrary node to target

Easy lower bound: number of diagonals that are not common
A* search algorithm

Reduces the number of visited nodes

Needs lower bound for distance from an arbitrary node to target

Easy lower bound: number of diagonals that are not common
A* search algorithm

- Reduces the number of visited nodes
A* search algorithm

- Reduces the number of visited nodes
- Needs lower bound for distance from an arbitrary node to target
A* search algorithm

- Reduces the number of visited nodes
- Needs lower bound for distance from an arbitrary node to target
- Easy lower bound: number of diagonals that are not common
Effects of these improvements

The graph shows the number of visited nodes for different algorithms as the parameter $n$ increases. The algorithms compared are Naive BFS, Two-sided BFS, and A* algorithm. The x-axis represents the value of $n$, while the y-axis represents the number of visited nodes.
Highest Difference
Highest Difference

Optimal flip sequence

Fabian Lipp
Computing the Flip Distance of Triangulations 11 / 19
HighestDifference

Optimal flip sequence

Fabian Lipp
Computing the Flip Distance of Triangulations 11 / 19
HighestDifference

Optimal flip sequence
Highest Difference

Optimal flip sequence
HighestDifference

Optimal flip sequence

Fabian Lipp

Computing the Flip Distance of Triangulations
Highest Difference

Optimal flip sequence
HighestDifference

Optimal flip sequence
HighestDifference
HighestDifference

Optimal flip sequence

Fabian Lipp
Computing the Flip Distance of Triangulations
HighestDifference
HighestDifference

Optimal flip sequence
HighestDifference

Diagram showing triangulations and flip operations.
HighestDifference

Optimal flip sequence

Fabian Lipp
Computing the Flip Distance of Triangulations
HighestDifference

Optimal flip sequence
HighestDifference

Optimal flip sequence

Fabian Lipp
Computing the Flip Distance of Triangulations 11 / 19
HighestDifference

Optimal flip sequence
HighestDifference

Optimal flip sequence
HighestDifference

Optimal flip sequence
HighestDifference

Optimal flip sequence

Fabian Lipp
HighestDifference

Optimal flip sequence
Highest Difference

Optimal flip sequence

Fabian Lipp

Computing the Flip Distance of Triangulations
Highest Difference

Optimal flip sequence

Fabian Lipp
Computing the Flip Distance of Triangulations
HighestDifference

Optimal flip sequence

Fabian Lipp
HighestDifference

Optimal flip sequence
HighestDifference

Optimal flip sequence
HighestDifference

Optimal flip sequence
Highest Difference

Optimal flip sequence
Highest Difference

Optimal flip sequence
Highest Difference

Optimal flip sequence
HighestDifference

Optimal flip sequence
Highest Difference

Optimal flip sequence
Variants of the Heuristic

**Input:** start triangulation $s$, target triangulation $t$

**Output:** short flip sequence from $s$ to $t$
Variants of the Heuristic

**Input:** start triangulation $s$, target triangulation $t$

**Output:** short flip sequence from $s$ to $t$

### Variants

- **MostIntersections** Choose diagonal with maximum
  $$\#(\text{Intersections before flipping})$$

- **FewestIntersectionsAfter** Choose diagonal with minimum
  $$\#(\text{Intersections after flipping})$$

- **HighestDifference** Choose diagonal with maximum
  $$\Delta := \#(\text{Intersections before flipping}) - \#(\text{Intersections after flipping})$$

Runtime: $O(n^2)$
Variants of the Heuristic

**Input:** start triangulation \( s \), target triangulation \( t \)

**Output:** short flip sequence from \( s \) to \( t \)

**Variants**

- **MostIntersections**  Choose diagonal with maximum 
  \#(Intersections before flipping)

- **FewestIntersectionsAfter**  Choose Diagonal with minimum 
  \#(Intersections after flipping)
## Variants of the Heuristic

**Input:** start triangulation $s$, target triangulation $t$  
**Output:** short flip sequence from $s$ to $t$

### Variants

<table>
<thead>
<tr>
<th>Variant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MostIntersections</td>
<td>Choose diagonal with maximum #(Intersections before flipping)</td>
</tr>
<tr>
<td>FewestIntersectionsAfter</td>
<td>Choose Diagonal with minimum #(Intersections after flipping)</td>
</tr>
<tr>
<td>HighestDifference</td>
<td>Choose diagonal with maximum $\Delta := #(\text{Intersections before flipping}) – #(\text{Intersections after flipping})$.</td>
</tr>
</tbody>
</table>
Variants of the Heuristic

**Input:** start triangulation \( s \), target triangulation \( t \)

**Output:** short flip sequence from \( s \) to \( t \)

### Variants

- **MostIntersections** Choose diagonal with maximum
  \(#(\text{Intersections before flipping})\)

- **FewestIntersectionsAfter** Choose Diagonal with minimum
  \(#(\text{Intersections after flipping})\)

- **HighestDifference** Choose diagonal with maximum
  \(\Delta := #(\text{Intersections before flipping}) - #(\text{Intersections after flipping})\).

**Runtime:** \( O(n^2) \)
Comparing the Variants

![Graph showing the comparison of different variants. The x-axis represents the number of intersections (n), and the y-axis represents the percentage. The graph includes lines for HighestDifference, FewestIntersectionsAfter, and MostIntersections.]
First Heuristic by Pallo (2000):
- Works on weight sequences representing binary trees
- Constructs a lattice containing the trees
- Length of path in lattice is upper bound for rotation distance
Heuristics of Baril and Pallo

First Heuristic by Pallo (2000):
- Works on weight sequences representing binary trees
- Constructs a lattice containing the trees
- Length of path in lattice is upper bound for rotation distance

Heuristic by Baril and Pallo (2006):
- Works on triangulations
- Local transformation to create an additional common edge
- Provides Upper and Lower Bound for the Flip Distance
Heuristics of Baril and Pallo

- **First Heuristic by Pallo (2000):**
  - Works on weight sequences representing binary trees
  - Constructs a lattice containing the trees
  - Length of path in lattice is upper bound for rotation distance

- **Heuristic by Baril and Pallo (2006):**
  - Works on triangulations
  - Local transformation to create an additional common edge
  - Provides Upper and Lower Bound for the Flip Distance

- **Runtime:** $O(n^3)$
Comparing to other Heuristics

$n = 50$

<table>
<thead>
<tr>
<th>Flip Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance</td>
</tr>
</tbody>
</table>

- Trivial Upper Bound
- PalloUp
- BarilPalloUp
- BarilPalloLow
- HighestDifference
- Trivial Lower Bound
Comparing to other Heuristics

\( n = 500 \)

Instance

Flip Distance

0 20 40 60 80 100 120 140 160 180 200

Instance

Flip Distance

- **Trivial Upper Bound**
- **PalloUp**
- **BarilPalloUp**
- **HighestDifference**
- **BarilPalloLow**
- **Trivial Lower Bound**
Comparing to other Heuristics

The diagram illustrates the percentage of highest difference for different heuristics as a function of the size of the input, denoted by $n$. The heuristics compared are:

- HighestDifference
- BarilPalloUp
- PalloUp

The x-axis represents the size $n$, while the y-axis shows the percentage.
Conclusion

Summary

- Exact Algorithms: Improvements to BFS
- Heuristics: HighestDifference
  - Efficient compared to other heuristics
  - Easy to implement

Open Problems

- Polynomial-time algorithm
- NP-hardness proof
- Algorithms with approximation ratio < 2
