On Unfolding Trees and Polygons on Various Lattices

Author: Sheung-Hung Poon,
Presented by: Elena Mumford

Technical University of Eindhoven (TU/e),
The Netherlands

The 19th Canadian Conference on Computational Geometry (CCCG), August 20-22, 2007
Rigid edges can rotate around vertex w/o crossing other edges.

“Straightening” = move rigid edges to lie on a straight line.

“Convexifying” = move rigid edges until polygon becomes convex.

“locking” = cannot be straightened or convexified.
Introduction

- Rigid edges can rotate around vertex w/o crossing other edges.
- "Straightening" = move rigid edges to lie on a straight line.
- "Convexifying" = move rigid edges until polygon becomes convex.
- "locking" = cannot be straightened or convexified.
Rigid edges can rotate around vertex w/o crossing other edges.

“Straightening” = move rigid edges to lie on a straight line.

“Convexifying” = move rigid edges until polygon becomes convex.

“locking” = cannot be straightened or convexified.
Rigid edges can rotate around vertex w/o crossing other edges.

“Straightening” = move rigid edges to lie on a straight line.

“Convexifying” = move rigid edges until polygon becomes convex.

“locking” = cannot be straightened or convexified.
Applications:

- movements of robot arms,
- molecular conformation,
- wire bending,
- rigidity & knot theory,
Applications:
- movements of robot arms,
- molecular conformation,
- wire bending,
- rigidity & knot theory,
Applications:

- movements of robot arms,

- molecular conformation,

- wire bending,

- rigidity & knot theory,

(Borrowed from Univ. College London)
Applications:

- movements of robot arms,
- molecular conformation,
- wire bending,
- rigidity & knot theory,

(Borrowed from Univ. College London)
Applications:
- movements of robot arms,
- molecular conformation,
- wire bending,
- rigidity & knot theory,
Applications:
- movements of robot arms,
- molecular conformation,
- wire bending,
- rigidity & knot theory,

(Borrowed from Univ. College London)
Applications:
- movements of robot arms,
- molecular conformation,
- wire bending,
- rigidity & knot theory,

(Borrowed from Univ. College London)
In this talk, we mainly consider lattice polygons/trees.

- A \textit{unit polygon/tree} = with all its edges of unit-length.

- A \textit{lattice polygon/tree} = with all its edges from a lattice.

Square lattice tree

Hexagonal lattice tree

Triangular lattice tree
In this talk, we mainly consider lattice polygons/trees.

- A *unit polygon/tree* = with all its edges of unit-length.

- A *lattice polygon/tree* = with all its edges from a lattice.

Square lattice tree  Hexagonal lattice tree  Triangular lattice tree
Previous Work

- **Carpenter’s Rule Conjecture** (solved):
  Chains (polygons) in 2D can be straightened (convexified).
  [Connelly, Demaine and Rote ’00][Strienu ’00]

- Trees (polygons) in 4D+ can be straightened (convexified).
  [Cocan and O’Rourke ’01]

- A tree in 2D & a 5-chain in 3D can lock.  [Biedl et al ’01]

(a) A locked tree in 2D.  
(b) A locked 5-chain in 3D.
Previous Work

- **Carpenter’s Rule Conjecture** (solved):
  Chains (polygons) in 2D can be straightened (convexified).
  [Connelly, Demaine and Rote ’00][Strienu ’00]

- Trees (polygons) in 4D+ can be straightened (convexified).
  [Cocan and O’Rourke ’01]

- A tree in 2D & a 5-chain in 3D can lock. [Biedl et al ’01]

(a) A locked tree in 2D.  
(b) A locked 5-chain in 3D
Previous Work

- **Carpenter’s Rule Conjecture** (solved):
  Chains (polygons) in 2D can be straightened (convexified).
  [Connelly, Demaine and Rote ’00][Strienu ’00]

- Trees (polygons) in 4D+ can be straightened (convexified).
  [Cocan and O’Rourke ’01]

- A tree in 2D & a 5-chain in 3D can lock. [Biedl et al ’01]

(a) A locked tree in 2D.  
(b) A locked 5-chain in 3D
- PSPACE-complete: to reconfigure 2D-trees or 3D-chains. [Alt et al. ’03]

- A unit tree of diameter 4 can always be straightened. [Poon ’05]

- A 2D/3D lattice tree can always be straightened, and A 2D lattice polygon can always be convexified. [Poon ’06]
• PSPACE-complete: to reconfigure 2D-trees or 3D-chains. [Alt et al. ’03]

• A unit tree of diameter 4 can always be straightened. [Poon ’05]

• A 2D/3D lattice tree can always be straightened, and A 2D lattice polygon can always be convexified. [Poon ’06]
PSPACE-complete: to reconfigure 2D-trees or 3D-chains.  
[Alt et al. ’03]

A unit tree of diameter 4 can always be straightened.  
[Poon ’05]

A 2D/3D lattice tree can always be straightened, and  
A 2D lattice polygon can always be convexified.  
[Poon ’06]
A “move” = a monotonic increase/decrease of angle at a vertex.

Theorem

A hexagonal/triangular lattice chain can be straightened in $O(n)$ moves and time ($n =$ no. of edges).

Theorem

A hexagonal/triangular lattice tree can be straightened in $O(n^2)$ moves and time.

Theorem

A hexagonal/triangular lattice polygon can be convexified in $O(n^2)$ moves and time.
A “move” = a monotonic increase/decrease of angle at a vertex.

**Theorem**

A hexagonal/triangular lattice chain can be straightened in $O(n)$ moves and time ($n$ = no. of edges).

**Theorem**

A hexagonal/triangular lattice tree can be straightened in $O(n^2)$ moves and time.

**Theorem**

A hexagonal/triangular lattice polygon can be convexified in $O(n^2)$ moves and time.
A “move” = a monotonic increase/decrease of angle at a vertex.

**Theorem**

- A hexagonal/triangular lattice chain can be straightened in $O(n)$ moves and time ($n =$ no. of edges).

**Theorem**

- A hexagonal/triangular lattice tree can be straightened in $O(n^2)$ moves and time.

**Theorem**

- A hexagonal/triangular lattice polygon can be convexified in $O(n^2)$ moves and time.
Algorithm:
- Fold up the end edges of the chain iteratively.
- Unfold the final folded spring/zig-zag path.

PS. The algorithm is similar to that for *square lattice*.
Given a hexagonal lattice tree $P$.
Let $r$ be root = the leftmost vertex of $P$.

**Algorithm:**

Our algorithm proceeds by pulling $P$ to the left successively until the whole tree is straightened.

In each pulling step:

- Each vertex $v$ is pulled along its edge connecting to its parent;
- The motion of $v$ stops when $v$ is coincident with its parent in the previous step.
In each pulling step:

- Each vertex \( v \) is pulled along its edge connecting to its parent;
- The motion of \( v \) stops when \( v \) is coincident with its parent in the previous step.
The pulling step is repeated $n$ times so that, finally, tree $P$ is straightened.

Each pulling step takes $O(n)$ moves. Thus $n$ pulling steps take $O(n^2)$ moves in total.
Hexagonal Lattice Polygons

- Use similar technique of *block-collapsing* as square lattice.
- New definition: A *block* = a hexagonal cell.

**Algorithm:**

 Collapse leftmost collapsible block iteratively.

- **Case 1a**
- **Case 1b**
- **Case 2**
- **Case 3**

Observe: operation in Case 3, no edges are reduced.

After at most $O(n)$ operations of Case 3, we reach one Case 1 or Case 2.

Thus whole algorithm takes $O(n^2)$ moves and time.
Triangular Lattice Chains

- Use similar technique as for square & hexagonal lattice.

**Algorithm:**
- Fold up the end edges of the chain iteratively.
- Unfold the final folded spring/zig-zag path.

**Cases to handle:**
1. When angle $\alpha \%$ end edge & its adjacent edge is $\pi/3$:
   
   ![Diagram](image)

   (a)  \hspace{1cm} (b)

2. When angle $\alpha = 2\pi/3$ or $\pi$, we can handle similarly.
Given a triangular lattice tree $P$.

Let $r$ be root = the leftmost vertex of $P$.

Algorithm:

Our algorithm proceeds by pulling $P$ to the left successively until the whole tree is straightened.

In each pulling step:

- Each vertex $v$ is pulled along its edge (or its extension) connecting to its parent;
- The motion of $v$ stops when $v$ is coincident with its parent in the previous step.
In each pulling step:

- Each vertex $v$ is pulled along its edge connecting to its parent;
- The motion of $v$ stops when $v$ is coincident with its parent in the previous step.

During motion, a vertex never exceed middle point of an extension edge (dashed blue). So no edge crossings occur.

Each pulling step takes $O(n)$ moves, and thus the whole algorithm takes $O(n^2)$ moves and time.
We extend block-collapsing technique for square/hexagonal lattice.

New definition:

A block = a (a) parallelogram, (b) trapezoidal or (c) triangular block.
Algorithm:

Collapse leftmost collapsible block iteratively.

Cases to handle:
(a) Collapse a parallelogram block;
(b) Collapse two trapezoidal/triangular blocks; and
(c) Collapse an extended triangular/trapezoidal block.
Conclusions

- We obtain results for chains/trees/polygons on hexagonal or triangular lattice can be straightened/covexified.

**Conjecture:**
A unit tree in two dimensions can always be straightened.
We obtain results for chains/trees/polygons on hexagonal or triangular lattice can be straightened/covexified.

Conjecture:
A unit tree in two dimensions can always be straightened.