AMES GRANT
IN-62-CR
158862
16P.

COMPUTER SYSTEMS LABORATORY

STANFORD UNIVERSITY · STANFORD, CA 94305-4055

PARTITIONING OF REGULAR COMPUTATION ON MULTIPROCESSOR SYSTEMS

FUNG FUNG LEE

Technical Report: CSL-TR-88-352

FEBRUARY 1988

(NASA-CR-183176) PARTITIONING OF REGULAR COMPUTATION ON MULTIPROCESSOR SYSTEMS (Stanford Univ.) 16 p CSCL 09B

N88-28676

Unclas

G3/62 0158862

The work described herein was supported by NASA Ames Research Center under contract NAG2-248, using facilities provided under NASA contract NAGW 419.

Partitioning of Regular Computation on Multiprocessor Systems

by

Fung Fung Lee

Technical Report: CSL-TR-88-352

February 1988

Computer Systems Laboratory

Departments of Electrical Engineering and Computer Science

Stanford University

Stanford, California 94305

Abstract

Problem partitioning of regular computation over two-dimensional meshes on multiprocessor systems is examined. The regular computation model considered involves repetitive evaluation of values at each mesh point with local communication. The computational workload and the communication pattern are the same at each mesh point. The regular computation model arises in numerical solutions of partial differential equations and simulations of cellular automata. Given a communication pattern, a systematic way to generate a family of partitions is presented. The influence of various partitioning schemes on performance is compared on the basis of computation to communication ratio.

Key Words and Phrases: Partition, Partitioning, Parallel Processing, Stencil, Regular Computation.

Copyright © 1988
by
Fung Fung Lee

Contents

1 Introduction **1**

2 Generation of Partitions **2**

3 Properties of Partitions **4**

4 Comparison of Partitions **7**

5 Conclusion **11**

1 Introduction

Applying parallel processing in solving computational intensive problems has been of much interest in recent years. There are many scientific and engineering problems in which the major computation structure is *regular*. This kind of regularity is a great advantage contributing to the good performance of many parallel implementation.

The regular computation model considered involves repetitive evaluation of values at each mesh point with local communication. The computational workload and the communication pattern are the same at each mesh point. This class of computations naturally arises in numerical solutions of partial differential equations and simulations of cellular automata.

The numerical solution of partial differential equations (PDE), by methods such as point Jacobi iteration, involves evaluation of the value at each mesh point at each iteration as the weighted sum of the previous values of its neighbors. The pattern of communicating neighbors is called the *stencil*. For example, if only the values of the north, south, east and west neighbors of a point is needed, the stencil used is called a 5-point stencil. Interestingly but maybe not suprisingly, a new non-PDE approach for solving physical problems also shares the characteristic of regular computation. Recent research in physics has shown that lattice gas cellular automata [1] have the great potential of simulating fluid flow phenomena. A cellular automaton consists of cells possessing discrete values. At each cycle, the value of a cell is evaluated as a function of the values of itself and its neighbors.

When this kind of regular computation is implemented on a multiprocessor system, it is generally preferable to divide the data space (mesh points) into partitions, and assign each partition to a different processor such that only the values of the boundary points of a partition have to be accessed by other processors [4,2]. Since performance is affected by both the computation and communication costs, the shape of partitions can have important effect on performance.

Historically, rectangular or square partitions have most commonly been assigned to processors, primarily because the resulting data structures can be easily indexed as two-dimensional arrays. Vrsalovic, et al. [4] considered the solution of Poisson's equation over a square region using a 5-point discretization stencil. They tested triangular, square, and hexagonal partitions. Reed, Adams and Patrick [3] conducted an analytical study on selecting optimal stencil/partition pairs. They considered rectangular, triangular, square and hexagonal partitions. If *computation to communication ratio* is used as the criterion for comparison, they found that square partitions are best for 9-point star stencils, hexagonal[1] partitions are best for 5-point stencils, 9-point cross stencils and 13-point stencils, and square and hexagonal partitions are equally good for

[1] As explained in Section 2, this kind of hexagons will be referred as R-hex.

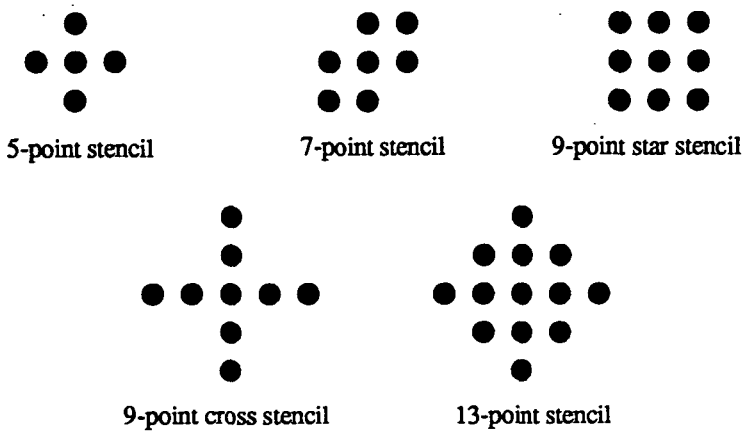

Figure 1: Five commonly used stencils

7-point stencils.

In this report, we will study problem partitioning of regular computation over two-dimensional meshes. We will show that for the various stencils (communication patterns) considered, there are other shapes of partitions which achieve higher computation to communication ratios than those previously discussed in the literature. In section 2, a systematic way to generate families of partitions using the concept of stencil neighborhood is presented. Section 3 discusses the properties of the partitions. Finally, in section 4 the computation to communication ratio is used as a metric to compare the performance of different partitioning schemes under various choice of stencils.

2 Generation of Partitions

In solving problems belonging to the class of regular computation, there is usually a choice of stencils. Figure 1 shows several commonly used stencils for two-dimensional meshes. The stencils to be considered in this report are the 5-point stencil, the 7-point stencil, the 9-point star stencil, the 9-point cross stencil and the 13-point stencil.

Let us define a *partition* to be a set of points in the two dimensional space Z^2, where Z is the set of integers. The *neighborhood*, $N(p)$, of a point, p, is a set which contains the point itself and some points positioned relative to the point, where N is called the neighborhood function. With this notation, we may denote the corresponding neighorhood function of the stencils considered above as N_5, N_7, N_{9s}, N_{9c}, and N_{13} respectively. For example, we can express N_5 and N_7 as follows:

$$\begin{aligned} N_5(p) &= \{p, p+(1,0), p+(0,1), p+(-1,0), p+(0,-1)\} \\ N_7(p) &= \{p, p+(1,0), p+(0,1), p+(-1,0), p+(0,-1), p+(1,1), p+(-1,1)\} \end{aligned}$$

Similary, the neighborhood functions for other stencils can be written down easily.

The *extension* of a partition P under the neighborhood function N is defined to be

$$E(P;N) = \{q : q \in N(p), p \in P\} \tag{1}$$

In other words, the extension of a partition is a new partition which contains exactly all the neighboring points of the points in the original partition.

Given any *seed* (initial) partition S, and neighborhood function N, we can recursively define a family of partitions as follows:

$$\begin{aligned} P_1 &= S \\ P_k &= E(P_{k-1};N) \quad \text{if } k > 1 \end{aligned} \tag{2}$$

If we denote $E(E(P;N);N)$ as $E^2(P;N)$, $E(E(E(P;N);N);N)$ as $E^3(P;N)$ and so on, we can rewrite the above *partition generation scheme* as follows:

$$P_k = E^{k-1}(S;N) \tag{3}$$

where $E^0(S;N) = S$.

Since it is actually the *geometric* properties of a partition which are important here, we consider two partitions P_1 and P_2 *equivalent* if P_2 is a *translation* of P_1, that is, if there exists a *translation vector* $u = (u_x, u_y)$ such that

$$P_2 = T(P_1;u) = \{q : q = p + u, p \in P_1\}$$

where T is the translation function. It is easy to see that the relation defined above is reflexive, symmetric and transitive, hence it is indeed an equivalence relation. This equivalence allows us to freely talk about the *shape* of the partitions without taking much care about the origin of the coordinate system. For our purposes, rotation equivalence and reflection equivalence are not considered here.

One type of seed we will use very often is a rectangle of size $m \times n$, denoted as $S_{m,n}$, where $S_{m,n} = \{(x,y) : 1 \le x \le m, 1 \le y \le n\}$. An important special case is the single-point seed, $S_{1,1}$. Suppose the seed is a single point, what kind of partitions will be generated if N is one of the corresponding neighborhood functions of the stencils shown in figure 1? Figure 2 shows the cases for N_5, N_7 and N_{9s}. We shall call these kinds of partitions *diamonds*, *hexagons* and *squares* respectively. It should be noted that this *hexagon* is different from the one as discussed in Reed's paper [3]. Reed's hexagon, denoted as *R-hex* here, is actually some kind of diamond partition with *variable* seed size, according to our classification. Suppose we choose N_5 as the neighboring function. If we set S to $S_{2,2}$ for generating P_1, S to $S_{3,2}$ for generating P_2, S to $S_{4,2}$ for generating P_3, and so on, we will get R-hex (see figure 2).

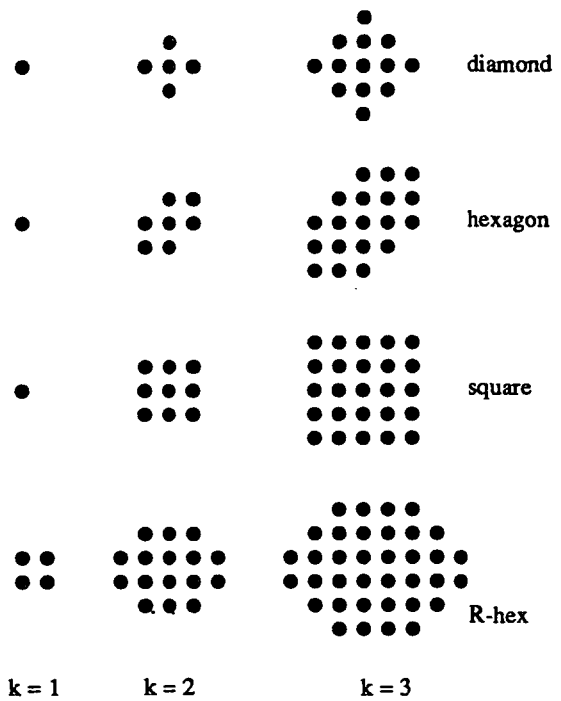

Figure 2: Four kinds of partitions

3 Properties of Partitions

A partition P is said to *tessellate* Z^2 if and only if for any finite region $R \subset Z^2$, there exists a finite number n of translation vectors u_i's such that

1. $R \subseteq \bigcup_{i=1}^{n} T(P; u_i)$ and
2. $T(P; u_i) \bigcap T(P; u_j) = \emptyset$ for all i, j.

In other words, a partition tessellates if some copies of it cover any given region without overlapping each other. In a given problem, if we use only one kind of partition which tessellates the 2-D plane Z^2, we may reduce the programming effort, because every processor will then see the same data structure and communication patterns (except possibly at boundaries).

In general, only some of the partitions of the form $E^k(S; N)$ tessellate the 2-D plane. The diamond, the hexagon, the square, and the R-hex are some examples. However, the family of partitions derived from the 9-point cross stencil, $E^k(S_{1,1}; N_{9c})$, does not tessellate, whereas those derived from the 13-point stencil also have the diamond shape. We will only consider those partitions which tessellate.

The *gird* of a partition under a neighborhood function N is defined to be

$$G(P;N) = E(P;N) - P \tag{4}$$

The gird points are exactly those *external* points which have to be accessed by a processor to which the partition is assigned. Figure 3 shows the girds of various partitions under different neighborhood functions (stencil structures).

It is very important to note that the neighborhood function N in equations 3 and 4 can be different. For example, if we start with a single-point seed, and choose N to be N_{9s} in equation 3, and N to be N_5 in equation 4, then the number of gird points is equal to $|G(E^{k-1}(S_{1,1};N_{9s});N_5)|$. However, interesting results do occur when the neighborhood function N in equations 3 and 4 are the same.

Since the neighborhood of a point includes itself by definition, it is obvious that $P \subseteq E(P;N)$. Combining this fact with equation 4, we have

$$E(P;N) = P \cup G(P;N) \tag{5}$$

Since $P \cap G(P;N) = \emptyset$ by definition of G (see equation 4), we also have

$$|E(P;N)| = |P| + |G(P;N)| \tag{6}$$

Suppose the family of partitions P_k is parametrized by S and N, then by applying equation 6 to the definition of P_k (equation 2), we have

$$\begin{aligned} |P_1| &= |S| \\ |P_k| &= |P_{k-1}| + |G(P_{k-1};N)| \quad \text{if } k > 1 \end{aligned} \tag{7}$$

Solving the recurrence equations, we get the formula for finding the size of a partition:

$$|P_k| = |S| + \sum_{i=1}^{k-1} |G_i| \tag{8}$$

where $G_i = G(P_i;N)$.

Equation 8 expresses the size of a partition in terms of its successive *layers* of girds. However, the size of a gird has to be found on a case by case basis. Tables 1 and 2 give the formula for $|G_k|$ and $|P_k|$ when the seed is a single point ($S_{1,1}$), and a rectangle ($S_{m,n}$) respectively. They can be readily derived by using mathematical induction.

It is interesting to note that $E^{k-1}(S_{1,1};N_{9s})$ only generates square with sides of odd length, with $|P_k| = 1+4k(k-1) = (2k-1)^2$, and $E^{k-1}(S_{2,2};N_{9s})$ only generates square with sides of even length, with $|P_k| = 4+(k-1)(4k+4) = (2k)^2$.

As a special case of diamond partitions with variable seed sizes, R-hex is generated as P_k, where $P_k = E^{k-1}(S_{k+1,2};N_5)$. Substituting $m = k+1$ and $n = 2$ into the formula for N_5 in table 2, we get $|G_k| = 6k+2$, and $|P_k| = 4k^2$.

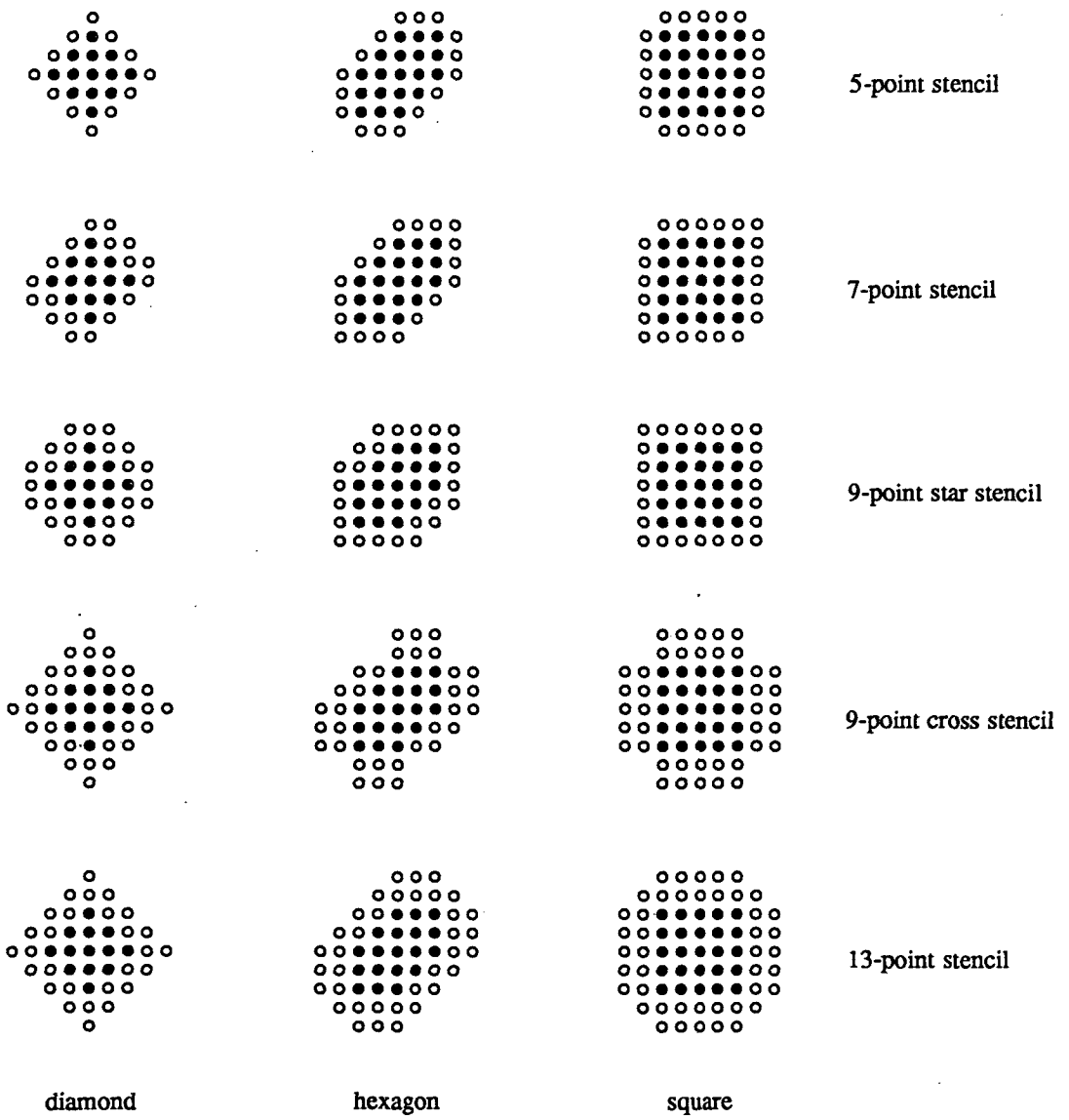

Figure 3: Girds of partitions with different stencils. The black circles are partition points, and the white circles are gird points.

Table 1: Size of partitions and girds with single-point seed ($S_{1,1}$)

name of partition	N	$\|G_k\|$	$\|P_k\|$
diamond	N_5	$4k$	$1+2k(k-1)$
hexagon	N_7	$6k$	$1+3k(k-1)$
square	N_{9s}	$8k$	$1+4k(k-1)$

Table 2: Size of partitions and girds with rectangular seed ($S_{m,n}$)

N	$\|G_k\|$	$\|P_k\|$
N_5	$4k+2m+2n-4$	$mn+(k-1)(2k+2m+2n-4)$
N_7	$6k+2m+2n-4$	$mn+(k-1)(3k+2m+2n-4)$
N_{9s}	$8k+2m+2n-4$	$mn+(k-1)(4k+2m+2n-4)$

4 Comparison of Partitions

For a given partition P with N_c as the stencil used in the communication, we assume that the amount of computation workload is equal to the size of the partition, $|P|$, and the amount of communication is equal to the size of the gird, $|G(P;N_c)|$. This assumption was also used in [3]. The *computation to communication ratio* is thus defined to be

$$CCR = |P|/|G(P;N_c)| \tag{9}$$

For example, if we use the 7-point stencil communication structure, but choose to divide the domain into diamond partitions P_k, then the amount of computation is equal to $1+2k(k-1)$, and the amount of communication is equal to $|G(P_k;N_7)| = 6k$ (see figure 3). Table 3 shows the amount of communication for the different combination of stencils and partitions.

Since the partitions have different *shapes*, it is not always possible to divide a given domain into idential subdomains such that each subdomain matches the right shape and size of a partition one would like to use. We may have to use a bigger partition of the same shape, but this may increase the amount of computation and change the pattern of communication. However, we can still compare the computation to communication ratio (CCR) of the various partitioning schemes in the asymptotic sense (see table 4). It is easy to see that for a given seed, the asymptotic CCR is independent of the seed itself. In table 4, A denotes the number of points contained in a partition. Note that different partitions may have different sets of possible A values.

Table 3: Amount of communication — $|G(P_k; N_c)|$

stencil,N_c	diamond	hexagon	square	diamond[$S_{m,n}$]	R-hex
N_5	$4k$	$6k-2$	$8k-4$	$4k+2m+2n-4$	$6k+2$
N_7	$6k$	$6k$	$8k-2$	$6k+2m+2n-4$	$8k+2$
N_{9s}	$8k$	$8k$	$8k$	$8k+2m+2n-4$	$10k+2$
N_{9c}	$8k+4$	$12k-4$	$16k-8$	$8k+4m+4n-4$	$12k+8$
N_{13}	$8k+4$	$12k$	$16k-4$	$8k+4m+4n-4$	$12k+8$

To calculate the (asymptotic) CCR, we let the area of a partition P_k be a constant A, and solve for k. For example, if we use the diamond partition P_k and the 7-point stencil, then we have

$$|P_k| = 1 + 2k(k-1) = A$$

Solving for k,

$$k = (\sqrt{2A-1}+1)/2$$

Hence,

$$\begin{aligned} CCR &= |P_k|/|G(P_k; N_7)| \\ &= A/6k \\ &= A/(3(\sqrt{2A-1}+1)) \\ &\asymp \sqrt{A/18} \end{aligned}$$

Similarly, we can derive the values in table 4 from tables 1, 2 and 3 according to equation 9.

From tables 4 we have the following observation:

1. In all the cases considered, CCR is proportional to $\sqrt{A}$. This is not surprising, because the size of a partition is a quadratic function of k, while the size of the corresponding gird is a linear function of k.

2. For each partition, CCR decreases or stays the same as $|N_c|$ increases. It is because for the same *area* A, the number of gird points increases or stays the same as there are more points contained in the communication stencil.

3. Diamond partitions yield the highest CCR ($\sqrt{A/8}$) for N_5, hexagons are best ($\sqrt{A/12}$) for N_7, squares are best ($\sqrt{A/16}$) for N_{9s}, and diamond partitions are also best ($\sqrt{A/32}$) for both N_{9c} and N_{13} stencils. This pattern suggests that

Table 4: Asymptotic computation to communication ratio (CCR)

stencil,N_c	diamond	hexagon	square	R-hex
N_5	$\sqrt{A/8}$	$\sqrt{A/12}$	$\sqrt{A/16}$	$\sqrt{A/9}$
N_7	$\sqrt{A/18}$	$\sqrt{A/12}$	$\sqrt{A/16}$	$\sqrt{A/16}$
N_{9s}	$\sqrt{A/32}$	$\sqrt{3A/64}$	$\sqrt{A/16}$	$\sqrt{A/25}$
N_{9c}	$\sqrt{A/32}$	$\sqrt{A/48}$	$\sqrt{A/64}$	$\sqrt{A/36}$
N_{13}	$\sqrt{A/32}$	$\sqrt{A/48}$	$\sqrt{A/64}$	$\sqrt{A/36}$

Table 5: Normalized asymptotic CCR

stencil,N_c	diamond	hexagon	square	R-hex
N_5	1.41	1.15	1	1.33
N_7	0.94	1.15	1	1
N_{9s}	0.71	0.87	1	0.8
N_{9c}	1.41	1.15	1	1.33
N_{13}	1.41	1.15	1	1.33

Table 6: Number of neighboring partitions (for P_k when $k \geq 3$)

stencil,N_c	diamond	hexagon	square	R-hex
N_5	4	6	4	6
N_7	6	6	6	6
N_{9s}	8	6	8	6
N_{9c}	8	6	4	6
N_{13}	8	6	8	6

there is a formal relationship between the optimal partition and the chosen stencil: The partition derived from a stencil N_c is the optimal partition in terms of computation to communication ratio when N_c is also the communication stencil, for $N_c = N_5, N_7, N_{9s}$.

4. The results on selecting the optimal stencil/partition pairs reported in [3] (see section 1) correspond to the last two columns of table 4.

5. R-hex is the second best whenever diamond is the best (when $N_c = N_5, N_{9c}, N_{13}$). It is the second worst whenever diamond is the worst (when $N_c = N_7, N_{9s}$). It is never the optimal partition in any of the cases considered.

Since square partitions probably result in most *regular* data structures, we are especially interested in knowing how well square partitions compare with other partitions. Hence, the *normalized* asymptotic CCR with respect to square partitions are calculated and displayed in table 5. It shows that square parition is never more than 41% worse than any other partitions under all the cases considered.

For our purpose of finding the optimal partitions under different cases, rectangular stripes, rectangular partitions and triangular partitions are not considered. They have been previously shown to be inferior to squares or R-hex's [3].

Good peformance involves many factors. Communication cost not only depends on the total amount of communication, but also depends on the actual patterns of communication, such as the number of communicating neighbors (see table 6) and the underlying machine architectures. This report intends to give the asymptotic bound on one of the issue — optimal partitioning with respect to the computaion to communication ratio. Maximizing the computation to communication ratio does not necessarily guarantee minimum execution time of a parallel program, but it is still an important indicator of the potential performance of the program. It is interesting to see how much this ratio varies under different combination of stencils and partitions.

5 Conclusion

This report has presented an analysis for selecting optimal partitions for regular computation over two-dimensional meshes given the communication stencil. The criterion used is the computation to communication ratio, which is defined to be the ratio of the size of a partition to that of its gird. It is shown that diamond partitions are best for 5-point stencils, 9-point cross stencils and 13-point stencils, hexagonal partitions are best for 7-point stencils, and square partitions are best for 9-point star stencils.

References

[1] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the navier-stokes equation. In *Physical Review Letters*, pages 1505–1508, 1986.

[2] Fung F. Lee and George B. Adams III. *Implementation of Lattice Gas Cellular Automata on Hypercubes.* Technical Report 87.27, Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, 1987.

[3] Daniel A. Reed, Loyce M. Adams, and Merrell L. Patrick. *Stencils and Problem Partitionings: Their Influence on the Performance of Multiple Processor Systems.* Technical Report 86-24, Institute for Computer Applications in Science and Engineering (ICASE), 1986.

[4] D. Vrsalovic, E. F. Gehringer, Z. Z. Segall, and D. P. Siewiorek. The influence of parallel decomposition strategies on the performance of multiprocessor systems. In *Proceedings of the 12th Annual International Symposium on Computer Architecture*, pages 396–405, IEEE Computer Society Press, June 1985.