

# Three-Dimensional Embedding of Binary Trees

Wolfgang W. Bein \*      Lawrence L. Larmore †      Charles Shields, Jr. ‡

I. Hal Sudborough §

## Abstract

We describe total congestion 1 embeddings of complete binary trees into three dimensional grids with expansion ratios 1.172 and 1.25. That is, we give a one-to-one embedding of any complete binary tree into a hexahedron shaped grid such that

1. the number of nodes in the grid is at most 1.172 (1.25) times the number of nodes in the tree, and
2. no tree nodes or edges occupy the same grid positions.

The first strategy embeds trees into cube shaped 3D grids. That is, 3D grids in which all dimensions are roughly equal, in size. The second strategy embeds trees into flat 3D grid shapes. That is, it maps complete binary trees into 3D grids with a fixed, small number of layers. An application suggests which embedding to use. For simulations in parallel computer environments, or possibly graph drawing, a cube shaped 3D grid is appropriate. For the sake of VLSI, or other graph drawing applications, embeddings with a small number of layers are better.

## 1 Introduction

Embeddings appear in the literature [8, 11] for the purpose of describing one of the following:

1. an efficient simulation of one parallel computer architecture by another,

\*Department of Computer Science, University of Nevada, Las Vegas, NV 89154-4019. Email: [bein@cs.unlv.edu](mailto:bein@cs.unlv.edu). Research supported by NSF grant CCR-9821009.

†Department of Computer Science, University of Nevada, Las Vegas, NV 89154-4019. Email: [larmore@cs.unlv.edu](mailto:larmore@cs.unlv.edu). Research supported by NSF grant CCR-9821009.

‡Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688 Email: [chasjr@utdallas.edu](mailto:chasjr@utdallas.edu). Research supported by a grant from the Girou Foundation.

§Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688 Email: [hal@utdallas.edu](mailto:hal@utdallas.edu).

2. an efficient method for using a parallel computer architecture to execute some standard computational process, or
3. to give area-efficient patterns for printing circuits on VLSI chips or wafers.

In an embedding, one has a *guest* graph  $G = (V, E)$  that represents the parallel architecture to be simulated, the computation graph to be mapped to the processors, or the circuit to be laid out. One also has a *host* graph  $H = (V', E')$  that represents the parallel computer architecture on which the computation is to be performed, or the positions for gates and routing patterns on a VLSI chip or wafer.

In this paper we consider embedding complete binary trees into three dimensional (3D) grids. The grid graph  $K_{a,b,c}$ , has  $a$  rows,  $b$  columns, and  $c$  faces, and can be defined to be the set of lattice points  $\{(x, y, z) : 1 \leq x \leq a, 1 \leq y \leq b, 1 \leq z \leq c\}$ .  $K_{a,b,c}$  has an edge between  $(p, q, r)$  and  $(s, t, u)$  if  $|p - s| + |q - t| + |r - u| = 1$ , that is, nodes of  $K_{a,b,c}$  are adjacent when their Euclidean distance is 1. Let  $T_h$  denote the complete binary tree of height  $h$  with  $2^{h+1} - 1$  vertices. We consider one-to-one, congestion one embeddings  $f$  of  $T_h$  into  $K_{a,b,c}$  for various values of  $h, a, b, c$ . Such an embedding is an injection assigning, to each vertex  $v$  of a tree  $T = T_h$ , a single vertex  $f(v)$  in a grid  $K = K_{a,b,c}$ , and assigning to each edge  $uv$  of  $T$  a path  $f(uv)$  in  $K$  between  $f(u)$  and  $f(v)$ , such that the internal nodes in  $f(uv)$  include neither  $f(z)$ , for any vertex  $z$  in  $T$ , nor any point in  $f(st)$  for any other edge  $st$  in  $T$ . In other words, such an embedding defines a subgraph of  $K$  which is homeomorphic to  $T$ . Such an embedding is commonly called a *layout*, and we shall use the two terms “embedding” and “layout” interchangeably. For an embedding  $f$  of  $T_h$  into  $K_{a,b,c}$  we define the *expansion ratio*  $r$  of  $f$ , to be the number of points in  $K_{a,b,c}$  divided by the number of vertices in  $T_h$ , namely

$$r = \frac{abc}{2^{h+1} - 1}$$

More generally, let  $f$  be an embedding from a guest graph  $G$  to a host graph  $H$ . The *dilation* of an edge  $uv$

under  $f$  is the length of the path  $f(uv)$ . The *dilation* of the embedding  $f$  is the maximum dilation of any edge of  $G$  under  $f$ . The *load* of a vertex  $x$  in  $H$  under  $f$ , denoted by  $load(x)$ , is the number of vertices mapped by  $f$  to  $x$ . The *congestion* of a vertex  $x$  in  $H$  under  $f$ , denoted by  $congestion(x)$ , is the number of paths of the form  $f(uv)$ , for an edge  $uv$  in  $G$ , containing  $x$  as an internal vertex. The *total congestion* of a vertex  $x$  is  $load(x) + congestion(x)$ . Note that, in this language, the embeddings we consider have total congestion 1.

The problem of embedding binary trees in two-dimensional (2D) grids has been studied extensively, although the objectives vary from paper to paper. Embeddings of the complete binary tree  $T_{2^p-1}$  into its optimum 2D square grid,  $K_{2^p, 2^p}$ , with load 1 were considered in several papers. An embedding with nearly optimum dilation, namely  $2 + (2^{p-1} - 1)/(p - 1)$ , is given in [6]. The vertex congestion of this embedding is  $\Omega((2^{p-1} - 1)/(p - 1))$ . Embeddings with vertex congestion 2 are given in [14] with dilation  $\frac{4}{3} \cdot 2^{p-1} + O(1)$  and in [7] with dilation  $2^{p-1}$ . Embeddings of trees into grids with small dilation are also the subject of several papers [1, 4, 13]. An example of the type of layout we consider, but for 2D grids, is the familiar H-tree layout [2]. It embeds complete binary trees of even height into square 2D grids, specifically,  $T_{2^p}$  into  $K_{2^{p+1}-1, 2^{p+1}-1}$ . The H-tree construction is an iterative approach. It begins with the trivial layout of  $T_0$  into  $K_{1,1}$  and, in general, combines four identical layouts of  $T_{2^p}$ , having an *escape path* (or free channel) from the image of its root to the periphery of the grid  $K_{2^{p+1}-1, 2^{p+1}-1}$ , by placing the four layouts in the shape of an H, where in the middle of the crossbar is the image of the root and the ends of the crossbar are the images of the root's children, together with the image paths of the edges incident to these vertices forming the crossbar and legs of the H. The previously mentioned escape path in this H-tree layout is from the image of the root and proceeds directly downward to the periphery. The H-tree construction uses about 50% of the area of the grid  $K_{2^{p+1}-1, 2^{p+1}-1}$  for the vertices of the tree, with the other 50% of the area of the grid used either to represent the edges of the tree or not used at all. In other words, the expansion ratio for embeddings given by the H-tree construction approaches 2 asymptotically.

One way to reduce the total space is to start with a larger tree than  $T_0$  and then construct, usually by *ad hoc* means, an embedding of this tree into a rectangular grid, not necessarily square, that is more space efficient than that given by the H-tree construction. Also, such efforts use more efficient recursive techniques than the H-tree recursion. This has been done in several papers

[3, 9, 12], with the last such paper [9] giving layouts of complete binary trees into 2D grids with expansion ratio bounded above by 1.4656. In this paper, we consider total congestion 1 embeddings of complete binary trees into 3D grids. So far as we know, such embeddings into 3D grids have not been the subject of any previous paper, although there are papers describing embeddings of complete binary trees into hypercubes [8] and butterfly networks [5]. Our techniques for embedding complete binary trees into 3D grids is, at least from a superficial perspective, somewhat similar to the usual techniques for embeddings into 2D grids. That is, it is an iterative procedure that obtains embeddings of the tree  $T_h$  from previous embeddings of  $T_{h-1}$ , together with initial embeddings of small trees. However, this process needs new ideas to work well for embeddings into 3D grids. For example, the H-tree construction, to create a 2D grid embedding of  $T_h$  from four copies of a previous embeddings of  $T_{h-2}$ , adds a new row and a new column to the four layouts of subtrees for the purpose of routing the connections between the roots of the subtrees. This is reasonable in 2D layouts, but the corresponding concept for 3D layouts, namely leaving an entire 2D plane between 3D layouts of subtrees, would be very space inefficient. We solve this problem by incorporating channels for routing connections in the layouts of the subtrees themselves. Other details are left to the formal description given in the Section 2. In Section 3, we give details for 3D embeddings with a fixed number of layers. The results, as already indicated, are total congestion 1 embeddings of complete binary trees into three dimensional grids with expansion ratios of 1.172 and 1.25.

## 2 The 1.25 Upper Bound

We let  $T_h$  be the complete binary tree of height  $h$ , and  $T_h^+$  be  $T_h$  together with one additional node, which we call the *escape point*, and an edge from that node to the root of  $T_h$ . For all  $h \geq 5$ , we will define five embeddings of  $T_h^+$  into  $K_{a,b,c}$ , where  $a = a_h$ ,  $b = b_h$ , and  $c = c_h$ , as defined below. We call these embeddings  $A_{a,b,c}$ ,  $B_{a,b,c}$ ,  $C_{a,b,c}$ ,  $D_{a,b,c}$ , and  $E_{a,b,c}$ . We let  $a_5 = b_5 = 4$  and  $c_5 = 5$ . For  $h > 5$ , the recursive definition of  $(a_h, b_h, c_h)$  is given by

$$\begin{aligned} a_h &= b_{h-1} \\ b_h &= c_{h-1} \\ c_h &= 2a_{h-1} \end{aligned}$$

The values of  $a_h$ ,  $b_h$ , and  $c_h$  for the first few values of  $h$  are found in the following table: Note that, for any

$h \geq 5$ ,  $K_{a,b,c}$  has exactly  $\frac{5}{4}$  as many nodes as  $T_h^+$ .

$h$	$a$	$b$	$c$
5	4	4	5
6	4	5	8
7	5	8	8
8	8	8	10
9	8	10	16
10	10	16	16
11	16	16	20

Our method is recursive. The base case is to define  $A_{4,4,5}$ ;  $B_{4,4,5}$ ;  $C_{4,4,5}$ ;  $D_{4,4,5}$ ;  $E_{4,4,5}$ . For  $h > 5$ , the five embeddings are defined recursively, using the embeddings of  $T_{h-1}^+$  already defined. The base case embeddings are illustrated in Figures 1 and 2. The recursive constructions are illustrated in Figures 4, 5, 6, 7, and 8.

**Nomenclature.** To facilitate the descriptions of the embeddings, we introduce a nomenclature for various parts of  $K_{a,b,c}$ .

We refer to the six faces of  $K_{a,b,c}$  (and, by implication,  $A_{a,b,c}$ , etc.) as *top*, *bottom*, *left side*, *right side*, *front end*, and *rear end*. The front end and rear end have size  $a \times b$ . The left side and right side have size  $a \times c$ . The top and bottom are the largest faces, of size  $b \times c$ . The twelve edges are given matching names, such as the *top left edge*, *front right edge*, etc.. The eight corner nodes have matching names, such as the *top left front corner*.

There is a node in the middle of the top, which we call the *top center node*. There is a node in the middle of the top left edge, which we call the *top left center node*. There may be no exact middle node. If  $b$  is even and  $c$  is odd, the top center node is the node just to the right of the middle. If  $b$  is even and  $c$  is even, the top center node is the node just to the right and just to the rear of the middle. If  $c$  is even, the top left center node is the node just to the rear of the middle of the top left edge. We also define the *alternative top center node*. If  $c$  is odd, the alternative top center node is equal to the top center node. If  $c$  is even, the alternative top center node is the node just to the front of the top center node. We refer to all nodes on the straight line path between the top left node and the top center node, including the top left center node, but not including the top center node, as the *top channel*. We refer to the set of nodes between the top left center node and the top left front corner, inclusive, as the *top left channel*. Note that the top channel has approximately  $\frac{b}{2}$  nodes, and the top left channel has approximately  $\frac{a}{2}$  nodes.

**Conditions.** Each of the five embeddings has *conditions*. For each of the five embeddings, the condition specifies that certain nodes of  $K_{a,b,c}$  must be *free* (i.e., not used for the embedding) and that the escape point of  $T_h^+$  must be at the top center node. An embedding of  $T_{h+1}^+$  into  $K_{b,c,2a}$  is obtained recursively from two copies of an embedding of  $T_h^+$  into  $K_{b,c,a}$ . The recursive process relies on specific conditions for each of the five embeddings.

- For all five embeddings, the escape point is placed at the top center node.
- In  $A_{a,b,c}$ , the left bottom edge is free.
- In  $B_{a,b,c}$ , the bottom front edge is free.
- In  $C_{a,b,c}$ , the top channel is free, and the right front edge is free.
- In  $D_{a,b,c}$ , the top channel is free, the top left channel is free, and the alternative top center node is free, unless it is equal to the top center node.
- In  $E_{a,b,c}$ , the top left center node is free, and the left front edge is free.

**Recursion.** We now describe the recursive constructions.  $K_{b,c,2a}$  (which we call the *mother cube*) is formed by two copies of  $K_{a,b,c}$ , joined at their top faces, after they are rotated or reflected and placed one behind the other. These are called the *daughter cubes*. We shall refer to the *rear* and *front* daughter cubes. The recursive steps are illustrated in figures 4, 5, 6, 7, and 8.

Embedded in each daughter cube is the *daughter tree*, a copy of  $T_h^+$ , and that embedding is specified by one of the five standard embeddings of  $T_h^+$  in  $K_{a,b,c}$ . If necessary, we refer to the *rear* or *front* daughter tree. The embedding of  $T_{h+1}^+$  in  $K_{b,c,2a}$ , which we call the *mother tree*, consists of the two daughter trees, together with some additional edges. The construction is defined below.

1. The following portions of the construction apply to each of the five cases.
  - (a) The escape point of the rear daughter tree is the root of the mother tree.
  - (b) The root of the mother tree is connected to the escape point of the front daughter tree by a path of length one or two. In the case of  $C_{b,c,2a}$ , that path consists of two edges; the middle node of the path is the alternative top

center node of the rear daughter cube. In the case of  $A_{b,c,2a}$ ,  $B_{b,c,2a}$ ,  $D_{b,c,2a}$ , and  $E_{b,c,2a}$ , that path consists of just one edge, as the two escape points of the daughter cubes are adjacent.

- (c) The top left center node of the rear daughter cube is the escape point of the mother tree.
- (d) There is a path from the root of the mother tree to the escape point of the mother tree, using the top channel of the rear daughter cube, which is free.

2. Specific details of each of the five embeddings:

- (a) The construction of  $A_{b,c,2a}$  is illustrated in Figure 4. The daughter embeddings are both  $C_{a,b,c}$ . The left side of the mother cube is the union of the front ends of the daughter cubes, and the top of the mother cube is the union of the left sides of the daughter cubes. The bottom left edge of  $A_{b,c,2a}$  is free because it is the union of the right front edges of the daughter cubes, both of which are free.
- (b) The construction of  $B_{b,c,2a}$  is illustrated in Figure 5. The rear daughter embedding is  $D_{a,b,c}$  and the front daughter embedding is  $A_{a,b,c}$ . The top of the mother cube is the union of the left side of the rear daughter cube and the right side of the front daughter cube. The left side of the mother cube is the union of the front ends of the daughter cubes. The bottom front edge of  $B_{b,c,2a}$  is the bottom left edge of  $A_{a,b,c}$ , which is free. The top channel of  $B_{b,c,2a}$  is the top left channel of  $D_{a,b,c}$ , which is free.
- (c) The construction of  $C_{b,c,2a}$  is illustrated in Figure 6. The rear daughter embedding is  $D_{a,b,c}$  and the front daughter embedding is  $B_{a,b,c}$ . The top of the mother cube is the union of the left sides of the daughter cubes. The left side of the mother cube is the union of the front end of the rear daughter cube and the rear end of the front daughter cube. The right front edge of  $C_{b,c,2a}$  is the bottom front edge of  $B_{a,b,c}$ , which is free. The top channel of  $C_{b,c,2a}$  is the top left channel of  $D_{a,b,c}$ , which is free.
- (d) The construction of  $D_{b,c,2a}$  is illustrated in Figure 7. The rear daughter embedding is  $D_{a,b,c}$  and the front daughter embedding is  $E_{a,b,c}$ . The left side of the mother cube is the union of the front ends of the daughter

cubes, and the top of the mother cube is the union of the left sides of the daughter cubes. The alternative top center node of  $D_{b,c,2a}$  is the top left center node of  $E_{a,b,c}$ , which is free. The top channel of  $C_{b,c,2a}$  is the top left channel of  $D_{a,b,c}$ , which is free. The top left channel of  $C_{b,c,2a}$  is the left front edge of  $E_{a,b,c}$ , which is free.

- (e) The construction of  $E_{b,c,2a}$  is illustrated in Figure 8. The rear daughter embedding is  $D_{a,b,c}$  and the front daughter embedding is  $B_{a,b,c}$ . The top of the mother cube is the union of the left sides of the daughter cubes. The left side of the mother cube is the union of the front ends of the daughter cubes. The top channel of  $E_{b,c,2a}$  is the top left channel of  $D_{a,b,c}$ , which is free. The left front of  $E_{b,c,2a}$  is the bottom front edge of  $B_{a,b,c}$ , which is free.

**Base Embeddings.** Figures 1 and 2 illustrate the base embeddings. In those figures, the top center node and all other nodes required to be free by the stated conditions are shown with small squares, unassigned (*i.e.* free) spaces are shown as unfilled circles, connections from a node in one plane to a node in a neighboring plane are suggested by arrows, and degree two points inserted into the tree during the embedding are shown as diamonds.

### 3 Layouts with a Fixed Number of Layers

Here we give an alternative layout strategy that starts with an embedding of  $T_6$  into  $K_{5,5,6}$ , and uses a recursive strategy that, for all integers  $m \geq 0$ , yields embeddings of  $T_{2m+6}$  into  $K_{5 \cdot 2^m, 5 \cdot 2^m, 6}$ . This results in an expansion ratio bounded by 1.172 and, more importantly, it yields a layout different from that previously given, as it preserves the initial number of layers. This could be of interest for applications, such as VLSI, where the number of layers is important.

We start with an embedding of  $T_6$  into  $K_{5,5,6}$ , which we illustrate in Figure 3.

The recursive procedure is described in Figure 9. Generally, for some fixed number  $c > 0$ , suppose we have layouts of  $T_h$  into  $K_{a,b,c}$  where in both layouts there is a free channel at fixed height, say  $d < c$ , in the front end and in the right side. We call the free channels in the front and right sides of these layouts *outside* channels. We call such a layout *type A* if there

is in addition a free channel from the image of the root to a position in the front end at height  $d$ , *i.e.*, within the outside channel. Similarly, we call such a layout *type B* if there is a free channel from the image of the root to a position strictly in the rear end at height  $d$ . Note that the embedding of  $T_6$  in  $K_{5,5,6}$  is of both type A and type B.

It follows that two copies of a layout of type A of  $T_h$  and two copies of a layout of type B can be put together, as shown in Figure 9 to create a layout of  $T_{h+2}$  into  $K_{2a,2b,c}$  of both type A and type B. That is, position four copies of the layout of  $T_h$  so that all four together form a hexahedron of height  $c$ , depth  $2a$ , and width  $2b$ . This is done by placing two type B layouts in front of two type A layouts, as shown in Figure 9, all four oriented in the same way, with their front ends forward and their bottom faces down. The rear left and the rear right copies are type A and the front left and front right copies are type B. The images of  $T_h$  in each type A layout can be joined with the image in each adjacent type B layout to create a layout of  $T_{h+1}$  into  $K_{2a,b,c}$ , by placing the image of the new root node in the outside channel of the type A layout and connecting it to the images of the roots of the two subtrees, using the free channels. Both layouts of  $T_{h+1}$  into  $K_{2a,b,c}$ , can then be combined into a layout of  $T_{h+2}$  into  $K_{2a,2b,c}$ , by placing the root of the whole tree in the node in the outside channel of the rear left layout that is common to both the front and right sides and connecting it to the images of its children using the outside channels. At this point one has a free channel to both the rear end and the front end of the hexahedral grid  $K_{2a,2b,c}$ , and this grid contains free channels at the same height  $d$  in its front end and right side. Thus, the derived embedding of  $T_{h+2}$  into  $K_{2a,2b,c}$  is simultaneously both a type A and a type B embedding.

The embedding of  $T_6$  into  $K_{5,5,6}$ , our current basis case, has an expansion ratio of  $150/128 = 1.172$ . This ratio remains constant as larger trees are embedded, since the sizes of both the tree and the grid increase by a factor of four as the recursive procedure is applied.

## 4 Conclusions

The recursive techniques we have described give space efficient methods to obtain total congestion one layouts of complete binary trees into 3D grids. The quality of the layouts obtained by these techniques are, however, heavily dependent on the quality of the initial, or base, layouts. At the moment we have used, respectively, layouts of  $T_5$  and  $T_6$  initially for the constructions in, respectively, Sections 2 and 3. Our work continues and

better initial layouts have already been found. This will yield improvements to the upper bounds given here.

A lower bound technique has been used [10] to show that any layout of a complete binary tree into a 2D grid requires an expansion ratio of at least 1.122. The lower bound argument is based on the fact that the number of nodes in a tree grows exponentially with its height, whereas the number of nodes in a 2D grid at distance  $d$  is quadratic, *i.e.*,  $2d^2 + 2d + 1$ . (The argument in the cited paper uses also some more sophisticated ideas.) The same argument can be applied to obtain a lower bound for the expansion ratio in laying out a complete binary tree in a 3D grid. However, as the number of nodes at distance  $d$  from a central point in a 3D grid is much larger than that in a 2D grid, *e.g.*, it is given by a cubic polynomial in  $d$ , the lower bound is not as large. In fact, using this technique we computed a lower bound of 1.0015. There is thus a large gap between the lower bound and the upper bound. In fact, as a 2D layout is a one layer layout, we note that going from one layer to six layers enables one to reduce expansion from 1.4656 [9] to 1.172, as we have shown here.

## References

- [1] S. N. Bhatt and S. S. Cosmadakis. The Complexity of minimizing wire lengths in VLSI layouts. *Information Processing Letters*, 25:263–267, 1987.
- [2] S. A. Browning. *The Tree machine, a highly concurrent computing environment*. PhD thesis, CIT, 1980.
- [3] B. Ducourthial and A. Merigot. Graph embedding in the associative mesh model. Manuscript, 1996.
- [4] A. Gregori. Unit-length embedding of binary trees in a square grid. *Information Processing Letters*, 31:167–173, 1989.
- [5] Ajay K. Gupta and Susanne E. Hambrusch. Embedding complete binary trees into butterfly networks. *Transactions on Computers*, 40(7):853–863, 1991.
- [6] R. Heckmann, R. Klasing, B. Monien, and W. Unger. Optimal embedding of complete binary trees into lines and grids. In *Proceedings of the 17th Graph Theoretic Concepts in Computer Science*, volume 570 of *Lecture Notes in Computer Science*. Springer, 1991.
- [7] M.-C. Heydemann, J. Opatrny, and D. Sotteau. Embedding complete binary trees into extended grids with edge congestion 1. In *Parallel Algorithms and Applications*, volume 805 of *Lecture Notes in Computer Science*. Springer, 1994.
- [8] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992.

- [9] Z. Miller, M. Perkel, D. Pritikin, and I. H. Sudborough. Expansions of layouts of complete binary trees into grids I - upper bounds. Submitted, 2000.
- [10] Z. Miller, M. Perkel, D. Pritikin, and I. H. Sudborough. Expansions of layouts of complete binary trees into grids II - lower bounds. Submitted, 2000.
- [11] B. Monien and I. H. Sudborough. Embedding one interconnection network in another. *Computing Supplement*, 7:257–282, 1990.
- [12] J. Opatrny and D. Sotteau. Embeddings of complete binary trees into grids and extended grids with total vertex-congestion 1. *Discrete Applied Mathematics*, 98:237–254, 2000.
- [13] M. S. Patterson, W. Ruzzo, and L. Snyder. Bounds on minimax edge length for complete binary trees. In *Proc. 23rd Symp. Theory of Computing*, 1991.
- [14] P. Zienicke. Embeddings of treelike graphs into 2-dimensional meshes. In *Proceedings of the Conference on Graph Theoretic Concepts in Computer Science*, volume 484 of *Lecture Notes in Computer Science*. Springer, 1990.

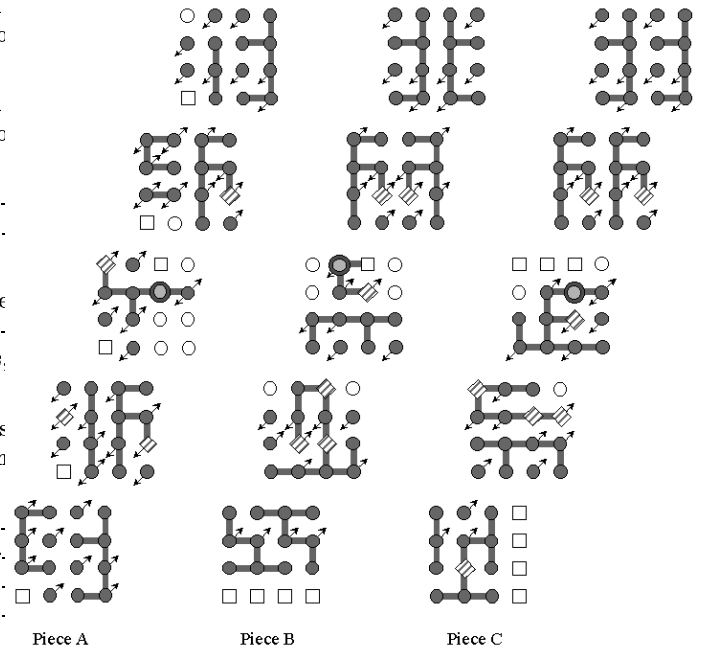


Figure 1: testing

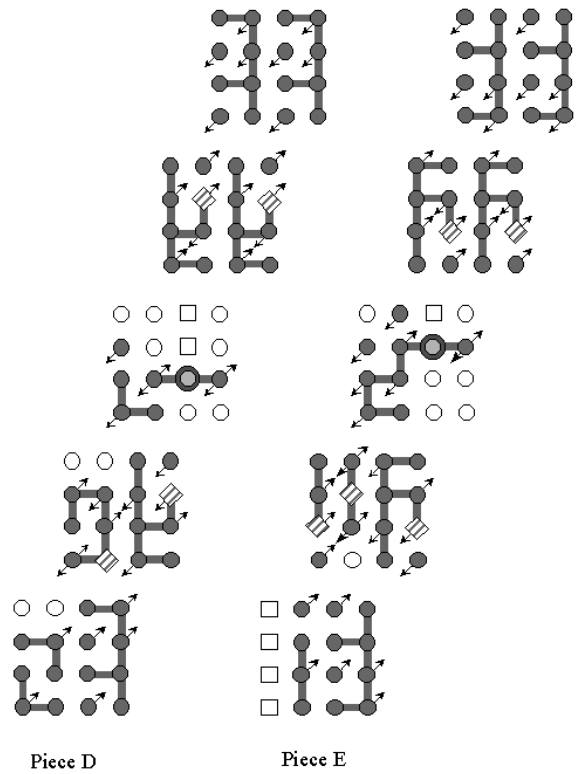


Figure 2: testing

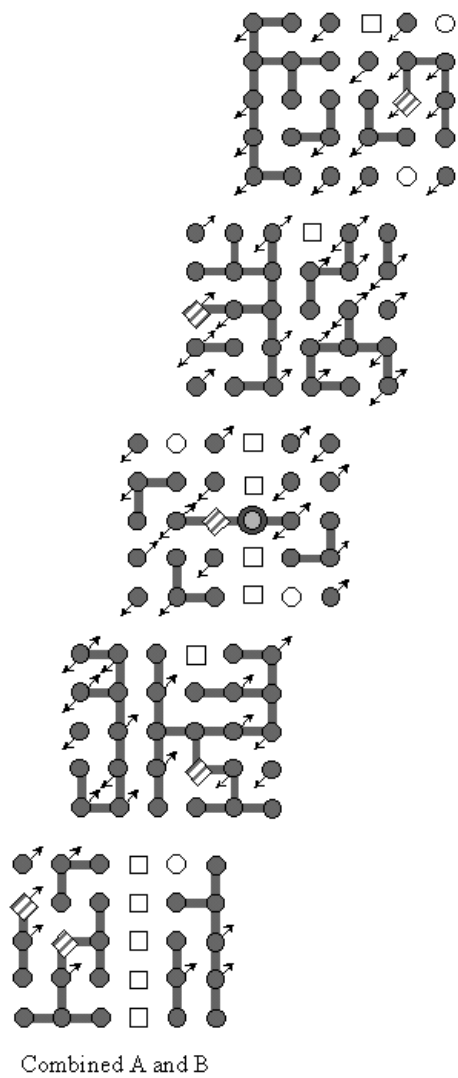


Figure 3: testing

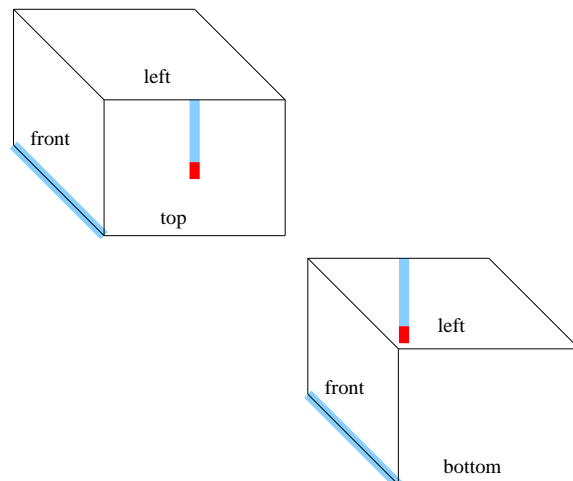


Figure 4:  $A_{b,c,2a} = C_{abc} + C_{abc}$

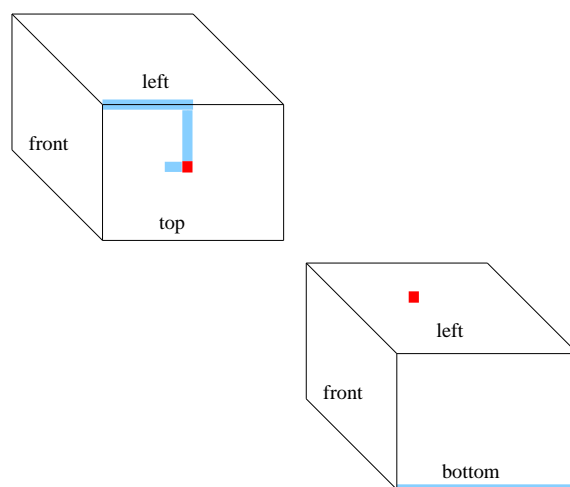


Figure 5:  $B_{b,c,2a} = D_{abc} + A_{abc}$

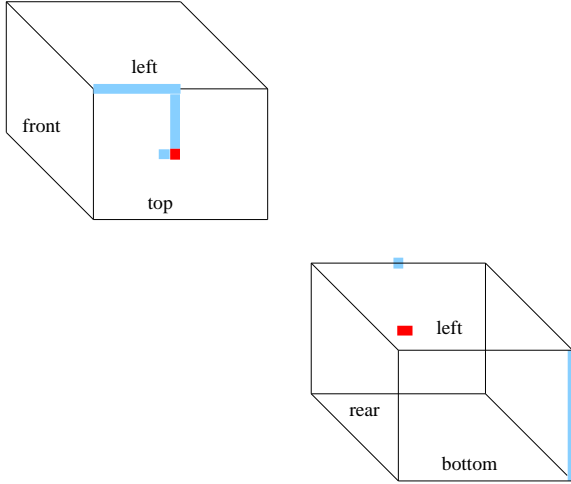


Figure 6:  $C_{b,c,2a} = D_{abc} + B_{abc}$

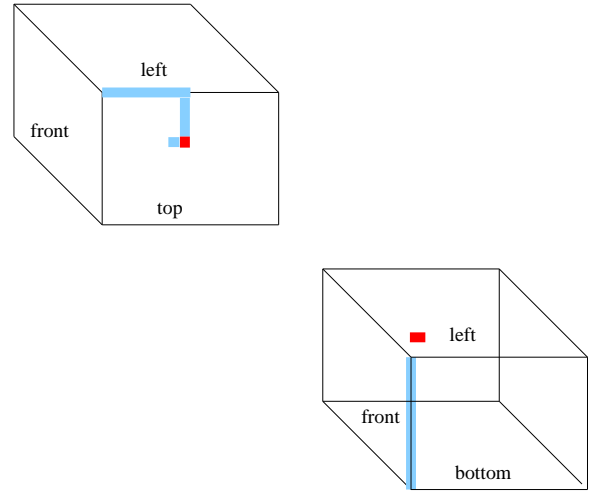


Figure 8:  $E_{b,c,2a} = D_{abc} + B_{abc}$

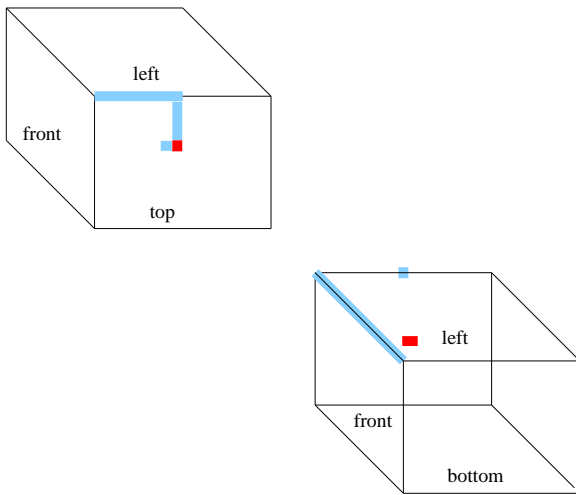


Figure 7:  $D_{b,c,2a} = D_{abc} + E_{abc}$

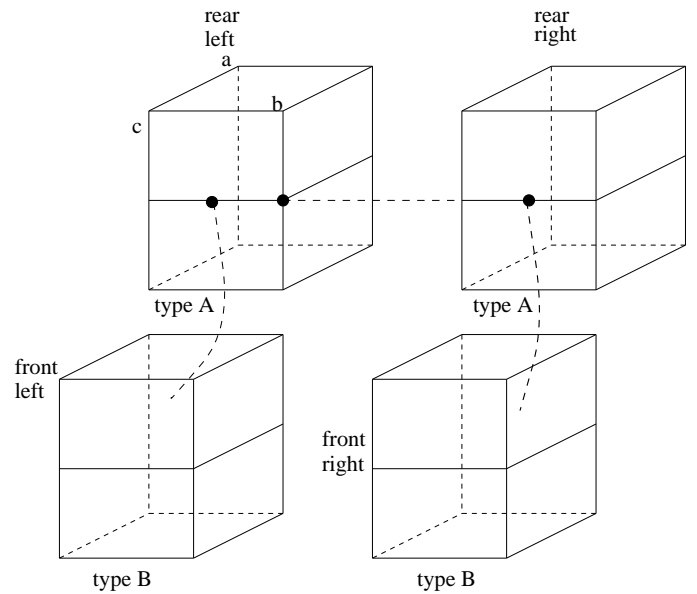


Figure 9: The recursive step.