Szymanski's Conjecture and the Complexity of Hypercube Routing

Tim Carnes

Advisor: Ran Libeskind-Hadas

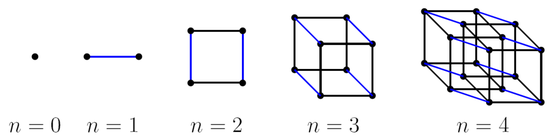
Harvey Mudd College

mathematics, harvey mudd college

Introduction

Distributed memory parallel computers comprise a collection of processors which are interconnected via some network topology. A complete graph would be ideal for allowing the swiftest communication between processors, but such connections are impractical due to physical and economic constraints. Instead hypercubes are often used as topologies as they offer a relatively small diameter while keeping the degree of each node fairly low.

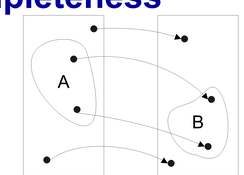
A hypercube may be defined inductively as follows. A 0-dimensional hypercube is simply a single vertex. To construct a (n+1)-dimensional hypercube, we make two copies of a n-dimensional hypercube and connect each vertex to the copy of itself. The directed hypercube results from replacing each edge in the undirected hypercube with two oppositely-oriented directed edges.



Establishing connections between processors motivates the following problem. Given a set of source-target pairs, which are ordered pairs of vertices in a hypercube, establish paths linking each source to its corresponding target such that no edge is used by more than one path.

Szymanski conjectured that for special sets of source-target pairs known as permutations, a routing would always be possible. A permutation is a set of source-target pairs in which each node appears at most once as a source and at most once as a target.

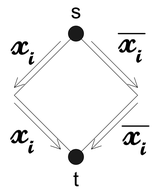
NP-Completeness



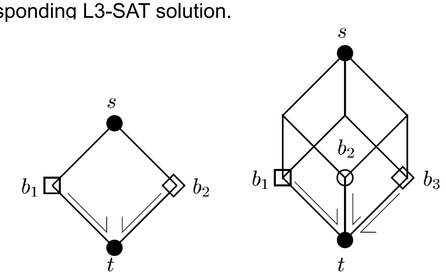
We are concerned not only with the existence of routings, but also if we can find them efficiently. We show that the problem of hypercube routing is NP-complete, which implies that it is computationally intractable. This is shown by simply reducing a known provably hard problem, *A*, to this problem, *B*. A reduction must map every problem instance of *A* into a problem instance of *B* such that the answers to both problem instances are identical.

Complexity Results

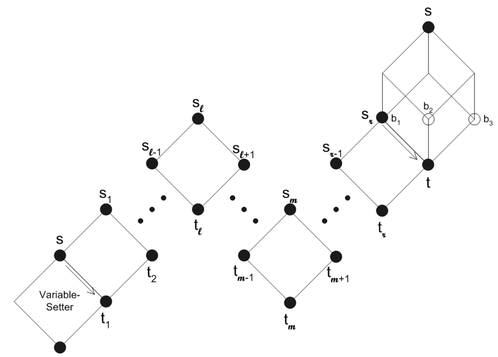
Gonzalez and Serena provide a reduction from L3-SAT to hypercube routing, where all paths must be of minimal length. An L3-SAT instance simply gives a specific type of boolean formula and asks whether or not it is satisfiable. The formula consists of a set of clauses connected with ANDs, where each clause is either two or three literals connected with ORs. A literal is simply a boolean variable in either its regular or complemented form. To reduce L3-SAT to hypercube routing, we use three main constructions: the variable-setter, the clause-checker and the convey apparatus.



The variable-setter is created for every variable *x* in the L3-SAT instance, and consists of a single request pair of distance two. There are only two possible shortest paths available to route source *s* to target *t*. The path that is chosen in the routing solution will determine the truth assignment of *x* in the corresponding L3-SAT solution.



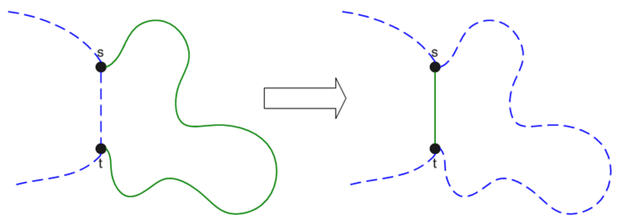
The clause-checker is created for each clause in the L3-SAT instance, and consists of a single request pair of distance two or three, depending on whether the clause has two or three variables. In order to make this reduction work, we want the clause-checker request pair to be satisfiable if and only if it contains a literal that is satisfied. To this end, we will connect each edge adjacent to *t* in the clause-checker to an edge corresponding to the negation of one of the literals in the clause in the variable-setter. The connection will be such that if the edge in the variable-setter is used, then so will the edge in the clause-checker. Thus if no literal in the clause is satisfied, then all edges leading to *t* will be used by the connection devices, and so there will be no minimal-length path available.



All that remains is to connect the edges in the variable-setters to the appropriate clause-checkers in the way that we require. This is done by the convey apparatus, which consists simply of a series of pair requests of distance two. If the edge in the variable-setter is used, then s_1 will be forced to route its path through t_2, which causes a chain reaction ultimately forcing the edge in the clause-checker to be used. This is precisely what we want, and we can cleverly design these devices so that no two convey apparatuses will intersect.

Extension

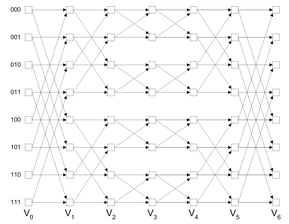
We can use the reduction of Gonzalez and Serena to show the NP-completeness of the problem where the path lengths may be arbitrary. To do this we simply observe that whenever a source and target are a unit distance apart, we may assume without loss of generality that they are connected by a single edge.



Now we can take the same construction of the components and simply force all paths to be shortest paths by adding enough source-target pairs to block off the necessary edges. To do this however, we relax the constraint that the set of source-target pairs is a permutation. The complexity of the problem where the path lengths are arbitrary and the set of source-target pairs is a permutation is still open.

Approximation Algorithm

Although we do not know how to solve the problem of routing a permutation on the hypercube, we can show a result nearly as good. If we changed the directed hypercube so that every edge appeared twice, then we can in fact route two permutations. To do this we first show how to transform the hypercube into what is known as a Multistage Interconnection Network, or MIN. This is a network where request pairs will be routed through a series of stages, where at each stage only certain edges will be available. The edges available correspond to all of the edges crossing a particular dimension of the hypercube. A MIN is thus a representation of the hypercube, and the one corresponding to the three-dimensional cube is shown below.



The advantage of transforming the hypercube into a MIN is that we can now apply known results, and in particular a routing method of Beneš that can be explained inductively. Clearly we can route two permutations on the 0-dimensional hypercube since there is only one vertex. Now, if we assume that we know how to route two permutations on a hypercube of dimension n, then we will show to do so for dimension $n+1$.

We first take one permutation, and map its sources to a n-dimensional subcube by using all edges crossing dimension 1. This will put two sources on each vertex in the subcube. We do similarly with the targets, now using all edges crossing dimension 1 in the opposite direction, which again puts two targets on each vertex in the subcube. This is equivalent to two permutation requests in the subcube, which we know how to route by our assumption. We follow the same procedure for mapping the other permutation to the other subcube, and now we've used edges crossing each dimension at most twice.

Acknowledgments

Many thanks to my advisor, Ran, for all of his support and encouragement.

Bibliography

T. Szymanski. On the Permutation Routing of a Circuit-Switched Hypercube. *Proc. International Conference on Parallel Processing (ICPP)*, 1:103-110, 1989.

T. F. Gonzalez, D. Serena. Complexity of *k*-Pairwise Disjoint Shortest Paths in the Undirected Hypercubic Network and Related Problems. *PDCS*, 61-66, 2002.