What Is the Best Way To Draw a Cube? A Hypercube?

Brian d'Auriol, Vladik Kreinovich,
Bindu George, Florence Muganda, and
Pramod Kumar Chikkapaiah

Department of Computer Science
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
contact email vladik@cs.utep.edu

Abstract

One of the possible connections between processors is a *hypercube*. The simplest case of a hypercube – a 4-vertex square – can be naturally represented on a 2-D page. To represent a 3-dimensional (or higher-dimensional) hypercube, we must *project* additional dimensions onto a 2-D page. In general, when we project a multi-D space into a 2-D plane, different points project into the same one. To get the best visualization, we must select a projection in such a way that the projections of different points are as distant from each other as possible. In this paper, we formalize and solve the corresponding optimization problem. Thus, we show what is the best way of drawing a cube.

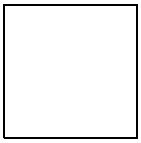
1 Introduction

One of the possible connections between processors in a parallel computer is an (n-dimensional) *hypercube* (see, e.g., [1]). In a hypercube, 2^n processors are in 1-1 correspondence with n-bit sequences (i.e., sequences of n 0's and 1's), and two processors are connected if and only if the corresponding sequences differ in exactly one bit.

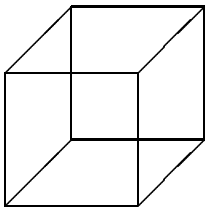
To analyze the behavior of different algorithms, we would like to *visualize* a hypercube, i.e., to represent its processors (vertices) and connections between these processors (edges) in a 2-D page (or on a 2-D screen).

For a 2-D "cube" (square), there is no problem: we can represent its four vertices – corresponding to (0,0), (0,1), (1,0), and (1,1) – as points on the plane

with exactly these coordinates:



For a 3-D cube, the situation is not that straightforward. To represent a 3-dimensional (or higher-dimensional) hypercube, we must *project* the additional dimension(s) onto a 2-D page. As a result, for a 3-D cube, we get a picture of the following type:



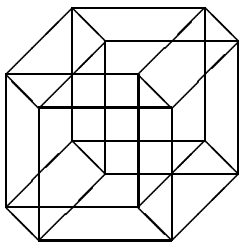
Let us describe this picture in precise terms. In this picture, four out of eight vertices still have the same "binary" coordinates vertices (0,0), (0,1), (1,0), and (1,1). To describe the coordinates of four other vertices, it is sufficient to describe the coordinates (a, b) of a vertex which is connected to (0,1). Then, the other three vertices have coordinates $(a, 1+b)$, $(1+a, b)$, and $(1+a, 1+b)$, and the eight points are:

$$(0,0),\ (0,1),\ (1,0),\ (1,1),$$

$$(a,b),\ (a,1+b),\ (1+a,b),\ (1+a,1+b). \tag{1}$$

Depending on the choice of a and b, we get different representations. *Which of them is the best?*

Similarly, when we add the 4-th dimension, we get a picture like this:



Here, 8 of 16 vertices have the same coordinates (1) as a 3-D cube. To describe the coordinates of the remaining eight vertices, it is sufficient to describe the coordinates (c, d) of the point connected to $(0, 0)$; then, the remaining 8 vertices have the following coordinates:

$$(c, d),\ (c, 1+d),\ (1+c, d),\ (1+c, 1+d),$$

$$(a+c, b+d),\ (a+c, 1+b+d),\ (1+a+c, b+d),\ (1+a+c, 1+b+d). \quad (2)$$

Again, a similar question appears: which values of a, b, c, and d lead to the "best" visualization of the 4-D cube?

2 "The Best" In What Sense? Formalizing the Problem

One of the main reasons why we want to represent a processor configuration – which is naturally represented in a multi-D space – on a 2-D page is that we want to *visualize* the configuration.

In general, when we project a multi-D space onto a 2-D plane, different points from the multi-D space may project onto the same point on a 2-D plane. It is therefore impossible to organize this projection in such a way that *all* different points from the original multi-D space map into different points on a 2-D plane. Luckily, we are interested only in *finitely many* points of the multi-D space: namely, we are only interested in the points corresponding to processors. If we restrict ourselves to finitely many points from a multi-D space, then, of course, it is possible to find a projection which maps all these points into different points on the plane. Hence, we must restrict ourselves only to such projections.

This restriction is important, but it does not solve the problem of selecting an appropriate projection, because many different projections satisfy this restriction. Which of these projections should we select?

A natural selection criterion comes from the following extension of this idea: for a better visualization, not only we want to avoid different points being projected into *the same* ones, but we should also avoid the situations when two different points are projected into two *very close* points on the plane. If this happens, then while we can distinguish between these two points in an ideal quality 2-D image, any minor distortion will make these points indistinguishable.

From this viewpoint, if, among N 2-D points $\{P_1, \ldots, P_N\}$ on the page, there is a pair of points $P_i \neq P_j$ which are too close, i.e., for which the distance $d(P_i, P_j)$ between these pints is too small, then this representation is not a very good one. Hence, as a measure of quality of a given 2-D representation, we can take the smallest of these distances, i.e., the following quantity:

$$\min_{i \neq j} d(P_i, P_j). \quad (3)$$

The smaller this quantity, the closer some points and thus, the worse the resulting representation. Therefore, at first glance, it seems like the larger this quantity, the better the representation.

This is not exactly true because, of course, we can always increase the above quantity (3) by simply "blowing up" the entire picture. It therefore makes more sense to consider, as a quality of a point configuration $\{P_1, \ldots, P_N\}$, not the *absolute* distances between the points, but rather the *relative* distances, i.e., distances that we get when we reduce different configurations to the same scale.

Since we are talking about the distances between the points, a natural choice of such a scale is the scale in which the largest distance between every pair of points is equal to 1. In other words, instead of the above quantity (3), we consider a modified quantity

$$\min_{i \neq j} \widetilde{d}(P_i, P_j), \tag{4}$$

where $\widetilde{d}(P, Q) = k \cdot d(P, Q)$, and the scaling parameter k is chosen in such a way that

$$\max_{i \neq j} \widetilde{d}(P_i, P_j) = 1. \tag{5}$$

From (5), we conclude that

$$\max_{i \neq j}(k \cdot d(P_i, P_j)) = k \cdot \left(\max_{i \neq j} d(P_i, P_j) \right) = 1,$$

hence,

$$k = \frac{1}{\max\limits_{i \neq j} d(P_i, P_j)},$$

and the criterion (4) takes the following form:

$$\frac{\min\limits_{i \neq j} d(P_i, P_j)}{\max\limits_{i \neq j} d(P_i, P_j)}. \tag{6}$$

We say that the configuration $\{P_1, \ldots, P_N\}$ is *the best* in a given family of configurations if it has the largest value of the quantity (6).

Now, we are ready to describe the results.

3 Main Results

3.1 3-D Case

The following is the best visualization of the cube:

Proposition 1. *Out of all possible configurations* (1)*, the best configuration corresponds to* $a = b = 0.5$.

Proof. Let us describe the proof, in detail, for the case when $0 \le a, b \le 0.5$. Due to symmetry, for all other possible cases (such as $0 \le a \le 0.5$ and $0.5 \le b \le 1$, $0.5 \le a \le 1$ and $0 \le b \le 0.5$, $0.5 \le a, b \le 1$, $-0.5 \le a, b \le 0$), the proof is similar.

In the case when $0 \le a, b \le 0.5$, we have $a \le 0.5 \le 1 - a$ and $b \le 0.5 \le 1 - b$. Thus, when we compare the distances between all possible pairs of points, we can easily see that the shortest distance is between the points (0,0) and (a, b), and this distance is equal to $\sqrt{a^2 + b^2}$.

Similarly, one can easily see that the largest of the distances is between the points (0,0) and $(1+a, 1+b)$, and this distance is equal to $\sqrt{(1+a)^2 + (1+b)^2}$. Thus, the quality of a configuration corresponding to the values a and b is equal to the ratio

$$\frac{\sqrt{a^2 + b^2}}{\sqrt{(1+a)^2 + (1+b)^2}}. \tag{7}$$

For $a = b = 0.5$, this ratio is equal to $1/3$. Thus, to prove that among all pairs (a, b) for which $0 \le a, b \le 0.5$, the best pair is indeed $(0.5, 0.5)$, we must show that

$$\frac{\sqrt{a^2 + b^2}}{\sqrt{(1+a)^2 + (1+b)^2}} < \frac{1}{3} \tag{8}$$

for $(a, b) \ne (0.5, 0.5)$. If we multiply both parts of the desired inequality (8) by the denominator of the left-hand side and square both sides, we get an equivalent inequality:

$$a^2 + b^2 < \frac{1}{9} \cdot \left((1+a)^2 + (1+b)^2\right). \tag{9}$$

To prove this inequality, it is sufficient to show that for every $x \in [0, 0.5]$, we have $x^2 \le \frac{1}{9} \cdot (1+x)^2$ – i.e., equivalently, that

$$x \le \frac{1}{3} \cdot (1+x), \tag{10}$$

and that for $x < 0.5$, we get an exact inequality. If we prove that, then, by combining the inequalities corresponding to a and b, we will prove the desired inequality (9).

This inequality (10) is equivalent to $\frac{2}{3} \cdot x \le \frac{1}{3}$, i.e., to $x \le 0.5$. So, (10) is true, hence (8) is true, i.e., for the case when $0 \le a, b \le 0.5$, the proposition is true.

Similar arguments show that the proposition is true for other cases as well.

3.2 4-D Case

To describe the best representation (1–2) for a 4-D cube, we start with an optimal cube representation – i.e., with the formula (1) with $a = b = 0.5$ – and

select c and d for which the quantity (6) attains the largest possible value. The resulting configuration turned out to be the best in this sense:

Proposition 2. *Out of all possible configurations* $(1-2)$ *with* $a = b = 0.5$*, the best configuration corresponds to* $c = -0.25$*,* $d = 0.25$*.*

The proof of Proposition 2 is similar to the above proof of Proposition 1.

4 Open Problems

In this paper, we have described the best representation for 3- and 4-D hypercubes. It would be nice to find out what is the optimal representation for 5-D hypercubes? 6-D? n-D for arbitrary n? What are the best 2-D representations of other multi-D constructions?

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, and by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-00-1-0365.

References

[1] Th. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, MIT Press, Cambridge, MA, and Mc-Graw Hill Co., N.Y., 1994.