

The Complexity of Finding Paths in Tournaments

Till Tantau

International Computer Science Institute
Berkeley, California

January 30th, 2004

Outline

1 Introduction

- What are Tournaments?
- What Does “Finding Paths” Mean?

2 Review

- Standard Complexity Classes
- Standard Complexity Results on Finding Paths

3 Finding Paths in Tournaments

- Complexity of: Does a Path Exist?
- Complexity of: Construct a Shortest Path
- Complexity of: Approximating the Shortest Path

Outline

1 Introduction

- What are Tournaments?
- What Does “Finding Paths” Mean?

2 Review

- Standard Complexity Classes
- Standard Complexity Results on Finding Paths

3 Finding Paths in Tournaments

- Complexity of: Does a Path Exist?
- Complexity of: Construct a Shortest Path
- Complexity of: Approximating the Shortest Path

Outline

1 Introduction

- What are Tournaments?
- What Does “Finding Paths” Mean?

2 Review

- Standard Complexity Classes
- Standard Complexity Results on Finding Paths

3 Finding Paths in Tournaments

- Complexity of: Does a Path Exist?
- Complexity of: Construct a Shortest Path
- Complexity of: Approximating the Shortest Path

Outline

1 Introduction

- What are Tournaments?
- What Does “Finding Paths” Mean?

2 Review

- Standard Complexity Classes
- Standard Complexity Results on Finding Paths

3 Finding Paths in Tournaments

- Complexity of: Does a Path Exist?
- Complexity of: Construct a Shortest Path
- Complexity of: Approximating the Shortest Path

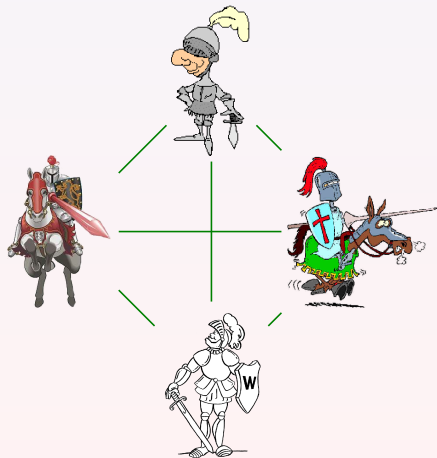
Tournaments Consist of Jousts Between Knights



What is a Tournament?

- A group of knights.
- Every pair has a joust.
- In every joust one knight wins.

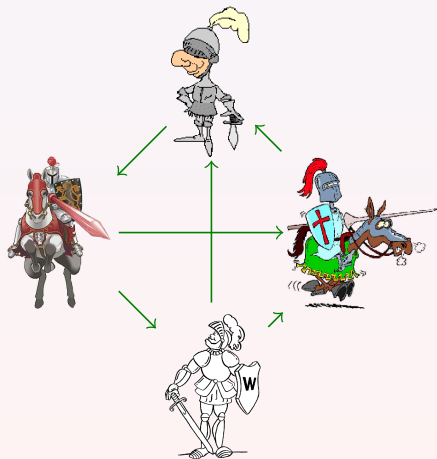
Tournaments Consist of Jousts Between Knights



What is a Tournament?

- A group of knights.
- Every pair has a joust.
- In every joust one knight wins.

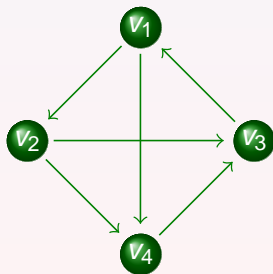
Tournaments Consist of Jousts Between Knights



What is a Tournament?

- A group of knights.
- Every pair has a joust.
- In every joust one knight wins.

Tournaments are Complete Directed Graphs



Definition

A **tournament** is a

- 1 directed graph,
- 2 with exactly one edge between any two different vertices.

Tournaments Arise Naturally in Different Situations

Applications in Ordering Theory

Elements in a set need to be sorted.

The comparison relation may be cyclic, however.

Applications in Sociology

Several candidates apply for a position.

Reviewers decide for any two candidates whom they prefer.

Applications in Structural Complexity Theory

A language L is given and a selector function f .

It chooses from any two words the one more likely to be in f .

Tournaments Arise Naturally in Different Situations

Applications in Ordering Theory

Elements in a set need to be sorted.

The comparison relation may be cyclic, however.

Applications in Sociology

Several candidates apply for a position.

Reviewers decide for any two candidates whom they prefer.

Applications in Structural Complexity Theory

A language L is given and a selector function f .

It chooses from any two words the one more likely to be in f .

Tournaments Arise Naturally in Different Situations

Applications in Ordering Theory

Elements in a set need to be sorted.

The comparison relation may be cyclic, however.

Applications in Sociology

Several candidates apply for a position.

Reviewers decide for any two candidates whom they prefer.

Applications in Structural Complexity Theory

A language L is given and a selector function f .

It chooses from any two words the one more likely to be in f .

Outline

1 Introduction

- What are Tournaments?
- What Does “Finding Paths” Mean?

2 Review

- Standard Complexity Classes
- Standard Complexity Results on Finding Paths

3 Finding Paths in Tournaments

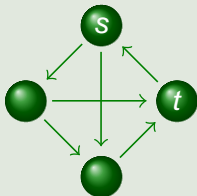
- Complexity of: Does a Path Exist?
- Complexity of: Construct a Shortest Path
- Complexity of: Approximating the Shortest Path

“Finding Paths” is Ambiguous

Input for Path Finding Problems

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.

Example Input



“Finding Paths” is Ambiguous

Input for REACH

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.

Variants of Path Finding Problems

Reachability Problem: Is there a path from s to t ?

Construction Problem: Construct a path from s to t ?

Optimization Problem: Construct a shortest path from s to t .

Distance Problem: Is the distance of s and t at most d ?

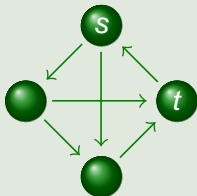
Approximation Problem: Construct a path from s to t of length approximately their distance.

“Finding Paths” is Ambiguous

Input for REACH

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.

Example Input



Example Output

“Yes”

“Finding Paths” is Ambiguous

Input for the Construction Problem

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.

Variants of Path Finding Problems

Reachability Problem: Is there a path from s to t ?

Construction Problem: Construct a path from s to t ?

Optimization Problem: Construct a shortest path from s to t .

Distance Problem: Is the distance of s and t at most d ?

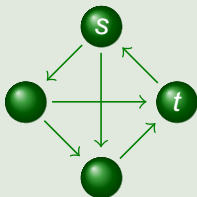
Approximation Problem: Construct a path from s to t of length approximately their distance.

“Finding Paths” is Ambiguous

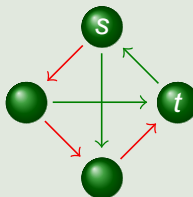
Input for the Construction Problem

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.

Example Input



Example Output



“Finding Paths” is Ambiguous

Input for the Optimization Problem

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.

Variants of Path Finding Problems

Reachability Problem: Is there a path from s to t ?

Construction Problem: Construct a path from s to t ?

Optimization Problem: Construct a shortest path from s to t .

Distance Problem: Is the distance of s and t at most d ?

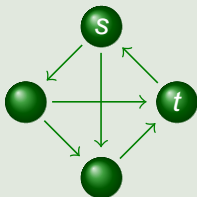
Approximation Problem: Construct a path from s to t of length approximately their distance.

“Finding Paths” is Ambiguous

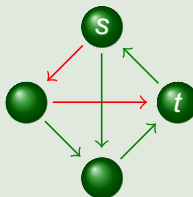
Input for the Optimization Problem

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.

Example Input



Example Output



“Finding Paths” is Ambiguous

Input for DISTANCE

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.
- A **maximum distance** d .

Variants of Path Finding Problems

Reachability Problem: Is there a path from s to t ?

Construction Problem: Construct a path from s to t ?

Optimization Problem: Construct a shortest path from s to t .

Distance Problem: Is the distance of s and t at most d ?

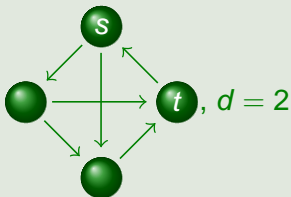
Approximation Problem: Construct a path from s to t of length approximately their distance.

“Finding Paths” is Ambiguous

Input for DISTANCE

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.
- A **maximum distance** d .

Example Input



Example Output

“Yes”

“Finding Paths” is Ambiguous

Input for the Approximation Problem

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.
- An **approximation ratio** $r > 1$.

Variants of Path Finding Problems

Reachability Problem: Is there a path from s to t ?

Construction Problem: Construct a path from s to t ?

Optimization Problem: Construct a shortest path from s to t .

Distance Problem: Is the distance of s and t at most d ?

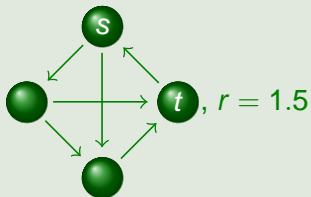
Approximation Problem: Construct a path from s to t of length approximately their distance.

“Finding Paths” is Ambiguous

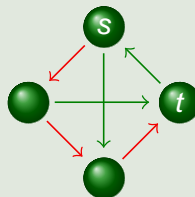
Input for the Approximation Problem

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.
- An **approximation ratio** $r > 1$.

Example Input



Example Output

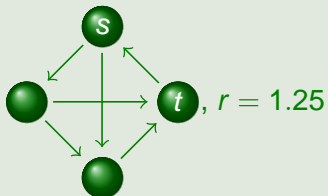


“Finding Paths” is Ambiguous

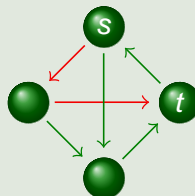
Input for the Approximation Problem

- A **graph** $G = (V, E)$, a **source** $s \in V$ and a **target** $t \in V$.
- An **approximation ratio** $r > 1$.

Example Input



Example Output



Outline

1 Introduction

- What are Tournaments?
- What Does “Finding Paths” Mean?

2 Review

- **Standard Complexity Classes**
- Standard Complexity Results on Finding Paths

3 Finding Paths in Tournaments

- Complexity of: Does a Path Exist?
- Complexity of: Construct a Shortest Path
- Complexity of: Approximating the Shortest Path

The Classes L and NL are Defined via Logspace Turing Machines

input tape (read only), n symbols

3401234*3143223=



work tape (read/write), $O(\log n)$ symbols

42

10690836937182

output tape (write only)

The Classes L and NL are Defined via Logspace Turing Machines

input tape (read only), n symbols

3401234*3143223=



work tape (read/write), $O(\log n)$ symbols

42

10690836937182

output tape (write only)

The Classes L and NL are Defined via Logspace Turing Machines

input tape (read only), n symbols

3401234*3143223=



work tape (read/write), $O(\log n)$ symbols

42

10690836937182

output tape (write only)

Logspace Turing Machines Are Quite Powerful

Deterministic logspace machines can compute

- addition, multiplication, and even division
- reductions used in completeness proofs,
- reachability in forests.

Non-deterministic logspace machines can compute

- reachability in graphs,
- non-reachability in graphs,
- satisfiability with two literals per clause.

Logspace Turing Machines Are Quite Powerful

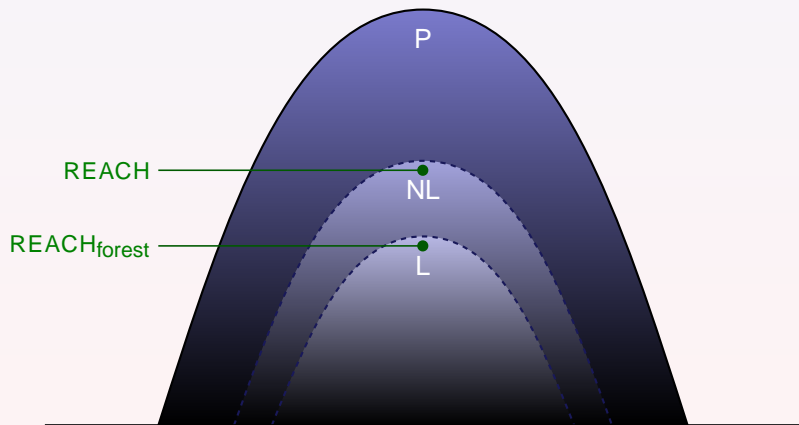
Deterministic logspace machines can compute

- addition, multiplication, and even division
- reductions used in completeness proofs,
- reachability in forests.

Non-deterministic logspace machines can compute

- reachability in graphs,
- non-reachability in graphs,
- satisfiability with two literals per clause.

The Complexity Class Hierarchy



The Circuit Complexity Classes AC^0 , NC^1 , and NC^2 Limit the Circuit Depth

Circuit Class AC^0

- $O(1)$ depth
- unbounded fan-in

Examples

- $ADDITION \in AC^0$.
- $PARITY \notin AC^0$.

Circuit Class NC^1

- $O(\log n)$ depth
- bounded fan-in

Examples

- $PARITY \in NC^1$.
- $MULTIPLY \in NC^1$.
- $DIVIDE \in NC^1$.

Circuit Class NC^2

- $O(\log^2 n)$ depth
- bounded fan-in

Examples

- $NL \subseteq NC^2$.

The Circuit Complexity Classes AC^0 , NC^1 , and NC^2 Limit the Circuit Depth

Circuit Class AC^0

- $O(1)$ depth
- unbounded fan-in

Examples

- $ADDITION \in AC^0$.
- $PARITY \notin AC^0$.

Circuit Class NC^1

- $O(\log n)$ depth
- bounded fan-in

Examples

- $PARITY \in NC^1$.
- $MULTIPLY \in NC^1$.
- $DIVIDE \in NC^1$.

Circuit Class NC^2

- $O(\log^2 n)$ depth
- bounded fan-in

Examples

- $NL \subseteq NC^2$.

The Circuit Complexity Classes AC^0 , NC^1 , and NC^2 Limit the Circuit Depth

Circuit Class AC^0

- $O(1)$ depth
- unbounded fan-in

Examples

- $ADDITION \in AC^0$.
- $PARITY \notin AC^0$.

Circuit Class NC^1

- $O(\log n)$ depth
- bounded fan-in

Examples

- $PARITY \in NC^1$.
- $MULTIPLY \in NC^1$.
- $DIVIDE \in NC^1$.

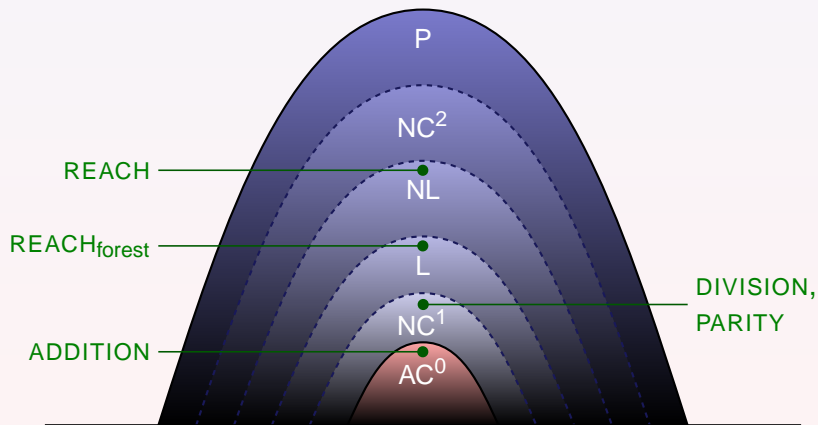
Circuit Class NC^2

- $O(\log^2 n)$ depth
- bounded fan-in

Examples

- $NL \subseteq NC^2$.

The Complexity Class Hierarchy



Outline

1 Introduction

- What are Tournaments?
- What Does “Finding Paths” Mean?

2 Review

- Standard Complexity Classes
- Standard Complexity Results on Finding Paths

3 Finding Paths in Tournaments

- Complexity of: Does a Path Exist?
- Complexity of: Construct a Shortest Path
- Complexity of: Approximating the Shortest Path

All Variants of Finding Paths in Directed Graphs Are Equally Difficult

Fact

REACH and DISTANCE are NL-complete.

Corollary

For directed graphs, we can solve

- the reachability problem in logspace iff $L = NL$.
- the construction problem in logspace iff $L = NL$.
- the optimization problem in logspace iff $L = NL$.
- the approximation problem in logspace iff $L = NL$.

All Variants of Finding Paths in Directed Graphs Are Equally Difficult

Fact

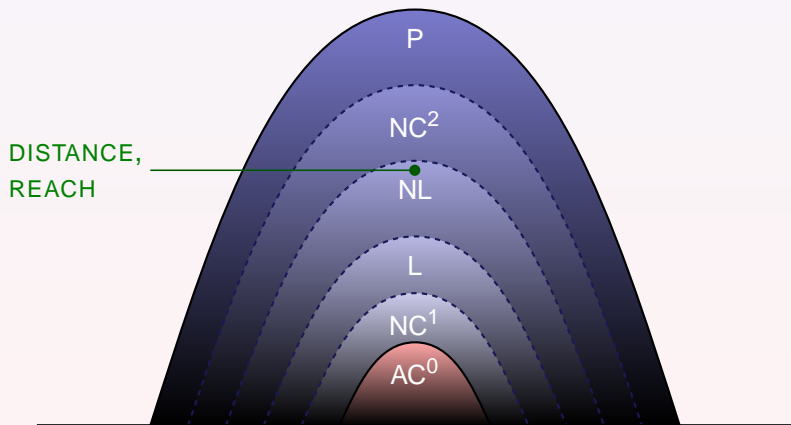
REACH and DISTANCE are NL-complete.

Corollary

For directed graphs, we can solve

- the reachability problem in logspace iff $L = NL$.
- the construction problem in logspace iff $L = NL$.
- the optimization problem in logspace iff $L = NL$.
- the approximation problem in logspace iff $L = NL$.

The Complexity Class Hierarchy



FindingPaths in Forests and Directed Paths is Easy, But Not Trivial

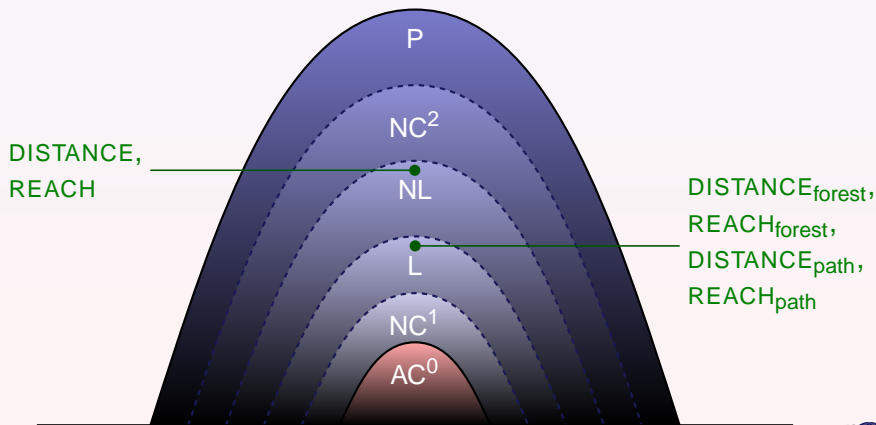
Fact

$\text{REACH}_{\text{forest}}$ and $\text{DISTANCE}_{\text{forest}}$ are L-complete.

Fact

$\text{REACH}_{\text{path}}$ and $\text{DISTANCE}_{\text{path}}$ are L-complete.

The Complexity Class Hierarchy



Outline

1 Introduction

- What are Tournaments?
- What Does “Finding Paths” Mean?

2 Review

- Standard Complexity Classes
- Standard Complexity Results on Finding Paths

3 Finding Paths in Tournaments

- **Complexity of: Does a Path Exist?**
- Complexity of: Construct a Shortest Path
- Complexity of: Approximating the Shortest Path

Definition of the Tournament Reachability Problem

Definition

Let $\text{REACH}_{\text{tourn}}$ contain all triples (T, s, t) such that

- 1 $T = (V, E)$ is a tournament and
- 2 there exists a path from s to t .

The Tournament Reachability Problem is Very Easy

Theorem

$\text{REACH}_{\text{tourn}} \in \text{AC}^0$.

Implications

- The problem is “easier” than REACH and even $\text{REACH}_{\text{path}}$.
- $\text{REACH} \not\leq_m^{\text{AC}^0} \text{REACH}_{\text{tourn}}$.

The Tournament Reachability Problem is Very Easy

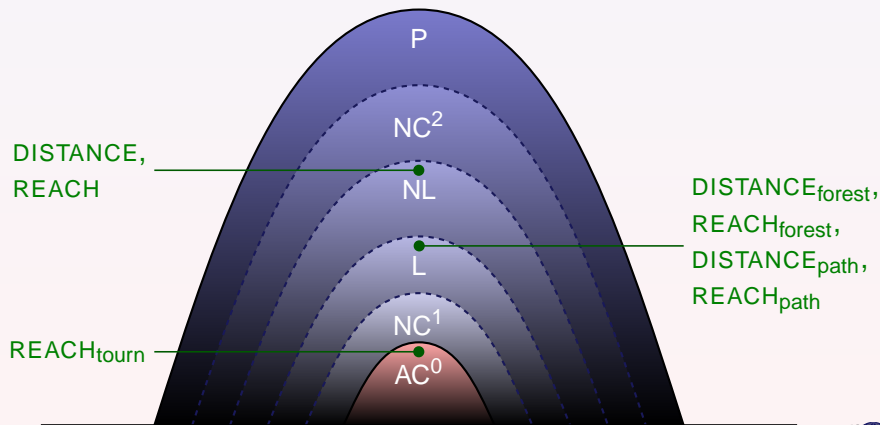
Theorem

$\text{REACH}_{\text{tourn}} \in \text{AC}^0$.

Implications

- The problem is “easier” than REACH and even $\text{REACH}_{\text{path}}$.
- $\text{REACH} \not\leq_m^{\text{AC}^0} \text{REACH}_{\text{tourn}}$.

The Complexity Class Hierarchy



Outline

1 Introduction

- What are Tournaments?
- What Does “Finding Paths” Mean?

2 Review

- Standard Complexity Classes
- Standard Complexity Results on Finding Paths

3 Finding Paths in Tournaments

- Complexity of: Does a Path Exist?
- **Complexity of: Construct a Shortest Path**
- Complexity of: Approximating the Shortest Path

Finding a Shortest Path Is as Difficult as the Distance Problem

Definition

Let **DISTANCE_{tourn}** contain all tuples (T, s, t, d) such that

- 1 $T = (V, E)$ is a tournament in which
- 2 the distance of s and t is at most d .

The Tournament Distance Problem is Hard

Theorem

$\text{DISTANCE}_{\text{tourn}}$ is NL-complete.

» Skip Proof

Corollary

Shortest path in tournaments can be constructed in logarithmic space, iff $L = NL$.

Corollary

$\text{DISTANCE} \leq_m^{\text{AC}^0} \text{DISTANCE}_{\text{tourn}}$.

The Tournament Distance Problem is Hard

Theorem

$\text{DISTANCE}_{\text{tourn}}$ is NL-complete.

» Skip Proof

Corollary

Shortest path in tournaments can be constructed in logarithmic space, iff $L = NL$.

Corollary

$\text{DISTANCE} \leq_m^{\text{AC}^0} \text{DISTANCE}_{\text{tourn}}$.

The Tournament Distance Problem is Hard

Theorem

$\text{DISTANCE}_{\text{tourn}}$ is NL-complete.

» Skip Proof

Corollary

Shortest path in tournaments can be constructed in logarithmic space, iff $L = NL$.

Corollary

$\text{DISTANCE} \leq_m^{\text{AC}^0} \text{DISTANCE}_{\text{tourn}}$.

Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

- 1 If $(G, s, t) \in \text{REACH}$, then $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}$.
- 2 If $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}$, then $(G, s, t) \in \text{REACH}$.

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

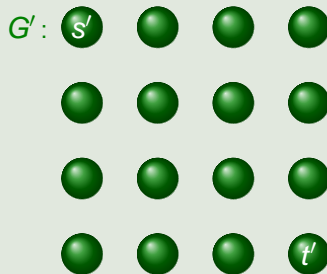
- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

1

2

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

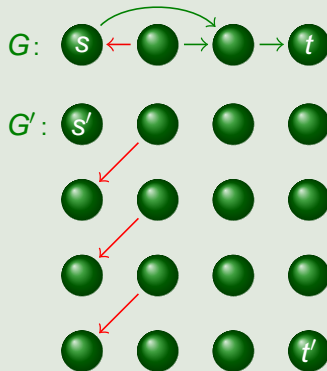
Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

- 1
- 2

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

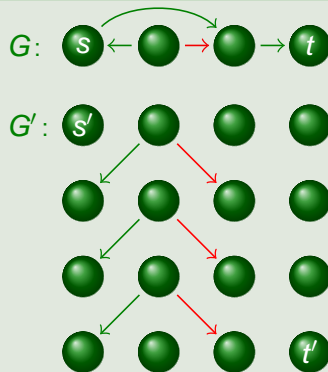
Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

- 1
- 2

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

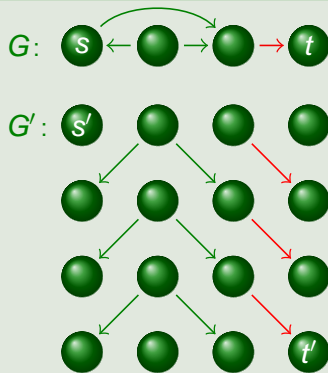
Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

- 1
- 2

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

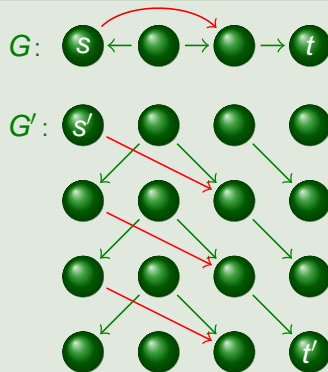
- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

1

2

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

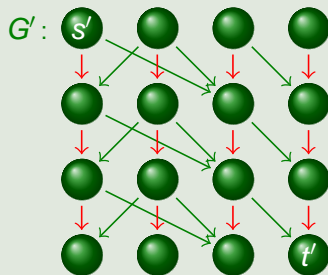
- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

1

2

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

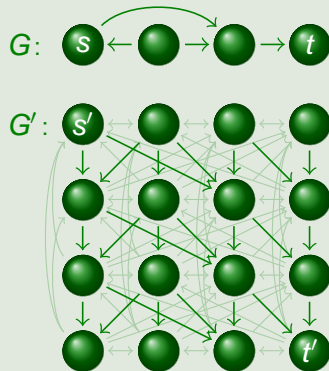
- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

1

2

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

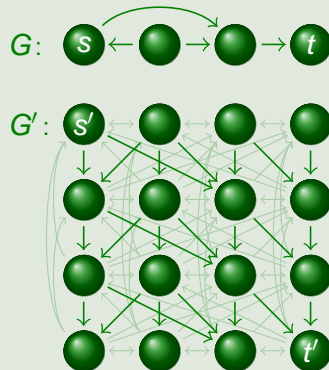
Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

- 1 A path in G induces a length-3 path in G' .
- 2 A length-3 path in G' induces a path in G .

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

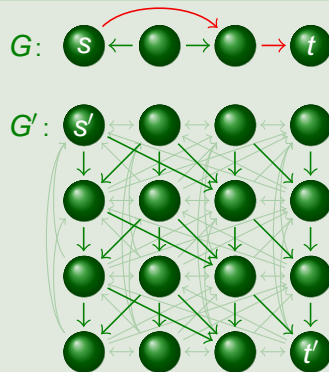
Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

- 1 A path in G induces a length-3 path in G' .
- 2 A length-3 path in G' induces a path in G .

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

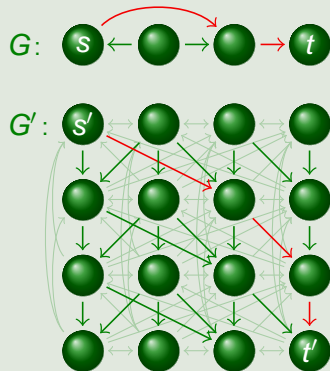
Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

- 1 A path in G induces a length-3 path in G' .
- 2 A length-3 path in G' induces a path in G .

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

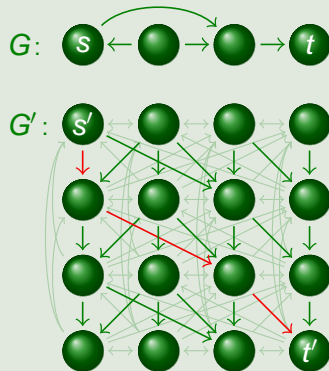
Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

Correctness

- 1 A path in G induces a length-3 path in G' .
- 2 A length-3 path in G' induces a path in G .

Example



Proof That $\text{DISTANCE}_{\text{tourn}}$ is NL-complete

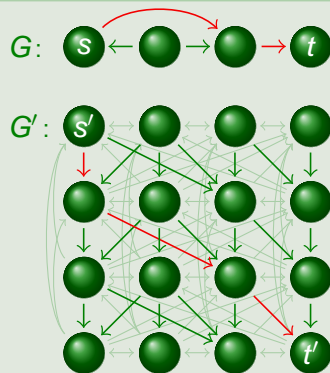
Reduce REACH to $\text{DISTANCE}_{\text{tourn}}$

- 1 Is input (G, s, t) in REACH?
- 2 Map G to G' .
- 3 Query:
 $(G', s', t', 3) \in \text{DISTANCE}_{\text{tourn}}?$

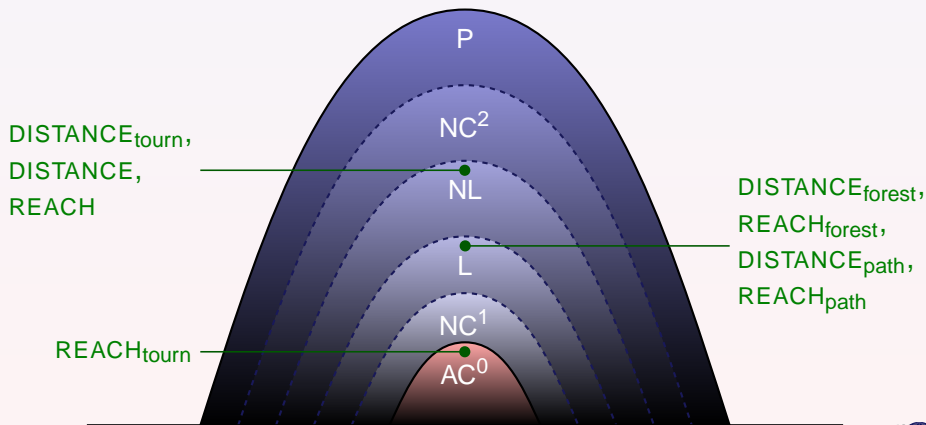
Correctness

- 1 A path in G induces a length-3 path in G' .
- 2 A length-3 path in G' induces a path in G .

Example



The Complexity Class Hierarchy



Outline

1 Introduction

- What are Tournaments?
- What Does “Finding Paths” Mean?

2 Review

- Standard Complexity Classes
- Standard Complexity Results on Finding Paths

3 Finding Paths in Tournaments

- Complexity of: Does a Path Exist?
- Complexity of: Construct a Shortest Path
- Complexity of: Approximating the Shortest Path

Approximators Compute Paths that Are Nearly As Short As a Shortest Path

Definition

An **approximation scheme** for TOURNAMENT-SHORTEST-PATH gets as input

- 1 a tuple $(T, s, t) \in \text{REACH}_{\text{tourn}}$ and
- 2 a number $r > 1$.

It outputs

- a path from s to t of length at most $r d_T(s, t)$.

There Exists a Logspace Approximation Scheme for the Tournament Shortest Path Problem

Theorem

There exists an approximation scheme for TOURNAMENT-SHORTEST-PATH that for $1 < r < 2$ needs space

$$O\left(\log |V| \log \frac{1}{r-1}\right).$$

Corollary

In tournaments, paths can be constructed in logarithmic space.

► More Details



There Exists a Logspace Approximation Scheme for the Tournament Shortest Path Problem

Theorem

There exists an approximation scheme for TOURNAMENT-SHORTEST-PATH that for $1 < r < 2$ needs space

$$O\left(\log |V| \log \frac{1}{r-1}\right).$$

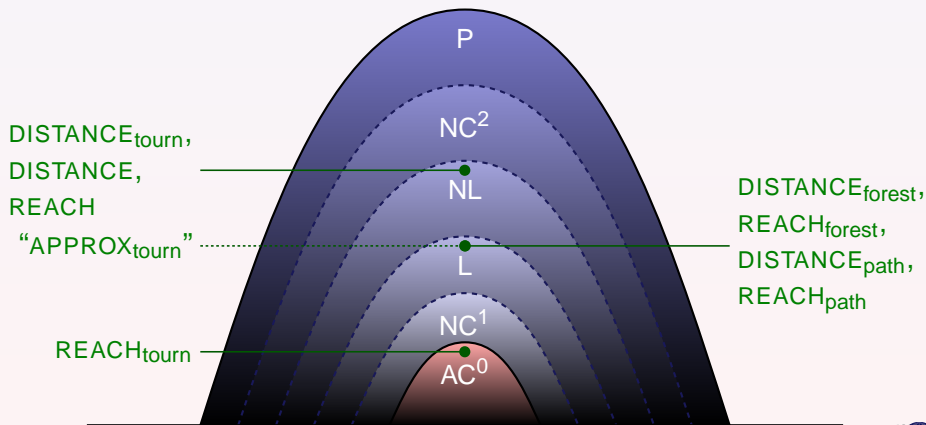
Corollary

In tournaments, paths can be constructed in logarithmic space.

► More Details



The Complexity Class Hierarchy



Summary

Summary

- Tournament **reachability** is in AC^0 .
- There exists a **logspace approximation scheme** for **approximating** shortest paths in tournaments.
- Finding **shortest paths** in tournaments is **NL-complete**.

Outlook

- The same results apply to graphs with bounded independence number.
- The complexity of finding paths in undirected graphs is partly open.

► More Details

► More Details

For Further Reading



John Moon.

Topics on Tournaments.

Holt, Rinehart, and Winston, 1968.



Arfst Nickelsen and Till Tantau.

On reachability in graphs with bounded independence number.

In *Proc. of COCOON 2002*, Springer-Verlag, 2002.



Till Tantau

A logspace approximation scheme for the shortest path problem for graphs with bounded independence number.

In *Proc. of STACS 2004*, Springer-Verlag, 2004.

In press.

Definition of Independence Number of a Graph

Definition

The **independence number** $\alpha(G)$ of a directed graph is the maximum number of vertices we can pick, such that there is no edge between them.

Example

Tournaments have independence number 1.

The Results for Tournaments also Apply to Graphs With Bounded Independence Number

Theorem

For each k , **reachability** in graphs with independence number at most k is in AC^0 .

Theorem

For each k , there exists a **logspace approximation scheme** for approximating the shortest path in graphs with independence number at most k

Theorem

For each k , finding the **shortest path** in graphs with independence number at most k is **NL-complete**.

The Complexity of Finding Paths in Undirected Graphs Is Partly Unknown.

Fact

$\text{REACH}_{\text{undirected}}$ is SL-complete.

Corollary

For undirected graphs, we can solve

- the reachability problem in logspace iff $L = \text{SL}$,
- the construction problem in logspace iff ?,
- the optimization problem in logspace iff ?,
- the approximation problem in logspace iff ?.

The Complexity of Finding Paths in Undirected Graphs Is Party Unknown.

Fact

$\text{REACH}_{\text{undirected}}$ is SL-complete.

Corollary

For undirected graphs, we can solve

- the reachability problem in logspace iff $L = \text{SL}$,
- the construction problem in logspace iff $L = \text{SL}$,
- the optimization problem in logspace iff $L = \text{NL}$,
- the approximation problem in logspace iff ?.

The Approximation Scheme is Optimal

Theorem

Suppose there exists an approximation scheme for TOURNAMENT-SHORTEST-PATH that needs space $O(\log |V| \log^{1-\epsilon} \frac{1}{r-1})$. Then $NL \subseteq DSPACE[\log^{2-\epsilon} n]$.

Proof.

- 1 Suppose the approximation scheme exists.
We show $DISTANCE_{\text{tourn}} \in DSPACE[\log^{2-\epsilon} n]$.
- 2 Let (T, s, t) be an input. Let n be the number of vertices.
- 3 Run the approximation scheme for $r := 1 + \frac{1}{n+1}$.
This needs space $O(\log^{2-\epsilon} n)$.
- 4 The resulting path has optimal length.