INSTITUT D'ELECTRONIQUE FONDAMENTALE
Université Paris Sud

Graph Embedding in the Associative Mesh Model

Bertrand Ducourthial et Alain Mérigot

Rapport Interne IEF 96-01

Abstract

The Associative Mesh Model is a new parallel computing model whose basic data movements consist of global operations on data distributed on arbitrary connected components of processing nodes. The graph that defines these connected components is a dynamically defined subgraph of the physical interconnection graph, that is a 2 dimensional mesh with diagonals. We study in this paper the usefulness of the model to embed any graph.

We first define the one-to-connected-set-embedding, a particular case of one-to-many-embeddings. Then, we present the computing model and give hypothesis to obtain useful embedding according to the model assumptions. Then, we state that any graph can be one-to-connected-set-embedded, with unit vertex-congestion, edge-congestion and load, in a properly sized mesh. Last, the embeddings of a set of useful classical graphs — butterfly, hypercube, binary trees — are studied.

Chapter 1

Introduction

Interprocessor communication is a major issue in parallel computation. Communication latency and bandwidth are still the major limitations in multiprocessor performances, and the cost of the communication network is one of the most important factor of the price of parallel computers. To reduce these limitations, many works are done to study the properties of interconnection graphs, to define new message passing schemes, etc. (see for instance [1, 2]). To this purpose, fancy network topologies have been proposed, in order to limit the number of steps required to send messages between nodes in a parallel computer. While these topologies (hypercubes, fat trees, etc.) have very interesting properties and have been used in several commercial parallel computers, like the CM2 [3], the CM5 [4, 5], etc., they are mostly profitable for parallel computers with a large number of processing nodes, and, in that case, they exhibit an implementation hardware cost that is one of the major limitations to the development of these computers.

In the last years, a new concept has been introduced to overcome these limitations, namely reconfigurable architectures [6]. The basic idea of these architectures is to replace the expensive communication topologies and message passing hardware presently in use, by a cheaper scheme, relying on simple circuit switching techniques on a low cost topology (such as a 2 dimensional mesh). This way, long distance communications are realized by defining a physical path between the sender and the receiver of the message.

Several models have been proposed for reconfigurable architectures. For instance, in the *reconfigurable bus* model, every node is surrounded by a circular bus, and can independently connect every arc of this bus [6]. While this model has not already been implemented in hardware, some realization relying on a simplified scheme has been done. For instance, in the polymorphic torus, every node shares four arcs with its closest neighbors in a mesh, and acts as a switch to interconnect these arcs [7]. Ideally, this switch is a full crossbar, and any connection can be set between these arcs (even if practical implementations have limited the number of connection schemes). For instance, arcs North and South can be interconnected concurrently with the interconnection of East and West; or one port can be connected to all the others, etc. Some source nodes can send data on one or several of their ports, and this information will propagate along the path formed by the interconnected nodes.

While relying on a cheap interconnection network (compared to high degree topologies), reconfigurable architectures can exhibit very interesting performances, thanks to the present speed of integrated switches. These architectures rely on a simple and regular structure that can be easily integrated and lead to quite compact architectures with a large number of processors.

However there are two limitations to these architectures. First, they have a limited capability to perform concurrently communication and computations. Indeed, several massively parallel computing models allow as an atomic primitive operation to combine messages in order to perform either an accumulation of a set of messages on a destination place (send with collisions), or to distribute an operator over an ordered set of data (scans), and this has proved to be useful for an efficient implementation of a large number of algorithms [8, 9]. Moreover, either the local routing capability of every individual node is complete (i.e. every connection can be done on the processing elements) and accordingly quite expensive (at the hardware or control level), or the limited interconnection scheme reduces the total number of communications available, and the usefulness of the model.

In this paper, we will consider a new computing model — the Associative Mesh Model — that aims to overcome these limits. The main differences with the reconfigurable architecture are the following: first, the individual message switching capabilities of every node are quite limited, but the underlying topology adds diagonal connectivity to the regular mesh structure to overcome these limitations. Secondly, the basic communication primitives allow to apply on the fly a combining operator to the routed data.

In section 2, we will describe the underlying model more precisely, define the specific graph embedding scheme that we will use, and give some other definitions that will be used all over the paper. Section 3 states the existence, for any graph, of an embedding on a properly sized mesh, with unit edge congestion, vertex congestion and load. The last sections are concerned with the embedding of classical regular graphs: section 4 presents results for butterfly embedding, section 5 is about hypercubes embedding and section 6 presents some schemes to embed binary trees within the model.

Chapter 2

Basic definitions

2.1 Graph related definitions

In this section, we give a set of graph oriented definitions useful for the rest of the paper.

We call *digraph* $G(V, E)$ where V is a set of *vertices* and E a binary relation on $V \times V$, that is the set of *edges*. When this relation is symmetric, it is possible to replace each pairs of edges $\{(v_1, v_2), (v_2, v_1)\}$ by a single one, said to be *undirected*, that holds for (v_1, v_2) and (v_2, v_1). We then obtain a *graph* instead a digraph. Two vertices that share an edge are said to be *adjacent*. We call *out-degree* of a vertex $v \in V(G)$, noted $\delta^+(v)$, the number of edges leaving v (edges (v, v'), $v' \in V(G)$). In the same way, $\delta^-(v)$ denotes the *in-degree*, which is the number of edges oriented towards v (edges (v', v), $v' \in V(G)$). The *degree* δ of a vertex in a digraph is the sum of its in-degree and its out-degree. For a graph, we call *degree* of v, noted δ, the number of incident edges of a vertex v. A *path* between the vertices u and v is a sequence of edges (u_n, u_{n+1}), $0 \leq n < m$, such that $u_0 = u$ and $u_m = v$. m is called the length of the path. The *(strong) connected components* of a (di)graph are the sets of vertices such that there exists a path between all pairs of vertices of the same component, but there exists no path between two vertices of different components. A *connected set* S of $G(V, E)$ is a set of vertices such that, for all vertices v_1, $v_2 \in S$ there exists a path from v_1 to v_2 where all intermediate vertices are in S. A connected component is a connected set, but several connected sets can belong to a same connected component. The *distance* from vertex u to vertex v of V is the minimal length of all paths from u to v in G. The *diameter* of a connected set/component of G is the maximum of the distances between pairs of vertices in the connected set/component. The *set of the ancestors*, or simply *the ancestors*, of a vertex $v \in V$, denoted by $\Gamma_G(v)$ is the set of all the vertices $u \in V$ such that there exists a path from u to v. If G is a graph (ie. undirected), $\Gamma_G(v)$ is the connected set of vertex v. A *spanning subgraph* or *subgraph* of the (di)graph $G(V, E)$ is a (di)graph $G'(V, E')$ where

$E' \subset E$. Last, a graph $G(V, E)$ is said to be *bipartite*, if there exists two subsets of V, V_1 and V_2, with $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$, such that every edge e of E links a vertex $v_1 \in V_1$ and a vertex $v_2 \in V_2$ together.

A $N \times N$ *4-mesh* is a digraph defined as follows (see figure 2.1). Each vertex is numbered by a couple (i, j), $0 \leq i, j < N$ ($N \in \mathbb{N}$). Vertex (i, j) has 4 incoming and outcoming edges from and to vertices $(i, j-1 \; mod \; N)$, $(i+1 \; mod \; N, j)$, $(i, j+1 \; mod \; N)$ and $(i-1 \; mod \; N, j)$.

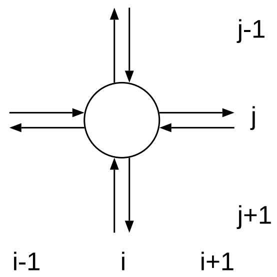

Figure 2.1: Vertex in a 4-mesh.

A $N \times N$ *8-mesh* is a digraph similar to 4-mesh but extended with diagonal edges (see figure 2.2): vertex (i, j) has 8 incoming and outcoming edges from and to vertices $(i, j-1 \; mod \; N)$, $(i+1 \; mod \; N, j-1 \; mod \; N)$, $(i+1 \; mod \; N, j)$, $(i+1 \; mod \; N, j+1 \; mod \; N)$, $(i, j+1 \; mod \; N)$, $(i-1 \; mod \; N, j+1 \; mod \; N)$, $(i-1 \; mod \; N, j)$ and $(i-1 \; mod \; N, j-1 \; mod \; N)$.

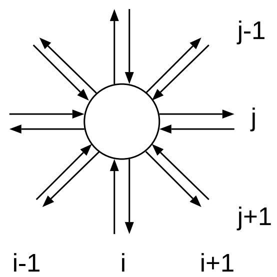

Figure 2.2: Vertex in a 8-mesh.

Theses topologies are also called a 4-connected or 8-connected 2 dimensional torus. In this paper, we won't use torus particularity and we just consider a mesh obtained from this torus removing edges between the first and last rows, and between the first and last columns. For example, figure 2.3:b shows a 5×3 8-mesh.

An *embedding* (f, P_f) of a digraph $G(V_G, E_G)$ into a digraph $H(V_H, E_H)$ is defined as follows: f is an application from V_G into V_H which send all vertices of G on vertices of H, and P_f is an application from edges of G on paths of H such that each edge (x, y) of G is mapped on a path linking $f(x)$ to $f(y)$ of H.

G is called *guest* or *logical* graph, while H is called *host* or *physical* graph. We define now a special case of an embedding of a graph into another, adapted to the studied computing model. Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be two graphs (the same definition holds for digraphs). A *one-to-connected-set embedding* — or otcs-embedding for short — (f, P_f) of G in H is given by an application f from V_G to the set of the connected sets of H, and by an application P_f from E_G to the set of paths of H, that associates to each edge (x, y) of G a path connecting a vertex of $f(x)$ to a vertex of $f(y)$. This is a special case of a one-to-many embedding, where sets representing a vertex of the guest graph G in the host graph H, are supposed to be connected.

An example of such an otcs-embedding is showed in figure 2.3. Figure 2.3:a presents the logical graph, while figure 2.3:b shows one of its embeddings in an undirected 8-mesh, which is the host graph in this example.

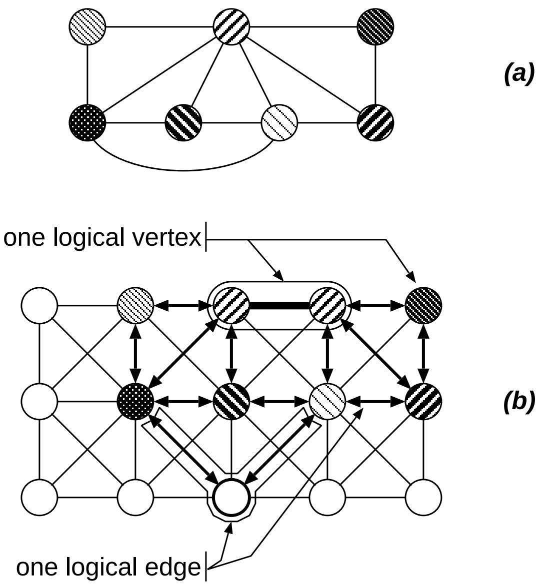

Figure 2.3: One-to-connected-set embedding example.

The quality of an embedding is traditionally evaluated by a set of measures. We adapt here the definitions of these classical parameters, defined among others in [10], to otcs-embeddings. The *expansion* of an embedding is the ratio of the number of nodes in H, to the number of nodes in G. The *edge congestion* of an embedding is the maximum over all edges $e \in E_H$, of the number of paths of $P_f(E_G)$ to which e belongs. The *vertex congestion* is the maximum number of non-empty intersections between connected sets of $f(V_G)$ and paths of $P_f(E_G)$. The *load of a vertex* v of H resulting from the embedding (f, P_f), is the number of connected sets of $f(V_G)$ containing v. The *load of the embedding* (f, P_f) is the maximum over all the loads of the vertices of H. The *dilation* is the maximum length of all paths in $P_f(E)$.

The expansion measures the required number of hardware nodes to embed a given graph. Most of the time, people try to minimize this number, that corresponds to an effective hardware cost, and expansion is frequently expected to be of the order of 1. To achieve this, it is most of the time required to affect several nodes of G to one node of H (load $\gg 1$) and to share some edges of H between several paths of $P_f(E)$ (edge congestion $\gg 1$). The vertex congestion measures the number of paths representing edges of G in H which go through vertices of G in H. The dilation gives an indication on the number of needed steps to achieve a communication phase.

Next sections describe the underlying computing model, the *Associative Mesh Model*. We first present the Associative Net Model, the basic topology-free formal model then, the AMM, that is an instance for 8-meshes.

2.2 The Associative Net Model

Given a set of computing nodes, interconnected according to digraph $\mathcal{H}$, the *Associative Net Model* — or ANM for short — is a new computing model whose communication primitives rely on the global manipulation of data on nodes belonging to connected components of some subgraph of $\mathcal{H}$.

The digraph $\mathcal{H}$ is defined as follows: its vertices are the nodes and its edges are the links between nodes. Moreover, $\mathcal{H}$ is supposed to be defined by a symmetric binary relation, that is if edge (u, v) exists in $\mathcal{H}$, then edge (v, u) also. $\mathcal{H}$ is called the *physical digraph*. Let the nodes be numbered from 0 to $P-1$. Thus, P is the number of nodes: $P = |V(\mathcal{H})|$. A *parallel variable*, called *pvar*, is a P-uplet of $\mathbb{N}^P$, distributed on the nodes such that each node stores one of its members.

The basic objects used for primitives are subgraphs of $\mathcal{H}$, which are directed as $\mathcal{H}$. A subgraph is totally described by defining in every vertex of $\mathcal{H}$ the subset of the edges of $E(\mathcal{H})$ which are incident to this node in the subgraph, that allows to code a subgraph by a d-bits pvar, where d represents the in-degree of $\mathcal{H}$. This particular pvar is called an *mgraph*. We note $\mathcal{M}_{\mathcal{H}}$ the set of all mgraphs corresponding to (directed) subgraphs of the physical digraph $\mathcal{H}$.

One of the basic operations that can be realized on a subgraph is an *association*. Given an associative and commutative operator $\otimes$, an mgraph $m \in \mathcal{M}_{\mathcal{H}}$ and a pvar $p = (p_0, \cdots, p_{P-1})$, the $\otimes$-*association*(m, p) is a function $(\mathcal{M}_{\mathcal{H}}, \mathbb{N}^P) \mapsto \mathbb{N}^P$:

$$\otimes\text{-}\mathit{association}(m,p) = \left\{ \begin{array}{l} a_i : 0 \leq i < P \\ \text{with} \\ a_i = \bigotimes_{j \in \Gamma_g(i)} p_j \end{array} \right\}$$

In other words, $\otimes$-*association*(m, p) will return a pvar whose value in vertex v is equal to the iterated application of operator $\otimes$ to pvar p in all the ancestors of v. Operators we will consider are logical functions (OR, AND, XOR), maximum, minimum and addition.

The associativity of the operator is necessary to perform an association in parallel. Moreover, incoming edges of a nodes are not ordered and we then need commutativity. When the subgraph degenerates to an oriented chain of nodes, an association is equivalent to a parallel prefix computation (scan) [8]. Since all subgraphs are allowed, and not only paths, the Associative Net Model generalize the Scan Vector Model.

Associations are powerful parallel operators that apply on the fly an operator to the routed data. They can also perform standard communications primitives, such as broadcasting, accumulation or gossiping [11]. For example, to compute an accumulation on the node v with addition, we span in $\mathcal{H}$ a leaves-to-root oriented tree rooted in v, and we realize a $+$-*association* on this mgraph. After this parallel calculus, the value in vertex v is the sum of values of all the vertices of $\mathcal{H}$.

2.3 The Associative Mesh Model

While the formal ANM can be applied to any graph, it is most likely to be useful for low-cost topologies with a limited degree, where its global processing capabilities can overcome the limitations of the topology. We will now present such an instance, the *Associative Mesh Model* (AMM for short), that relies on an 8-mesh. We first discuss the realizability of such a structure and present rapidly characteristics of an hardware implementation in section 2.3.1, then, we compare the capabilities of the AMM to reconfigurable architectures in section 2.3.2.

2.3.1 Hardware implementation of the AMM

An hardware implementation of the AMM has been studied, namely the *Orsay Associative Mesh (OAM)* [12]. Key aspects of the architecture are the following (see figure 2.4 that emphasizes on the association data-path). In order to implement associations, processing nodes have the ability to apply on the fly a combining operator to data arriving from their eight neighbors. The available operators at the hardware level have the peculiarity that 0 is the identity element, providing a simple mean to consider a subset of the neighborhood thanks to the so-called *mask register*, that masks out to zero data from arcs that do not belong to the current mgraph. This way, data manipulation required to implement any association on any mgraph is an atomic operation at the node level, and the global computation on a connected set can be done by combining neighbor combination with the locally stored data, and by rediffusing asynchronously the result. After some time, the network will evolve to a steady state, and, when this state is reached, the final result of the association is obtained in every node. Distributed stability detectors are used to check for this completion. Thanks to the speed of present MOS technology, association computation time can be in general faster than point to point data communication in most parallel computers, even for a machine with a very large number of processors (10K or more). This is why we will consider them as unit time. Other details on the OAM can be found in [13].

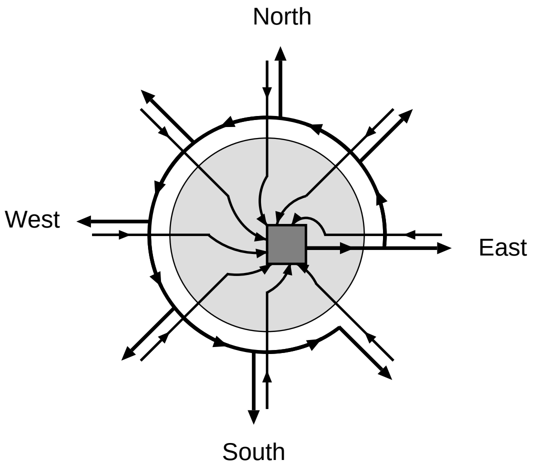

Figure 2.4: Modeling the communication node.

We can note a difference between the formal model definition and its actual implementation: if the considered operator is not idempotent (like the addition), one must take care not to take into account a data several times. In the OAM, this is provided by a primitive called *spanning-tree*, that insures this uniqueness by forcing following computations to take place along a directed tree. We will not consider this distinction in this paper.

2.3.2 Comparing the AMM with reconfigurable architectures

The associative mesh model shares some similarities with reconfigurable architectures. Both rely on a cheap mesh based topology instead of more complex topologies and use circuit switching techniques for atomic communication primitives. One of the main difference lie in the possibility to apply a global operator over a connected set on nodes in the associative mesh. Besides its computational aspects, this capability can also be used to distribute a piece of data along any path, as in reconfigurable architectures. For instance, to perform a broadcast in an associative mesh, we span in $\mathcal{H}$ root-to-leaves trees rooted on emitting (source) nodes s_i — containing data d_i to distribute — such that vertices which should receive data d_i are on the tree where root is v_i (see figure 2.5, where source nodes are hatching). Let call m this subgraph (mgraph). Every s_i will belong to one connected component of m, and, data diffusion of d_i by node s_i will be equivalent to compute the OR-*association* over mgraph m of a pvar $P = \{p_i\}$ defined by

$p_i = s_i$ if s_i is a source node and $p_i = 0$ elsewhere. This way, an OR-*association* in the AMM is strictly equivalent to a broadcast in a reconfigurable architecture.

However, there are some differences as well. First, this model that mixes communication and computations is much more powerful to implement algorithms — provided it is possible to embed a graph in the topology. A large number of algorithms can be expressed with this computing model [14]. The basic primitives allow to distribute an operator over data spread along any connected set of an undirected or directed subgraph of the physical communication graph, that cannot be done in any other computing model. Compared with reconfigurable architectures, this leads to an interesting possibility to merge communications and computations, but, in terms of communication flexibility, it is much more limited.

Indeed, reconfigurable architectures allow paths to cross each other on every vertex. This behavior is forbidden in the AMM, where each vertex can only combine data from a subset of its physical incoming edges and redistribute it, or transfer informations from one of its neighbors as a special case. To overcome this limited behavior and to allow non-planar graphs embedding, AMM extends the 4-mesh topology with diagonals. This way, path crossing can take place on diagonal edges (see for instance figure 2.5).

The purpose of this paper is to study the ability of this structure to embed graphs, either irregular or regular. Before that, we explain what kind of embeddings is useful for the AMM.

2.4 AMM-restrictions for embedding

Underlying assumptions of the embedding under the AMM are slightly different from usual: load, edge congestion and vertex congestion must be equal to one. These hypotheses are necessary to allow association computation in one phase. In others words, a node of the 8-mesh may either be member of a connected set, image of a single vertex of the embedded graph, or it can be a vertex of a single path $P_f(e)$, image of an edge of the logical graph.

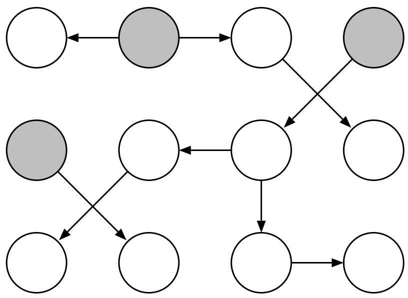

Figure 2.5: Broadcasting in one step in the AMM.

Because edge congestion must be equal to one, one-to-many embeddings are useful to overcome the limited degree of nodes in the 8-mesh, which is 8. Such an embedding is efficient in the AMM if images of logical vertices are connected sets: this way, in a single association, global information on logical nodes will be obtained. So, we are interested by otcs-embedding with edge congestion, vertex congestion and load equal to one.

This will lead to an expansion that is generally $\gg 1$, and to large dilations. The underlying assumptions are that an hardware AMM implementation will be done by means of (almost) cost-less tiny processing elements, and, while we will try to minimize expansion, large expansions are not as critical as in other architectures.

While dilation is generally an important measure of the performances of an embedding (in terms of communication time), it is no longer true for the AMM, where all operations on a connected set (including communication) are supposed to be unit time.

Last, when a graph as been embedded, basic primitives of the formal associative net model (associations) can be used on it. This can be realized by means of associations on the otcs-embedded graph, and of data diffusion techniques described above.

Chapter 3

One-to-connected-set embedding in the 8-mesh

In this section, we show that all graphs can be otcs-embedded in an appropriate 8-mesh under the AMM assumptions. To do so, we use some basic results about "row-to-row routing". First, this routing is defined and we study its properties. Then, we give a theorem that proves the existence of an otcs-embedding for any graph, in a large enough 8-mesh, which respects restrictions given in section 2.4.

3.1 Routing a row to another row in the 8-mesh

In this section, we define a routing from a row of the 8-mesh to another row. Of course, it will be the same for a routing from a column to another column.

3.1.1 Overview

Let us consider an n vertices row in a 8-mesh whose vertices are numbered from 0 to $n-1$. We assume that each vertex stores a couple *(value, destination column address)*; such a couple is called a *message*. Our goal is to send each of these *values* to a vertex whose column is *destination column address* in another row. We assume that every destination column will receive a unique *value*. To do that, we will try to find a set of disjoint paths that lead directly from source vertices to destination vertices, so that the considered routing can be performed in a single step using AMM. Thus, nodes of the AMM can be source nodes, sink nodes or intermediate nodes used to achieve the interconnection paths. We will call this routing "row-to-row routing in one phase" in the following. Obviously, several rows of the 8-mesh are necessary to achieve this routing, but we are interested to reduce the number of these intermediate rows. Figures 3.2 and 3.3 of this subsection illustrate this kind of routing and present the defined paths. On these figures, only the destination addresses are given: values are omitted for simplicity.

Every intermediate node in row i will transmit data from one of the nodes in row $i-1$ from which it has an incoming link. It is easy to show that, to keep the number of messages unchanged, the basic operation must be to swap pairs of neighboring vertices. Otherwise, some nodes would receive no data, and others several values, that leads to undisjoint paths. So, given a pair of nodes in row i, (i,j) and $(i,j+1)$, corresponding nodes $(i+1,j)$ and $(i+1,j+1)$ in row $i+1$ will either receive directly or exchange values from this pair. This basic operation is used by all routing algorithms able to realize row-to-row routings. These algorithms differ only in the choice of the pairs to swap.

3.1.2 Odd-even algorithm revisited

The odd-even algorithm allows to sort a line of n values numbered from 0 to $n-1$, by means of neighbor swapping (see [2]). At step $2p+1$ (odd step), we consider the *odd family of pairs* defined as $(2k, 2k+1)$ for all $k \in \{0, .., \lfloor \frac{n-2}{2} \rfloor\}$, and at step $2p$ (even step), the *even family of pairs* defined as $(2k+1, 2k+2)$ for all $k \in \{0, ..., \lfloor \frac{n-3}{2} \rfloor\}$. During a step, the pairs of the considered family can swap their values. Given a pair (α, β), there is a swap if and only if α is greater than β. In this case, this pair becomes (β, α) (see figure 3.1).

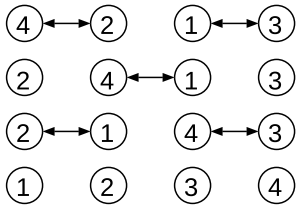

Figure 3.1: Example of odd-even sort.

We will show how to adapt this classical sorting algorithm to our row-to-row routing problem.

3.1.3 Row to row routing by means of the odd-even routing

We can easily adapt this algorithm to the row-to-row routing problem. Defining the paths used to route the n messages *(value, number of the destination column)* is equivalent to determine the successive data swaps required to sort the destinations. Indeed, every step of the odd-even sort corresponds to send the values from the current row to the next row in the routing. Swapping a pair in the odd-even sort consists to cross the paths in the routing, and, if no swapping is required, the values are sent in the same column to the following row. To complete this adaptation of odd-even algorithm, we may choose the family of pairs (odd or even) which will be considered for the first exchange, to (marginally) optimize the number of rows. This way, we have an algorithm for the routing problem - we call it *odd-even routing* - and some results of the odd-even sort can be adapted.

The first result concerns the completion of the routing problem. In [2], Leighton showed that the odd-even algorithm can sort any line of n values in n steps, which gives the upper bound. Thus, routing an n messages row to another row is always feasible by means of $n-1$ intermediate rows, that leads to an $(n+1)\times n$ 8-mesh. This bound is reached for some peculiar configurations, like "reverse permutation", as shown in figure 3.3.

Given a message configuration, a lower bound is easily determined for this algorithm: it is equal to the longest path that a value must follow to reach its destination column. This lower bound is then computed with the formula: $MAX_{i=1}^{n}|i - destination(i)|$. Figure 3.2 presents a routing achieved with the lower number of rows.

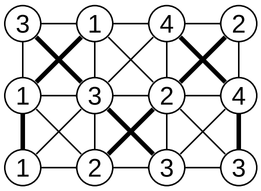

Figure 3.2: Row-To-Row routing with lower bound number of row.

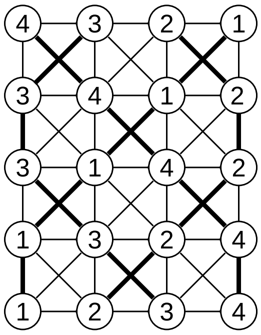

Figure 3.3: Some configurations can only be routed with a number of row equal to the upper bound.

Unfortunately, some configurations cannot be routed with this lower bound, and require a larger number of additional rows. For instance, consider the data configuration of figure 3.3. The largest distance between a message and its destination is 3, and the routing should be performed in 4 rows, according to the lower bound. But, the configuration is such that it is impossible to reduce simultaneously the distance of these messages to their destination, and a fifth additional row is required. On figure 3.3, first row, value 3 cannot progress in its direction since value 4 is more distant, and, hence, has priority. Such a local configuration is called a *collision* below.

To conclude, the row-to-row routing problem in one phase has always a solution on a properly sized 8-mesh. The lower bound is $MAX_{i=0}^{n-1}|\, i - destination(i)|$ for a given configuration and the upper bound is $n+1$ for a row of n values.

3.2 Existence theorem

In this subsection, we prove that every graph can be otcs-embedded in a properly sized 8-mesh respecting AMM-restrictions (see 2.4). The proof is based on the row-to-row routing in one phase. We call $G(V, E)$ the logical graph to embed in the 8-mesh.

Theorem 1 (existence theorem) *There exists a one-to-connected-set-embedding of any graph $G(V, E)$, having edge-congestion, vertex-congestion and load equal to one, in a $M \times M$ 8-mesh, where*

$$\begin{aligned} M &= \textstyle\sum_{v\in V}(\delta^-(v)+1) \\ &= \textstyle\sum_{v\in V}(\delta^+(v)+1) \end{aligned}$$

Proof. The proof is given for a digraph but a similar proof can be established with a graph (undirected). Thus, let us assume that $G(V, E)$ is a digraph.

We associate to each vertex v of G a set of $(\delta^+(v)+1)$ consecutive vertices in a line A of the 8-mesh. Line A is then composed with $M = \sum_{v\in V}(\delta^-(v)+1)$ vertices of the 8-mesh. We do the same operation with the input degrees and we obtain another line B of $M = \sum_{v\in V}(\delta^+(v)+1)$ vertices (lines A and B have the same length). We then arrange theses two lines in the rightmost and leftmost position of a 8-mesh, such that they are parallel, with $M-2$ lines between them (see figure 3.4).

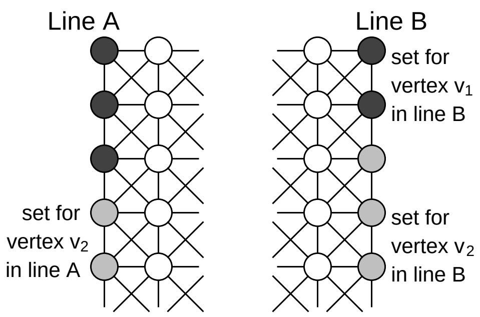

Figure 3.4: Lines A and B in the 8-mesh.

Connecting the vertices of line A to vertices of line B with disjoint paths is always feasible, for every possible configuration, as shown in the preceding subsection, using row-to-row routing. So, it is possible to associate, for each edge $(v_i, v_j) \in E$ a path from the set corresponding to v_i in line A to the set corresponding to v_j in line B. After that, all these sets still have a free vertex. Thus we can connect the sets corresponding to the same logical vertex in lines A and B (see figure 3.5).

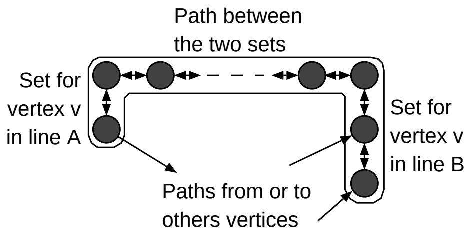

Figure 3.5: Image of a vertex of G in the 8-mesh.

Afterwards, we consider the image of a vertex v of G, by the embedding, as the set formed by the vertices representing v in lines A and B and the ones used to realize the path which connects these two sets. A vertex of G is then represented by several vertices of the 8-mesh (see figure 3.5). The image of any edge (v_i, v_j) of G is the path given by the routing from the set of vertices in line A representing v_i to the set of vertices representing v_j in line B. The embedding is vertex congestion 1 by construction. It is edge congestion 1 since the routing of line A to line B is feasible in one phase under our assumptions. Its load is 1 by construction.

Thus, we have obtained the desired embedding, having load, vertex congestion and edge congestion equal to one. The size of the needed 8-mesh for the embedding of $G(V, E)$ is given by the number of vertices in the lines A and B, that is $\sum_{v \in V}(\delta^-(v) + 1)$. The routing of line A to line B should need M steps (and thus $M - 1$ intermediate lines between lines A and B), but it is always feasible to have a straight path from line A to B at the extremities of theses lines (see figure 3.6). Thus, the maximal routing distance is $M - 2$ and the upper bound of this routing is smaller than M. That gives the result[1]. ■

The construction of the embedding explained above is illustrated by an example in figure 3.6, for a digraph. For a graph (undirected), we consider every edge of this graph as a pair of opposite oriented edges and we use the same technique.

The upper bound on the expansion of the embedding given by the theorem is $\frac{(|V|+|E|)^2}{|V|}$. It can lead to the use of large meshes. The following sections try to optimize the embedding of some classical regular graphs.

[1]We should consider a $M \times (M - 1)$ 8-mesh in fact.

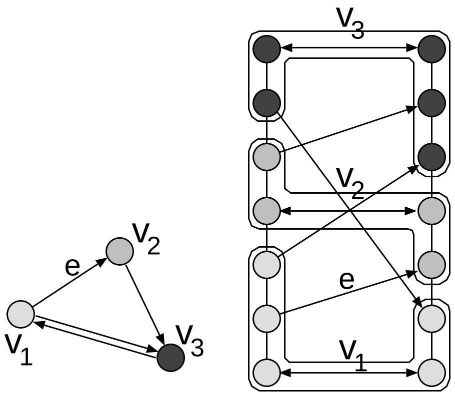

Figure 3.6: Example of an otcs-embedding in the 8-mesh, with unit load, edge congestion and vertex congestion.

Chapter 4

Butterfly embedding

In this section, we show that a better embedding than the one given by the existence theorem exists for the butterfly graphs in the 8-mesh. This embedding has a better expansion and it is simpler to build and to use.

4.1 Overview

4.1.1 Definition

The butterfly graph of dimension n, denoted as $B(n)$, is a recursive multi-layer graph. Nodes of $B(n)$ are represented by couples $(l, x_{n-1} \cdots x_1 x_0)$, with $l \in \{0, \cdots, n\}$ and $x \in \{0, 1\}$. Node $(l, x_{n-1} \cdots x_l \cdots x_1 x_0)$ is adjacent to nodes $(l + 1, x_{n-1} \cdots x_l \cdots x_1 x_0)$ and $(l + 1, x_{n-1} \cdots \bar{x}_l \cdots x_1 x_0)$, $\forall l : 0 \leq l < n$.

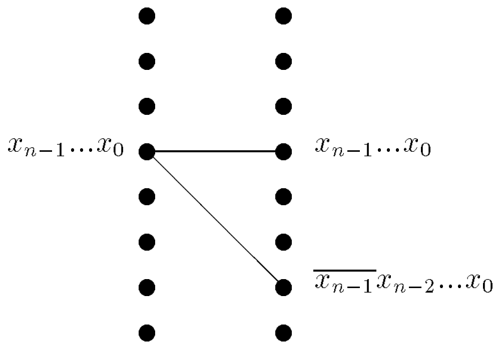

Figure 4.1: Rule for linking layer $n - 1$ and n.

We have the following relation between the number of vertices of two butterfly of consecutive dimensions: $|B(n)| = 2 \times |B(n-1)| + 2^n$. Since $B(0)$ is a single vertex, we have $|B(n)| = (n + 1) \times 2^n$. The degree of each node is 4, except for the nodes of the two extreme layers ($l = 0$ or $l = n$) which have degree 2. The diameter of $B(n)$ is n. Figure 4.2 represents the butterfly graph $B(3)$.

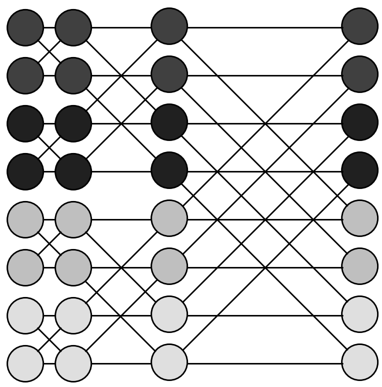

Figure 4.2: Butterfly graph $B(3)$.

4.1.2 Aim

The butterfly graph is often used for parallel computing. Our goal is to otcs-embed the butterfly graphs in a properly sized 8-mesh with edge-congestion, vertex congestion and load equal to one.

We can calculate the expansion of butterfly embedding given by the existence theorem considering this graph as a digraph, where edges are oriented from layers with the smaller index to layers with the larger one. Hence, $B(n)$ has $n \times 2^n$ vertices with input degree 2 (that is vertices of the n layers from 1 to n), and $n \times 2^n$ vertices with output degree 2 (that is vertices of the n layers from 0 to $n-1$). Thus, $M = \sum_{v \in V}(\delta^-(v)+1) = \sum_{v \in V}(\delta^+(v)+1) = 3n \times 2^n$, and according to existence theorem, it is possible to otcs-embed the butterfly graph of dimension n in a $(3n \times 2^n) \times (3n \times 2^n)$ 8-mesh.

We try in this section to decrease the expansion of butterfly embedding and to find a simpler embedding which will be easier to build.

4.2 Butterfly embedding theorem

Theorem 2 gives a simple one-to-connected-set embedding of butterfly graphs in a 8-mesh. The expansion of this embedding is better than the one given by the existence theorem, and expansion is better. Proof of the theorem is based on the row-to-row routing. We first demonstrates 2 propositions about this routing.

Let us call *elementary transposition*, noted $\tau_{j,i}$, a routing configuration where $dest(i) = j$ and $dest(j) = i$, with $j > i$, and $dest(k) = k$ for all $k \neq i, j$. We call $j - i$ the *the order* of this elementary transposition. The lower bound on the number of rows to root such a permutation with odd-even routing is $j-i$. If the order $j-i$ is odd, we will call it an *odd elementary transposition.*

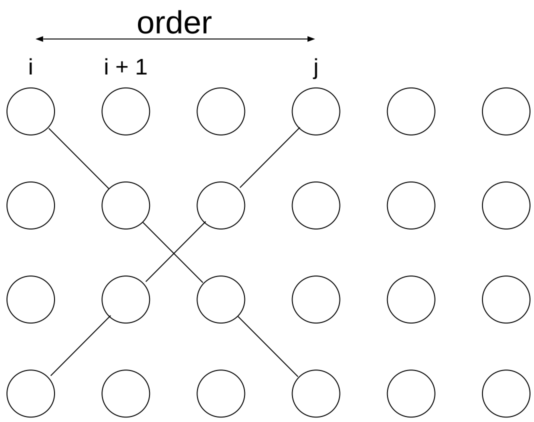

Figure 4.3: Elementary transposition.

Proposition 1 *The routing of a permutation composed of a unique odd elementary transposition can be performed in lower bound with the odd-even algorithm.*

Proof. The proof is given by induction on the order of the transposition. The proposition is trivially true for $\tau_{i+1,i}$. Assuming there exists an integer k such that proposition is true for any $\tau_{i+j,i} : 1 \leq j < k$, j odd and $i, j, k \in \mathbb{N}$, we can show that it is still true for $\tau_{i+k+2,i}$. To study this permutation, it is sufficient to consider $i = 0$; we can as well restrict the considered columns to the $k+3$ first ones. The permutation is then $\tau_{k+2,0}$: $k+2, 1, 2, \cdots, k, k+1, 0$. It leads to a collision free swapping step with odd-even algorithm, and, after this step, we obtain the following row: $1, k+2, 2, 3, \cdots, k, 0, k+1$. We consider now the central part of this row $k+2, 2, 3, \cdots, k, 0$. It happens to be equivalent to transposition $\tau_{k,0}$, by means of a trivial renumbering $k+2 \to k$ and $x \to x-1, \forall x : 0 < x < k+2$ (which does not change the relative order of the values, and hence the routing paths).

The recurrence assumption states that odd-even routing routes this $\tau_{k,0}$ permutation in k steps. We then obtain after k steps the configuration $0, 2, 3, \cdots, k-1, k, k+2$. So, odd-even routing transforms the initial row to the row $1, 0, 2, 3, \cdots, k-1, k, k+2, k+1$ after $1+k$ steps. This last row can now be routed in one step, since it is collision free. Thus, odd-even routing routes any odd elementary transposition $\tau_{i+k,i}$, k odd, in k steps, which is the lower bound of this permutation. ∎

Considering a row with two odd elementary transpositions $\tau_{l,i}$ and $\tau_{m,j}$ with $i < l < m$, $i < j < m$, we call *interval* the difference $l-j$. If the interval is negative, the elementary transpositions are independent. But if the interval is positive, the routing paths of the two transpositions are interleaved, and for some configurations the interaction of the routing paths can create some collisions that will lead to a routing which will need more steps than the lower bound, even if the elementary transpositions are odd. The following proposition gives a sufficient condition to reach the lower bound when several odd elementary transpositions are interleaved.

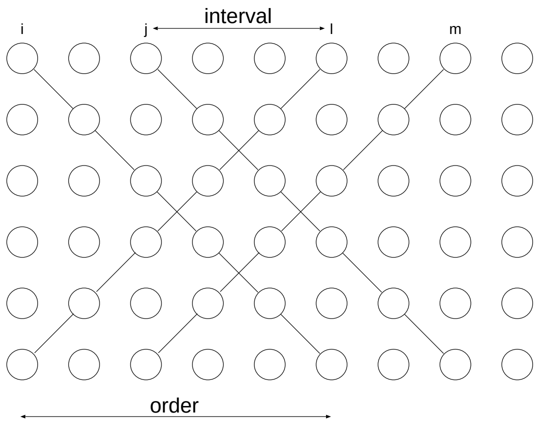

Figure 4.4: *Interval* definition.

Proposition 2 *A permutation composed of several odd elementary transpositions with the same odd order, and such as all the positive intervals are odd, is routed in lower bound with the odd-even routing.*

Proof. The proof is given by recurrence on the intervals. First, let us consider such a row with all the intervals smaller than or equal to 1 (see figure 4.5). We call k the (common) order of the elementary transpositions. We have $k \geq 3$, since the order is supposed to be odd and is necessarily larger than the largest interval. There are no collision in this configuration for the following reason. Assume that there is a collision between messages i and j, with $i > j$ such that i belongs to transposition $\tau_{i,l}$, and j to $\tau_{j,m}$. We have $i - j = i - l - (j - l)$. $i - l$ is an odd order by hypothesis, $j - l$ is an odd interval (equal to one in that case) for the same reason, thus $i - j$ is necessarily even, and the corresponding messages cannot be neighbors.

After one step of the odd-even algorithm, all the intervals become negative, since they were smaller or equal to 1. So we obtain several independent odd elementary transposition of odd order $k - 2$. By means of a renumbering, the previous proposition states that these elementary transpositions needs $k-2$ steps to be routed. We then obtain a row that can be routed in one step. Thus, the

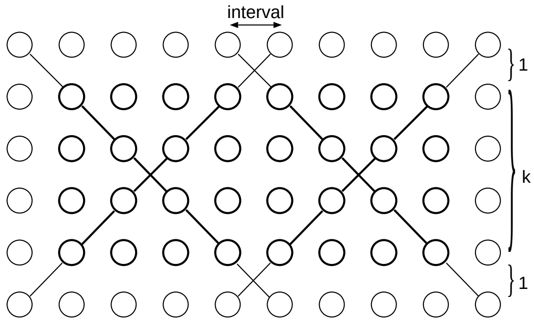

Figure 4.5: Initial step of the demonstration of proposition 2.

initial row needs $1 + (k - 2) + 1$ steps to be routed by the odd-even routing, ie. the row is routed in lower bound, as presented in figure 4.5.

Now, we suppose that every row composed of several odd elementary transpositions of the same order and whose positive intervals are odd and smaller than or equal to q can be routed in a number of steps equal to the lower bound. We will show that it is still true if the intervals are equal to $q + 2$. Let us consider such a configuration, with intervals smaller or equal to $q + 2$. It has no collision for the same reason as above. Thus, after one step, we can renumber the row to obtain a similar one but whose intervals are smaller than or equal to q. Thanks to the recurrence hypothesis, we know that it needs q steps to be routed. We then apply the reverse renumbering to achieve the routing with a last step. The row has then been routed in $1 + q + 1$ steps with the odd-even algorithm, which is the lower bound to realize this permutation. ■

Using the two previous lemma, we can now demonstrate the following theorem.

Theorem 2 *The butterfly graph of dimension n, $B(n)$, can be one-to-connected-set embedded in the 8-mesh of side $2^{n+1} \times (2^{n+1} - n)$, with edge and vertex congestion and minimal expansion.*

Proof. Let the different layers of the butterfly graph be mapped on parallel columns in a 8-mesh and assume that, in each column, a vertex of $B(n)$ is mapped on a set of two successive interconnected vertices of the 8-mesh. That gives $n + 1$ parallel columns of size 2^{n+1}. We assume that each of the two output links of a butterfly vertex leaves one of these nodes, as showed in figure 4.6. This way, edges between two successive layers respect the assumptions we made concerning row-to-row routing as just one link starts from and arrives to every vertex of the columns. Thus, each pair of successive layers of the embedding can be considered as row-to-row routing. Moreover, these routing are composed of several odd elementary transpositions, with odd intervals between them. Indeed, nodes $(l, x_{n-1} \cdots x_0)$ and $(l+1, x_{n-1} \cdots \bar{x} \cdots x_0)$, spaced out by 2^l logical vertices, ie. by $2^{l+1}-1$ physical vertices since each logical vertex is mapped on 2 physical vertices, form an odd elementary transposition (see figure 4.6).

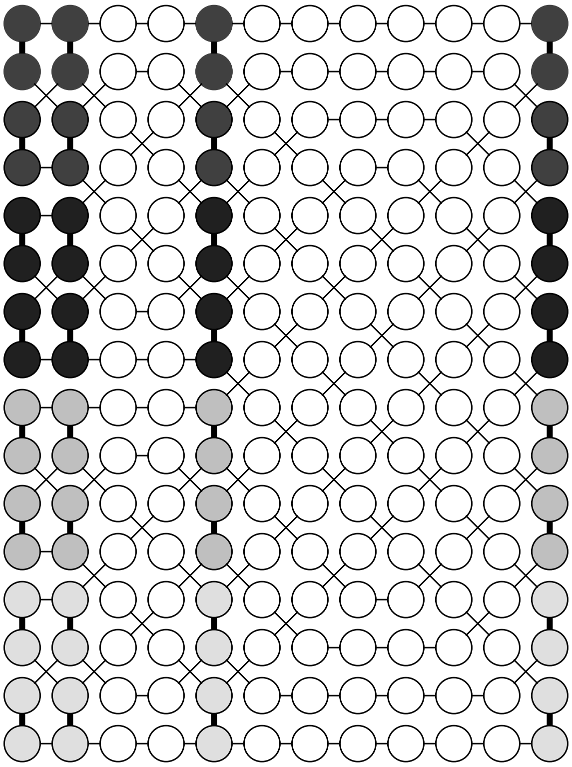

Figure 4.6: Embedding of the butterfly graph ($B(3)$ here).

So, we can use proposition 2 to state that for each routing corresponding to each layer, the minimal bound is reached. The number of lines of the 8-mesh needed for each layer can then be calculated: layer l needs a number of lines equal to the minimal bound, that is 2^{l+1}. The total number of lines for all the layers is then equal to $\sum_{l=0}^{n-1}\left(2^{l+1}-1\right)+1=2^{n+1}-n$ (columns shared between two layers must not be counted twice, but the last column must not be forgotten). That gives the result. ■

The expansion of this embedding is equal to
$\frac{2^{n+1}\times(2^{n+1}-n)}{(n+1)2^n} \approx 2^n$, that largely improves the existence theorem bound.

Chapter 5

Hypercube embedding

In this section, we show that a better embedding than the one given by the existence theorem can be found for the hypercube graph. This embedding is simpler and has a better expansion. It is based on a more general bipartite graphs embedding technique. First, we present this technique, called *bundles technique*, and then its application to embed the hypercube graph.

5.1 Bundles technique

Let us call *bundle* a set of parallel diagonal paths separated each other by empty diagonals. There are 2 bundles in a 8-mesh. More formally, in a $n \times n$ 8-mesh, the first bundle is the set of paths defined as follows: $(i,0) \leftrightarrow (i+1,1) \leftrightarrow \cdots \leftrightarrow (n-1,n-1-i)$, or $(0,i) \leftrightarrow (1,i+1) \leftrightarrow \cdots \leftrightarrow (n-1-i,n-1)$, for all $0 \leq i \leq n-1$ and i even. The second bundle is the set of paths defined as follow: $(i,0) \leftrightarrow (i-1,1) \leftrightarrow \cdots \leftrightarrow (0,i)$, and $(n-1,i) \leftrightarrow (n-2,i+1) \leftrightarrow \cdots \leftrightarrow (i,n-1)$ for all $1 \leq i \leq n-1$ and i odd.

The first and second bundles can cross themselves without having any common vertex (see figure 5.1). Moreover, every diagonal of the first bundle can be linked to any diagonal of the second bundle with just one edge. Such a link can always be done, even if several other links already exist between the two bundles, assuming diagonals are long enough.

We can use two bundles like in figure 5.1 to embed a bipartite graph. Indeed, we map the first partition on the first bundle and the second partition on the second bundle. Every vertex of the logical graph is mapped to an entire diagonal of the corresponding bundle. That gives an otcs-embedding, with unit load, edge congestion, vertex congestion and dilation.

This process may be interesting for peculiar bipartite graphs having a recursive construction and a degree growing with the recursion. Indeed, adding a new vertex to the embedded graph is always feasible and consists to add a new diagonal to one of the bundles. This new vertex will always be able to be linked to

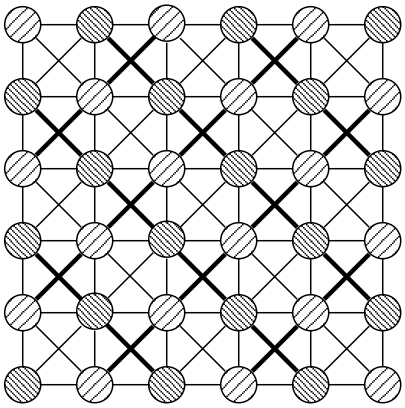

Figure 5.1: Two crossing bundles.

all the vertices of the other partition because a diagonal of the first bundle can always be linked with the diagonals of the other bundle. Increasing the degree of the vertices can be done by enlarging the size of the diagonals to establish new links. All these modifications can simply be done by enlarging the 8-mesh (and thus the diagonal size).

The following proposition summarizes the preceding results and states precisely the expansion of the embedding for a given bipartite graph.

Proposition 3 *A bipartite graph, with p vertices in the first partition and q in the second, can be one-to-connected-set embedded with unit load, edge congestion, vertex congestion and dilation, in the* $(2 \times max(p,q) - 2) \times (2 \times max(p,q) - 2)$ *8-mesh.*

Proof. Let us consider the complete bipartite graph, with n vertices in each partition. We can show that such a graph can be embedded in a $(2n-2)\times(2n-2)$ 8-mesh, by a recurrence on $2n$, the number of vertices of the graph.

Indeed, a 4 vertices bipartite graph, with 2 vertices in each partition, needs a 2×2 8-mesh. Adding one vertex to each partition of an embedded complete bipartite graph, with n vertices in each partition, implies to add one diagonal in each bundle, that increases by 2 the side of the 8-mesh (see figure 5.2). Each of these new diagonals can be linked to all diagonals of the other bundle, and we then obtain a complete bipartite graph with one additional vertex in each partition, that is a bipartite complete graph with $n+1$ vertices in each partition. Thus, by recurrence, the side of the 8-mesh is equal to $2n - 2$.

To embed a bipartite graph with p vertices in the first partition and q in the second, it is sufficient to embed the complete bipartite graph having $max(p,q)$ vertices in each partition. That needs a $(2max(p,q) - 2) \times (2max(p,q) - 2)$ 8-mesh. ■

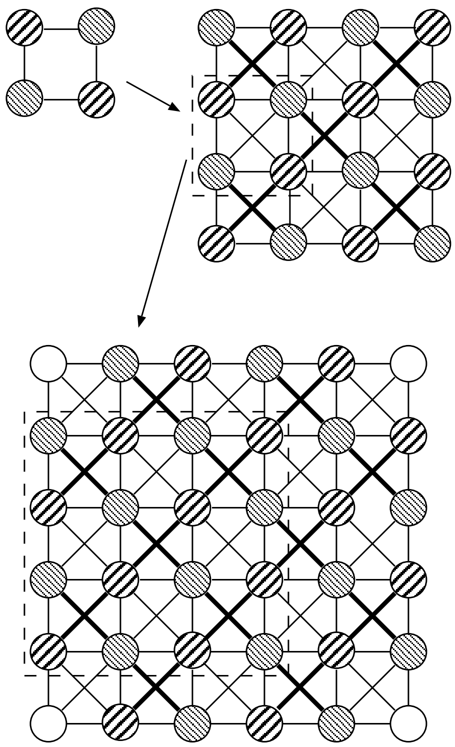

Figure 5.2: First steps of bundles construction.

5.2 Application to hypercube graph

5.2.1 Definition

The hypercube $H(n)$ is a graph having 2^n vertices numbered, in binary, from $0_{n-1}...0_0$ to $1_{n-1}...\ 1_0$. Two vertices are linked if and only if they have one and only one different digit in their binary representation.

The diameter of $H(n)$ is equal to n. Its number of vertices is 2^n. It is a regular graph: all the vertices have the same degree, which is equal to n. Also, $H(n)$ is bipartite graph: the first partition is composed of vertices having an odd number of 1 in their binary representation, while the second is composed of those having an even number of 1. This property allows us to use the bundles technique for the hypercube.

5.2.2 Theorem

The existence theorem indicates that it is possible to embed $H(n)$ in the $\lceil \frac{n}{2} \rceil (2^n + 1) \times \lceil \frac{n}{2} \rceil (2^n + 1)$ 8-mesh (cf. section 3.2). The following theorem gives a best expansion for this embedding. It is based on the bundles technique.

Theorem 3 *The hypercube graph $H(n)$, of dimension n, can be one-to-connected-set-embedded in the $(2^n - 2) \times (2^n - 2)$ 8-mesh, with unit load, edge congestion, vertex congestion and dilation.*

Proof. The Hypercube graph is bipartite: $H(n)$ is composed of 2 partitions of 2^{n-1} vertices each. Then we can use proposition 3. That gives the result. ∎

This embedding is simpler to build and use than those of the existence theorem. It is illustrated in figure 5.3. The double arrows in this figure indicate the choice of logical edges we have done for this embedding. Other choices are of course possible.

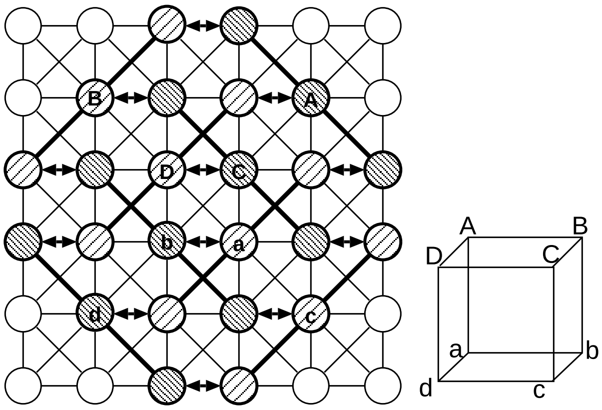

Figure 5.3: Hypercube embedding (here $H(3)$).

Chapter 6

Binary trees embedding

6.1 Overview and goal

The *complete binary* tree T_n, of height n, is defined recursively as follows. T_0 is a vertex called the root of T_0. For $n \geq 1$, T_n is obtained from two copies of T_{n-1} by joining their roots to a new vertex, called the root of T_n. We call *level* k the set of vertices at distance k of the root. The number of vertices of T_n is $2^{n+1} - 1$.

Our aim is to embed complete binary trees in a square 8-mesh with unit load, edge congestion, vertex congestion and minimal expansion. These hypothesis are needed for efficiency in the AMM. Here, the idea is to find a function $f : \mathbb{N} \to \mathbb{N}$ so that T_n can be otcs-embedded with AMM restrictions (cf. 2.4) in a $f(n) \times f(n)$ 8-mesh.

Theorem 1 states that it is possible to embed the binary tree $T(n)$ in a 8-mesh of side $\sim 2^{n+2}$.

Tree T_n is a bipartite graph. The parts consist of levels of same parity. If the height n is odd, by considering pairs of adjacent levels, it is possible to show that the part with even levels will have exactly $\frac{1}{3}$ of the total number of nodes, and the odd one the remaining $\frac{2}{3}$. With this partition, thanks to proposition 3, as the maximum cardinal of the parts is $\lfloor \frac{2}{3} \times (2^{n+1} - 1) \rfloor$, we know it is possible to embed T_n in a $(\lfloor \frac{2}{3} \times (2^{n+1} - 1) \rfloor - 2)$ side 8-mesh, using the bundles technique. A similar estimation can be done for trees of even height.

This result is better that the one given by the existence theorem, but we are interested by an embedding having a simpler construction and a better expansion.

6.2 Tree embedding method

In this section, most of the results stands for either 4- or 8-meshes. Thus we consider a mesh, whose degree is only specified when necessary. We present here a technique to embed binary trees; its consists in paving with small patterns, called tiles. First, tiles are squares. Second, we explain a new technique based on rectangle tiles.

6.2.1 Paving with square tiles

The paving technique consists to cluster simple tiles to get a tree. Indeed, a tree is composed of several smaller trees and a mesh is formed of several smaller meshes. Thus, if a small tree can be embedded in a small mesh, a larger tree should be embedded in a larger mesh, composed of small meshes aggregated, as shown in figure 6.1 (cf. [15]). Of course, some precautions must be taken.

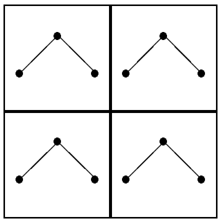

Figure 6.1: Principle of embedding with square paving.

First, the basic tile must be found. It must be unit load, edge and vertex congestion otcs-embedding of a small tree in a small mesh. For a given tree, the smaller the size of tile, the better the expansion of the embedding.

Second, a clever interconnection scheme must be found. The problem is to create a new root and to link it to the roots of two simple tiles. For this purpose, the root of the tile needs to be reachable from the outside of the tile. And a place must be foreseen for the root of the new tree. It is a bad idea to foresee this place in the tile itself, as the recursive construction of the embedding will stop when the tiles will be full. Furthermore, the new root obtained by linking two tiles will require a path to be reached from the outside of the two tiles in order to embed a larger tree, and lead to sparse tiles, if this path belongs to the initial tiles. To conclude, the root of the tile must be reachable from the outside of the tile, and a path (and a place) for new roots must be foreseen *between* the tiles (see figure 6.2).

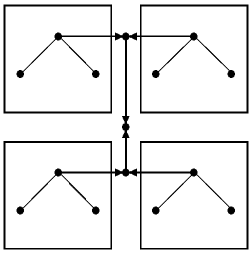

Figure 6.2: Embedding with square paving.

Let the tile be T_p, embedded in the $m \times m$ square mesh. If we aggregate 4 tiles with a path between them, we can embed T_{p+2} in a $(2m+1) \times (2m+1)$ mesh. By applying this process recursively, we will obtain the embedding of any tree T_{p+2k}, $k \in \mathbb{N}$ in a $(2^k \times m + 2^k - 1) \times (2^k \times m + 2^k - 1)$ mesh (see theorem 4 for details). Here, we see that we need 2 tiles to build all the trees: one for odd trees (T_{2h+1}, $h \in \mathbb{N}$) and one for even trees (T_{2h}, $h \in \mathbb{N}$). Function f can now be defined. The following theorem states precisely its definition for any mesh (4-mesh or 8-mesh).

Theorem 4 (Square paving) *Let the odd tile be T_p, embedded in the $m \times m$ mesh (p is odd) and the even tile be $T_{p'}$, embedded in the $m' \times m'$ mesh (p' is even). Function f, which gives the mesh size for the embedding of a tree with square paving, is defined as follows:*

$$f(n) = \begin{cases} 2^{\frac{n-p}{2}} m + 2^{\frac{n-p}{2}} - 1 \\ \qquad \textit{if } n \textit{ is odd and } n \geq p \\ 2^{\frac{n-p'}{2}} m' + 2^{\frac{n-p'}{2}} - 1 \\ \qquad \textit{if } n \textit{ is even and } n \geq p' \end{cases}$$

Proof. We prove this result by recurrence. First, it is true if $n = p$ or if $n = p'$. Second, assume we have tree T_n embedded in a square mesh of side $2^{\frac{n-p}{2}} m + 2^{\frac{n-p}{2}} - 1$. Then, T_{n+2} can be embedded in a mesh of side $2 \times (2^{\frac{n-p}{2}} m + 2^{\frac{n-p}{2}} - 1) + 1$. That gives the result. ∎

We can note that trees of a smaller order than the building block are not given by the recursive construction. They are peculiar cases and must be obtained by finding smaller specific tiles.

Moreover, the paths between the tiles are not full. So, the expansion of the embedding can always be improved, from a minimal number of aggregated tiles, if a longer tile, with a better expansion than the aggregate of several small tiles, is used. But the recursive construction of the embedding will be more complex because the number of particular cases will increase. In subsection 6.3 and 6.4, the simplicity of the embedding is preferred to a better expansion.

6.2.2 Paving with rectangles

Rectangle tiles are sometimes better than square tiles. Indeed, if the rectangle tile perimeter is smaller than the square tile perimeter, the expansion of the embedding is better when rectangles are used instead squares. The recursive construction consists to aggregate 4 rectangles aligned in the same direction to obtain the next rectangle as it is showed below (see figure 6.3).

The idea is similar to the square paving technique, except for the last step. Indeed, in both square and rectangle paving, we want to obtain an embedding of a binary tree in a square mesh. With square paving, each step gives an embedding

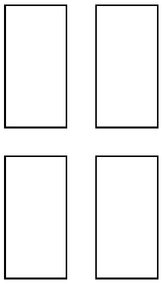

Figure 6.3: Rectangle obtained with 4 small rectangles.

in a square mesh. But with rectangles, the last step must be different from the others to obtain a square mesh instead a rectangular one.

The last step of the recurrence consists to aggregate 4 rectangles to obtain a square, like in figure 6.4.

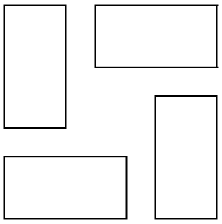

Figure 6.4: Square obtained with 4 small rectangles.

Center condition

It is necessary to verify when the aggregation of figure 6.4 is possible. This configuration must ensure a large enough room to place 3 new roots and to link them together.

There is a central square between the 4 rectangles of figure 6.4. This central square must be large enough to contain 3 roots and theirs links. If the smaller side of each rectangle is called l and the larger one is called L, the side of the central square is then equal to $L - l + 1$ (see figure 6.5).

$L - l$ is always larger than 0 else we do not have a rectangle paving but a square paving. Thus, the minimal size of the square paving is 2. As shown in figure 6.6, the minimal size of a square capable of containing 3 roots and theirs links is 3 in a 4-mesh, and 2 in a 8-mesh.

To conclude, the recursive construction of the embedding using rectangle paving must begin with a rectangle having $L - l \geq 2$ in a 4-mesh. But this construction can begin with any rectangle ($L - l \geq 1$) in a 8-mesh. This restriction will be called *center condition* in the following. The center condition is always respected in a 8-mesh. In a 4-mesh, it is true if and only if $L - l \geq 2$.

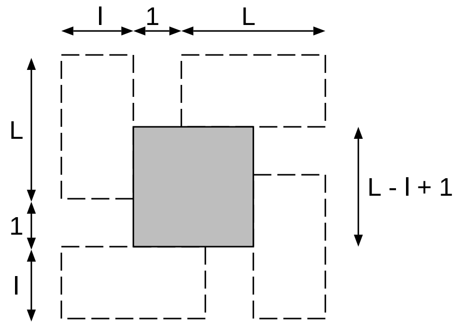

Figure 6.5: The side of the central square is equal to $L - l + 1$.

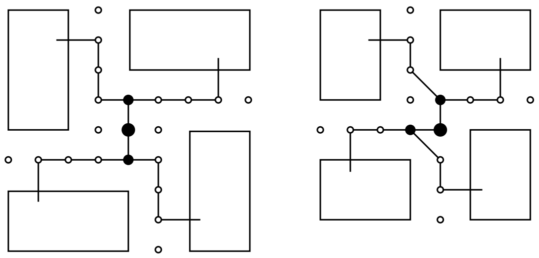

Figure 6.6: Minimal central square in a 4-mesh and in a 8-mesh.

Note that square paving does not need center condition because it is possible to display the tiles so that the output of the paths from the roots are facing each other. Thus, two roots can be linked without going through the central square (see figure 6.2).

The following theorem states precisely the definition of the function f, which gives, for a particular tree, the minimal size of the mesh able to contain the embedding of the tree. Two lemmas help to demonstrate the theorem.

Lemma 1 *If tree T_p is embedded in an $m_1 \times m_2$ mesh, then tree T_{p+2k}, $(k \in \mathbb{N})$, can be embedded in a $(2^k m_1 + 2^k - 1) \times (2^k m_2 + 2^k - 1)$ mesh.*

Proof. The proof is similar to that of theorem 4. We know that T_{p+2k} can be embedded in the $(2^{\frac{(p+2k)-p}{2}} m_1 + 2^{\frac{(p+2k)-p}{2}} - 1) \times (2^{\frac{(p+2k)-p}{2}} m_2 + 2^{\frac{(p+2k)-p}{2}} - 1)$ mesh. That gives the result. ∎

Lemma 2 *If tree T_p is embedded in an $m_1 \times m_2$ mesh, then tree T_{p+2} can be embedded in a $(m_1 + m_2 + 1) \times (m_1 + m_2 + 1)$ mesh, if the "center condition" is respected.*

Proof. The side of the square is equal to the larger side of the rectangle, plus the smaller side of the rectangle plus 1 (the path between the rectangles to place the 3 new roots, and to link them together). The square mesh size is then $m_1 + m_2 + 1$. ∎

Theorem 5 (rectangle paving) *Let the odd tile be T_p embedded in the $m_1 \times m_2$ rectangle mesh (p is odd) and the even tile be $T_{p'}$ embedded in the $m'_1 \times m'_2$ rectangle mesh (p' is even). If the* center condition *is true, the function f, which gives the size of the final square mesh to embed a tree with rectangle paving, is defined as follows:*

$$f(n) = \begin{cases} 2^{\frac{n-2-p}{2}}(m_1 + m_2) + 2^{\frac{n-p}{2}} - 1 \\ \quad \text{if } n \text{ is odd and } n \geq p \\ 2^{\frac{n-2-p'}{2}}(m'_1 + m'_2) + 2^{\frac{n-p'}{2}} - 1 \\ \quad \text{if } n \text{ is even and } n \geq p' \end{cases}$$

Proof. Here, the proof is done for n odd. T_{n-2} can be embedded in the $(2^{\frac{n-2-p}{2}} m_1 + 2^{\frac{n-2-p}{2}} - 1) \times (2^{\frac{n-2-p}{2}} m_2 + 2^{\frac{n-2-p}{2}} - 1)$ rectangle mesh, according to lemma 1. If we aggregate 4 of these rectangles, like in figure 6.4, we obtain T_n in a square mesh of size $(2^{\frac{n-2-p}{2}} m_1 + 2^{\frac{n-2-p}{2}} - 1) + (2^{\frac{n-2-p}{2}} m_2 + 2^{\frac{n-2-p}{2}} - 1) + 1$, according to the lemma 2 (we assume the "center condition" is respected). That gives a $2^{\frac{n-2-p}{2}}(m_1 + m_2) + 2^{\frac{n-p}{2}} - 1$ square mesh. Proof for n even is similar. ∎

As we can remark, two starting tiles are still needed here, one for the odd trees (T_{2h+1}, $h \in \mathbb{N}$), and one for even trees (T_{2h}, $h \in \mathbb{N}$). Moreover, the center condition must be respected. But, as we showed above, it is not so restricting. The interest of paving with rectangles instead of squares can now be explained. The following theorem states precisely this result.

Theorem 6 (Interest of rectangle paving) *If a tree T_n can be embedded in a $f(n) \times f(n)$ mesh with a square tile T_p embedded in a $m \times m$ mesh, and if T_p can be embedded in a $m_1 \times m_2$ rectangle mesh, then tree T_n can be embedded in a $(f(n) - 2^{\frac{n-2-p}{2}}(2m - m_1 - m_2)) \times (f(n) - ^{\frac{n-2-p}{2}}(2m - m_1 - m_2))$ square mesh instead of a $f(n) \times f(n)$ one.*

Proof. According to theorem 4, the function f_s obtained with the square tile T_p embedded in the $m \times m$ square mesh, is defined as follow: $f_s(n) = 2^{\frac{n-p}{2}} m + 2^{\frac{n-p}{2}} - 1$ (if n and p have the same parity and $n \geq p$). According to theorem 5, the function f_r obtained with the rectangle tile "T_p embedded in the $m_1 \times m_2$ rectangle mesh" is defined as follows: $f_r(n) = 2^{\frac{n-2-p}{2}}(m_1 + m_2) + 2^{\frac{n-p}{2}} - 1$ (if n and p have the same parity and $n \geq p$). We then have $f_s(n) - f_r(n) = 2^{\frac{n-p}{2}} m - 2^{\frac{n-2-p}{2}}(m_1 + m_2) = 2^{\frac{n-2-p}{2}}(2m - m_1 - m_2)$. That gives the theorem. ∎

So, as we said in the beginning of the subsection, if a rectangle tile has a smaller perimeter than the square tile (ie. $2m > m_1 + m_2$), and contains the same tree, then paving with rectangle is better in terms of expansion.

6.3 On the 4-mesh

In this subsection, we apply the technique of paving to obtain binary trees embedding on the 4-mesh. First, we use square paving. Then, we use rectangle paving.

For both, two tiles are needed: one for odd trees and one for even trees. This part allows to compare square and rectangle paving in situ and also to study interest of diagonals in a mesh.

6.3.1 Using square paving

Figure 6.7 shows how we can embed T_1 on the 2×2 square 4-mesh. One the same figure, the first step of the recurrence is done: T_3 can be embedded in the 5×5 square 4-mesh.

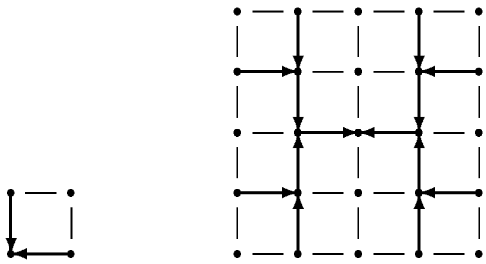

Figure 6.7: Odd trees embedding on the 4-mesh.

The tile for even trees is shown on figure 6.8. On this figure, the first step of the recursive construction is also displayed. This construction is called *H-tree* in [15]. As the embedding obtained seems to be the letter H. They are used in VLSI circuits to distribute clock line on integrated circuits, with equal lengths on the different branches.

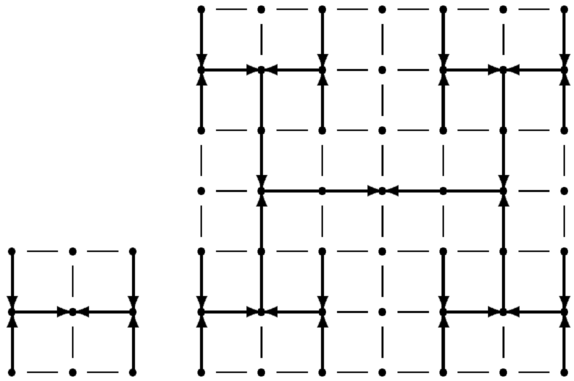

Figure 6.8: Even trees embedding on the 4-mesh.

These four tiles give the following function f — remember $f(n)$ gives the side of the 4-mesh that allow to embed T_n —, according to theorem 4:

$$f(n) = \begin{cases} 2^{\frac{n+1}{2}} + 2^{\frac{n-1}{2}} - 1 \\ \qquad \text{if } n \text{ is odd and } n \geq 1 \\ 3 \times 2^{\frac{n}{2}-1} + 2^{\frac{n}{2}-1} - 1 \\ \qquad \text{if } n \text{ is even and } n \geq 2 \end{cases}$$

6.3.2 Using rectangle paving

It is not possible to improve the expansion of T_1 in a square 4-mesh. And there does not exist a rectangular 4-mesh able to contain T_1 and with a perimeter smaller than 4 (which is the perimeter of the 2×2 4-mesh). Moreover, it is not possible to improve the expansion of the embedding of T_3 in a square 4-mesh. However, T_3 can be embedded in the 4×5 4-mesh, as shown in figure 6.9. Although the center condition is not respected here, the construction can begin with the tile T_3 embedded in the 4×5 4-mesh, because the corner vertex near the root is free. Thus, there is no problem to place the new roots between the 4 rectangles and to link them to the 4 subtrees. The recursive construction of embedding based on rectangle paving can then be used starting from T_3.

Similarly, the embedding of T_2 cannot be improved neither in a square 4-mesh, nor in a rectangle 4-mesh having a smaller perimeter. And, it is not possible to improve the embedding of T_4 in a square 4-mesh. But T_4 can be embedded in a 6×7 rectangle 4-mesh (see figure 6.9). However, the center condition is not respected for T_4, because the difference between the sizes of the two sides of the rectangle is not large enough. Thus, we must use the following tile, which is T_6 embedded in the 13×15 rectangle 4-mesh. The recursive construction of embedding based on rectangle paving should then start from T_6.

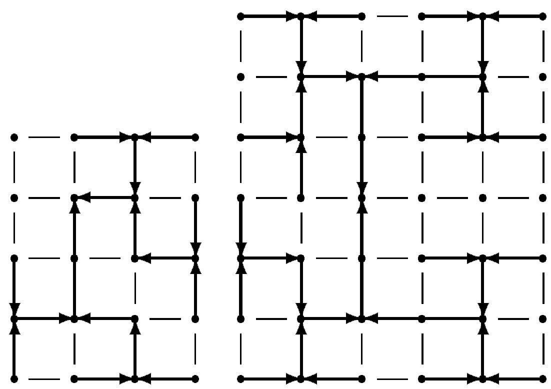

Figure 6.9: Rectangle tiles on the 4-mesh.

The resulting f function, which characterizes the embedding, is defined as follows, according to theorem 5:

$$f(n) = \begin{cases} 9 \times 2^{\frac{n-1}{2}-2} + 2^{\frac{n-1}{2}-1} - 1 \\ \qquad \text{if } n \text{ is odd and } n \geq 3 \\ 7 \times 2^{\frac{n}{2}-2} + 2^{\frac{n}{2}-3} - 1 \\ \qquad \text{if } n \text{ is even and } n \geq 6 \end{cases}$$

6.4 On the 8-mesh

6.4.1 Using square paving

To profit from the diagonals of the 8-mesh, the odd tile should be the embedding of T_3 at least. Indeed, no improvement can be found with T_1. Similarly, the even tile should at least be the embedding of T_4, because T_2 needs a 3×3 square 4-mesh, and also a 3×3 8-mesh.

Figure 6.10 shows the embedding of T_3 in the 4×4 square 8-mesh, and the embedding of T_4 in the 6×6 8-mesh.

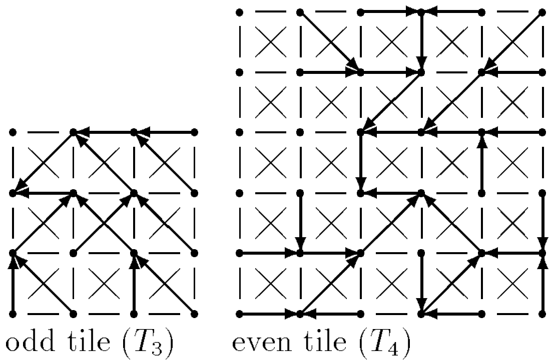

Figure 6.10: Square tiles on the 8-mesh.

According to theorem 4, the previous tiles allow us to define the f function to characterize the embedding of trees in the 8-mesh as follows :

$$f(n) = \begin{cases} 2^{2+\frac{n-3}{2}} + 2^{\frac{n-3}{2}} - 1 \\ \qquad \text{if } n \text{ is odd and } n \geq 3 \\ 3 \times 2^{\frac{n}{2}-1} + 2^{\frac{n}{2}-2} - 1 \\ \qquad \text{if } n \text{ is even and } n \geq 4 \end{cases}$$

6.4.2 Using rectangle paving

It is easy to see that the square tiles found for the 8-mesh (see figure 6.10) cannot be improved, even if the tile becomes a rectangle. Indeed, in both cases, the number of free vertices in the 8-mesh (which is equal to 1 for the odd tile and 5 for the even tile) is inferior to the smaller 8-mesh side.

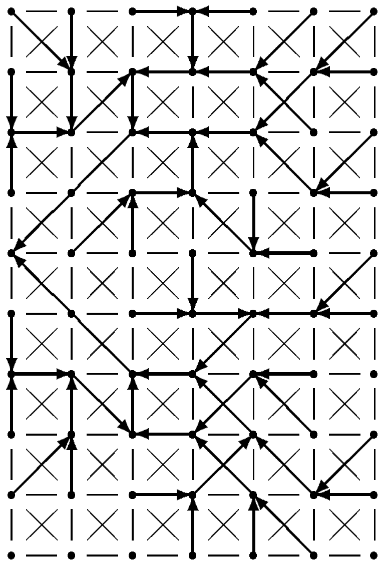

Figure 6.11: A tile with T(5) in a rectangular 8-mesh.

Figures 6.11 and 6.12 give rectangle paving with trees T_5 and T_6.

Tree $T(5)$ has 63 vertices and need at least an 8×8 8-mesh. But if the root is placed on the border of the 8-mesh, the height of the tree is to be compared to the side of the 8-mesh (see 6.5). The height of the tree is 5, which is smaller than the side of the 8-mesh. Thus, there are at least three nodes used for dilation of edges in the 8-mesh, and the 8×8 8-mesh is not large enough. The 9×9 8-mesh has a perimeter equal to 36. But the perimeter of the rectangular tile in figure 6.11 is equal to 34. Thus, it is better to use this tile and rectangle paving instead of square paving with a square tile built with $T(5)$.

The following subsection details the limitations due to diameter comparison. The resulting f function defined with these two tiles is given below:

$$f(n) = \begin{cases} 17 \times 2^{\frac{n-1}{2}-3} + 2^{\frac{n-1}{2}-2} - 1 \\ \qquad \text{if } n \text{ is odd and } n \geq 5 \\ 25 \times 2^{\frac{n}{2}-4} + 2^{\frac{n}{2}-3} - 1 \\ \qquad \text{if } n \text{ is even and } n \geq 6 \end{cases}$$

6.5 About dilation

In this section we discuss about dilation in the tiles. Dilation between the tiles is necessary to link roots (see 6.2.1). But dilation is also necessary if the tiles are larger than a certain size. Ullman studied this problem in [15], and showed for VLSI circuits that dilation of an embedding of tree T_n is greater or equal to $\sqrt{\frac{2^{n+1}-1}{\pi n^2}}$. The geometric reasoning that he used is not strictly valid in our case, and we will establish results for 4- and 8-meshes.

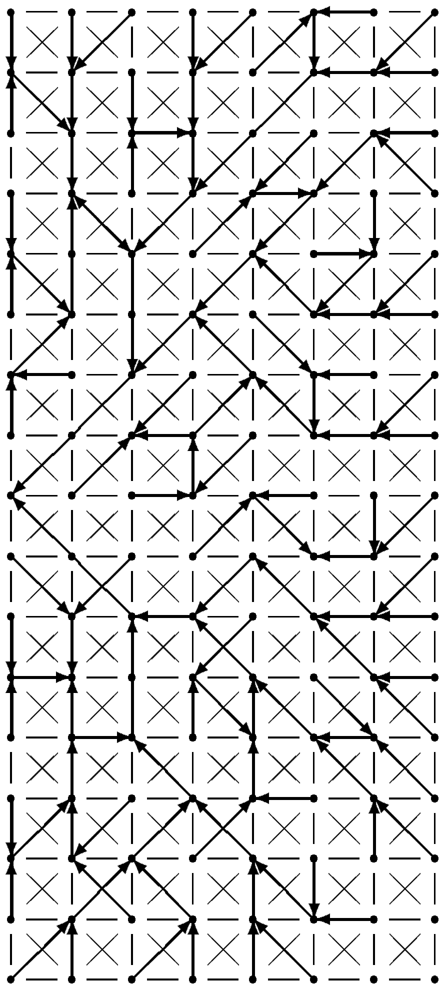

Figure 6.12: A tile with T(6) in a rectangular 8-mesh.

Indeed, diameter of the $p \times q$ 8-mesh is $max(p,q)-1$, and this graph has $p \times q$ vertices. Tree $T(n)$ has $2^{n+1}-1$ vertices but a diameter of $2n$. The minimal square 8-mesh in which it can be embedded, has a side of size $\lceil\sqrt{2^{n+1}-1}\rceil$. This graph has a diameter equal to $\lceil\sqrt{2^{n+1}-1}\rceil-1$, which is larger than $2n$ for $n > 6$. Thus, dilation must be larger than 1 if $n \geq 7$, and grows with n afterwards.

Moreover, if the root of the tree is not exactly in the center of the tile, dilation is required for trees smaller than T_7. For instance, if the root is on a side of the 8-mesh, dilation will take place as soon as the lateral dimension of the tile is larger than the height of the tree, that is for T_5.

If we use a 4-mesh, we should compare the diameter of the minimal square 4-mesh containing T_n, which is $2\lceil\sqrt{2^{n+1}-1}\rceil-2$, to $2n$. With a similar calculus, we found that dilation larger than 1 should be used from tree $T(4)$. But this result is just an approximation in the 4-mesh because all the vertices of the sides of the 4-mesh are not at the same distance from the center of the 4-mesh (thus, there may exist some embeddings which need large 4-meshes but don't use vertices near corners of the 4-meshes). However, the tile found with $T(4)$ in the 4-mesh use dilation greater than 1 (see figures 6.8 and 6.9).

Chapter 7

Conclusion

We have presented in this paper the use of a new computing model, the associative mesh, to embed arbitrary graphs in a 2D 8-mesh. The global communication facilities of the model allow to use a set of connected nodes to implement either virtual nodes with an arbitrarily large degree or long interconnection arcs, giving a flexible embedding scheme. While any graph can be embedded, as stated by the existence theorem, provided the size of the 8-mesh is large enough, the expansion is in general too large for a practical use. Anyway, we have shown that some specific graphs — like the hypercube, butterfly, trees — give a largely improved expansion with ad-hoc techniques. In either case, assumptions of the model state that any communication along an embedded edge can be done in unit time.

To conclude, we would like to point out that our main motivations are not theoretical. Present VLSI technology allows a very large number of nodes to be integrated within a chip, and to communicate very rapidly along any connected set of nodes. The proposed scheme can be an efficient alternative to the use of traditional routing mechanisms in parallel computers, improving simultaneously cost-effectiveness and performances.

Acknowledgments

A special thanks to Dominique Sotteau for her careful rereading of this paper and for the numerous discussions that we had.

Bibliography

[1] H. J. Siegel. *Interconnection networks for large-scale parallel processing.* Lexington books, 1985.

[2] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures : Arrays, Trees, Hypercubes.* Morgan-Kaufmann, San Mateo, CA, USA, 1992.

[3] Thinking Machines Corporation, Cambridge, Massachusetts. *Connection Machine Model CM-2 Technical Summary*, Thinking Machines Corporation edition, November 1990. Version 6.0.

[4] Thinking Machine Corporation. *The CM-5*, 1992.

[5] C. E. Leiserson, Z. S. Abuhamdeh, D.C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. S. St. Pierre, D. S. Wells, M. C. Wong, S-W. Yand, and R. Zack. The network architecture of the Connection Machine CM-5. Technical report, Thinking Machine Corporation, Cambridge, Massachusetts 02142, July 1992.

[6] H. Li and Q. Stout. *Reconfigurable Massively Parallel Computers.* Prenctice Hall, 1991.

[7] M. Maresca and H. Li. Polymorphic VLSI arrays with distributed control. In *Reconfigurable Massively Parallel Computers* [6].

[8] Guy E. Blelloch. *Vector Models for Data-Parallel Computing.* MIT Press, Cambridge, MA, USA, 1990.

[9] S. Lakshmivarahan and Sudarshan K. Dhall. *Parallel Computing Using the Prefix Problem.* Oxford University Press, 1994.

[10] B. Monien and H. Sudborough. Embedding one interconnection network in another. *Computing Suppl.*, pages 7.257–282, 1990.

[11] Stephen T. Hedetniemi Sandra M. Hedetniemi and Arthur L. Liestman. A survey of gossiping and broadcasting in communication networks. *NETWORKS*, 18:319–349, 1988.

[12] D. Dulac, S. Mohammadi, and A. Mérigot. Implementation and evaluation of a parallel architecture using asynchronous communications. In L. Lombardi V. Cantoni and al., editors, *Proceedings of the 1995 Workshop on Computer Architecture for Machine Perception*, pages 106–111, Pavia, Italia, September 1995.

[13] Didier Dulac. *Contribution au Parallélisme Massif en Analyse d'Images : Une Architecture SIMD Fondée sur la Reconfigurabilité et l'Asynchronisme.* PhD thesis, University Paris-Sud, 1996.

[14] Alain Mérigot. Associative nets: A new parallel computing model. *IEEE Trans. on Computers*, to appear.

[15] J. D. Ullman. *Computational Aspects of VLSI.* Computer Science Press, 1984.