Szymanski’s Conjecture and the Complexity of Permutation Routing in the Hypercube

Timothy Carnes
Advisor: Professor Ran Libeskind-Hadas
Reader: Professor Francis Edward Su

Harvey Mudd College
Presentation Days
May 2, 2005
Introduction

Distributed memory parallel computers are comprised of:
- A collection of processors
- A network topology connecting the processors

Complete graph is the theoretically ideal topology. The degree of each vertex increases linearly as more processors are added. Due to economical and physical constraints, other topologies are generally used instead. Hypercubes are one common topology. The degree of each vertex increases logarithmically as more processors are added.
Introduction

- Distributed memory parallel computers are comprised of
Introduction

• Distributed memory parallel computers are comprised of
  ◦ A collection of processors
Introduction

- Distributed memory parallel computers are comprised of
  - A collection of processors
  - A network topology connecting the processors

Complete graph is the theoretically ideal topology
Degree of each vertex increases linearly as more processors are added

Due to economical and physical constraints, other topologies are generally used instead

Hypercubes are one common topology
Degree of each vertex increases logarithmically as more processors are added
Introduction

- Distributed memory parallel computers are comprised of
  - A collection of processors
  - A network topology connecting the processors
- Complete graph is the theoretically ideal topology
Introduction

- Distributed memory parallel computers are comprised of
  - A collection of processors
  - A network topology connecting the processors
- Complete graph is the theoretically ideal topology
- Degree of each vertex increases *linearly* as more processors are added
Introduction

- Distributed memory parallel computers are comprised of
  - A collection of processors
  - A network topology connecting the processors
- Complete graph is the theoretically ideal topology
- Degree of each vertex increases *linearly* as more processors are added
- Due to economical and physical constraints, other topologies are generally used instead
Introduction

- Distributed memory parallel computers are comprised of
  - A collection of processors
  - A network topology connecting the processors
- Complete graph is the theoretically ideal topology
- Degree of each vertex increases linearly as more processors are added
- Due to economical and physical constraints, other topologies are generally used instead
- Hypercubes are one common topology
Introduction

- Distributed memory parallel computers are comprised of
  - A collection of processors
  - A network topology connecting the processors
- Complete graph is the theoretically ideal topology
- Degree of each vertex increases *linearly* as more processors are added
- Due to economical and physical constraints, other topologies are generally used instead
- Hypercubes are one common topology
- Degree of each vertex increases *logarithmically* as more processors are added
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

\[ n = 0 \]
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

\[ n = 0 \quad n = 1 \]
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

\[ n = 0 \quad n = 1 \]
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

$n = 0 \quad n = 1$
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

$n = 0 \quad n = 1 \quad n = 2$
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

$n = 0 \quad n = 1 \quad n = 2$
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

$n = 0$ \hspace{1cm} n = 1 \hspace{1cm} n = 2 \hspace{1cm} n = 3$
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

$n = 0$  $n = 1$  $n = 2$  $n = 3$
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

$n = 0 \quad n = 1 \quad n = 2 \quad n = 3$
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

$n = 0 \quad n = 1 \quad n = 2 \quad n = 3 \quad n = 4$
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

\[
\begin{align*}
  n &= 0 & n &= 1 & n &= 2 & n &= 3 & n &= 4
\end{align*}
\]
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

- To obtain the directed hypercube, replace each edge with two oppositely-oriented directed edges.
What is a Hypercube?

- The $n$-dimensional hypercube, for $n \geq 0$

  $n = 0$  $n = 1$  $n = 2$  $n = 3$  $n = 4$

- To obtain the directed hypercube, replace each edge with two oppositely-oriented directed edges.
Hypercube Routing

• Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.
Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target. We also want no two paths to share an edge.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target
- We also want no two paths to share an edge
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.

- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.
- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.

- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.

- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.
- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.

- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.

- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.

- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.

- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.

- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.

- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.

- There are two main variants: minimal-length paths and arbitrary-length paths.
Hypercube Routing

- Given a set of source-target pairs of vertices, we want to establish paths connecting each source to each target.
- We also want no two paths to share an edge.

- There are two main variants: minimal-length paths and arbitrary-length paths.
Permutations

We are interested in special sets of source-target pairs known as permutations. A permutation is a set of source-target pairs for which each vertex appears at most once as a source, and appears at most once as a target.

Example:

1 7!
2 7!
3 7!
4 7!
5

Non-Example:

1 7!
2 1 7!
3 4 7!
2
Permutations

- We are interested in special sets of source-target pairs known as permutations
Permutations

- We are interested in special sets of source-target pairs known as permutations
- A permutation is a set of source-target pairs for which each vertex
Permutations

- We are interested in special sets of source-target pairs known as permutations.
- A permutation is a set of source-target pairs for which each vertex
  - appears at most once as a source, and
Permutations

- We are interested in special sets of source-target pairs known as permutations.
- A permutation is a set of source-target pairs for which each vertex
  - appears at most once as a source, and
  - appears at most once as a target.
Permutations

• We are interested in special sets of source-target pairs known as permutations

• A permutation is a set of source-target pairs for which each vertex
  ◦ appears at most once as a source, and
  ◦ appears at most once as a target

1 ↔ 3
2 ↔ 1
3 ↔ 4
4 ↔ 2

Example:
Permutations

- We are interested in special sets of source-target pairs known as permutations.
- A permutation is a set of source-target pairs for which each vertex
  - appears at most once as a source, and
  - appears at most once as a target.

Example:

\[
\begin{align*}
1 & \leftrightarrow 3 \\
2 & \leftrightarrow 1 \\
3 & \leftrightarrow 4 \\
4 & \leftrightarrow 2
\end{align*}
\]

Non-Example:

\[
\begin{align*}
1 & \leftrightarrow 3 \\
4 & \leftrightarrow 2
\end{align*}
\]
What Do We Know?

- Do routings for permutations always exist? Szymanski conjectured they do always exist.
  - In the case of minimal-length paths, a counterexample was found for 4-dimensional hypercube.
  - For arbitrary-length paths, the problem remains open.
- Can we find routings for permutations efficiently? Gonzalez and Serena have proved that the minimal-length path version of the problem is NP-complete, which means it is computationally intractable!
- The complexity of the arbitrary-length path version remains open.
- If we had twice as many edges, we could route two permutations simultaneously!
What Do We Know?

- Do routings for permutations always exist?
What Do We Know?

- Do routings for permutations always exist?
  - Szymanski conjectured they do always exist
What Do We Know?

- Do routings for permutations always exist?
  - Szymanski conjectured they do always exist
  - In the case of minimal-length paths, counterexample found for 4-dimensional hypercube
What Do We Know?

- Do routings for permutations always exist?
  - Szymanski conjectured they do always exist
  - In the case of minimal-length paths, counterexample found for 4-dimensional hypercube
  - For arbitrary-length paths the problem remains open
What Do We Know?

- Do routings for permutations always exist?
  - Szymanski conjectured they do always exist
  - In the case of minimal-length paths, counterexample found for 4-dimensional hypercube
  - For arbitrary-length paths the problem remains open
- Can we find routings for permutations efficiently?

Gonzalez and Serena have proved that the minimal-length path version of the problem is NP-complete, which means it is computationally intractable! The complexity of the arbitrary-length path version remains open. If we had twice as many edges, we could route two permutations simultaneously!
What Do We Know?

- Do routings for permutations always exist?
  - Szymanski conjectured they do always exist
  - In the case of minimal-length paths, counterexample found for 4-dimensional hypercube
  - For arbitrary-length paths the problem remains open

- Can we find routings for permutations efficiently?
  - Gonzalez and Serena have proved that the minimal-length path version of the problem is NP-complete = computationally intractable!
What Do We Know?

- Do routings for permutations always exist?
  - Szymanski conjectured they do always exist
  - In the case of minimal-length paths, counterexample found for 4-dimensional hypercube
  - For arbitrary-length paths the problem remains open

- Can we find routings for permutations efficiently?
  - Gonzalez and Serena have proved that the minimal-length path version of the problem is NP-complete = computationally intractable!
  - The complexity of the arbitrary-length path version remains open
What Do We Know?

- Do routings for permutations always exist?
  - Szymanski conjectured they do always exist
  - In the case of minimal-length paths, counterexample found for 4-dimensional hypercube
  - For arbitrary-length paths the problem remains open

- Can we find routings for permutations efficiently?
  - Gonzalez and Serena have proved that the minimal-length path version of the problem is NP-complete = computationally intractable!
  - The complexity of the arbitrary-length path version remains open
  - If we had twice as many edges, we could route two permutations simultaneously!
NP-Completeness

- Some problems have been found to be really hard (computationally intractable)
NP-Completeness

- Some problems have been found to be really hard (computationally intractable)

Large Class of Problems (NP)
NP-Completeness

- Some problems have been found to be really hard (computationally intractable)

Large Class of Problems (NP)

3SAT (Cook-Levin)
NP-Completeness

- Some problems have been found to be really hard (computationally intractable)

1. Large Class of Problems (NP)
2. Problem of Interest
3. 3SAT (Cook-Levin)
L3-SAT

- A satisfiability problem in propositional logic
L3-SAT

- A satisfiability problem in propositional logic
- Example:

\[(x_1 \lor x_4) \land (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_3 \lor x_4) \land (x_1 \lor x_2 \lor x_4)\]
L3-SAT

- A satisfiability problem in propositional logic
- Example:

\[(x_1 \lor x_4) \land (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor x_3 \lor \overline{x_4}) \land (\overline{x_1} \lor x_2 \lor x_4)\]

- Each clause has two or three literals
L3-SAT

- A satisfiability problem in propositional logic
- Example:

\[(x_1 \lor x_4) \land (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor x_3 \lor \overline{x_4}) \land (\overline{x_1} \lor x_2 \lor x_4)\]

- Each clause has two or three literals
- No literal appears more than twice
L3-SAT

- A satisfiability problem in propositional logic
- Example:

\[(x_1 \lor \overline{x_4}) \land (x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3 \lor x_4) \land (\overline{x_1} \lor x_2 \lor x_4)\]

- Each clause has two or three literals
- No literal appears more than twice
- Non-Example

\[(x_1 \lor x_2) \land (x_1 \lor x_3) \land (x_1 \lor x_4)\]
Reduction Idea

- Want solution to routings problem to correspond to solution to L3-SAT
Reduction Idea

- Want solution to routings problem to correspond to solution to L3-SAT
Reduction Idea

• Want solution to routings problem to correspond to solution to L3-SAT

• By choosing a path, we also choose a truth assignment for a variable
Reduction Idea

- Want solution to routings problem to correspond to solution to L3-SAT

\[ x_i \text{ is TRUE} \]

- By choosing a path, we also choose a truth assignment for a variable
Reduction Idea

- Want solution to routings problem to correspond to solution to L3-SAT

- By choosing a path, we also choose a truth assignment for a variable
Clause-Checking Gadget

\((x_1 \lor \overline{x_2} \lor x_3)\)
Clause-Checking Gadget

\[(x_1 \lor \overline{x_2} \lor x_3)\]
Clause-Checking Gadget

\[(x_1 \lor \overline{x_2} \lor x_3)\]
Clause-Checking Gadget

\((x_1 \lor \bar{x}_2 \lor x_3)\)
Clause-Checking Gadget

\((x_1 \lor \overline{x_2} \lor x_3)\)
Clause-Checking Gadget

\[(x_1 \lor \overline{x_2} \lor x_3)\]

\[s \rightarrow t\]

\[\cdots\]
Clause-Checking Gadget

\[(x_1 \lor \overline{x_2} \lor x_3)\]
Clause-Checking Gadget

\[(x_1 \lor \overline{x_2} \lor x_3)\]
Clause-Checking Gadget

\((x_1 \lor \overline{x_2} \lor x_3)\)
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?

\[ (x_1 \lor \overline{x_2} \lor x_3) \]
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?

\[(x_1 \lor \overline{x_2} \lor x_3)\]
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?
Convey Apparatus

• How can we connect the variable-setter to the clause-checker?

\[(x_1 \lor \overline{x}_2 \lor x_3)\]
Convey Apparatus

- How can we connect the variable-setter to the clause-checker?

\[(x_1 \lor \overline{x_2} \lor x_3)\]
Convey Apparatus

• How can we connect the variable-setter to the clause-checker?
Remarks

- Problem of permutation routing on hypercube using any length paths is still open
Remarks

- Problem of permutation routing on hypercube using any length paths is still open
- We conjecture that this problem is also NP-complete
Remarks

- Problem of permutation routing on hypercube using any length paths is still open
- We conjecture that this problem is also NP-complete
- We have shown that arbitrary-length path routing on the hypercube is NP-complete, but does not apply to permutations
Remarks

- Problem of permutation routing on hypercube using any length paths is still open
- We conjecture that this problem is also NP-complete
- We have shown that arbitrary-length path routing on the hypercube is NP-complete, but does not apply to permutations
- Approximation algorithms have been found which cleverly transform the hypercube into a Multistage Interconnection Network, or MIN
Remarks

- Problem of permutation routing on hypercube using any length paths is still open
- We conjecture that this problem is also NP-complete
- We have shown that arbitrary-length path routing on the hypercube is NP-complete, but does not apply to permutations
- Approximation algorithms have been found which cleverly transform the hypercube into a Multistage Interconnection Network, or MIN
- If we double every edge in the hypercube, we can route two arbitrary permutations
Remarks

- Problem of permutation routing on hypercube using any length paths is still open
- We conjecture that this problem is also NP-complete
- We have shown that arbitrary-length path routing on the hypercube is NP-complete, but does not apply to permutations
- Approximation algorithms have been found which cleverly transform the hypercube into a Multistage Interconnection Network, or MIN
- If we double every edge in the hypercube, we can route two arbitrary permutations
- We can route one permutation by doubling $2^n$ edges
References
