The cookie cutter algorithm: asymptotic optimality for grid-graph partitioning

Winston C. Yang* winston@cs.wisc.edu
Robert R. Meyer rrm@cs.wisc.edu

Computer Sciences Department
University of Wisconsin-Madison

June 17, 2003

Abstract

For rectangular grid-graphs, a new algorithm called the cookie cutter algorithm generates asymptotically optimal partitions, in the sense that as problem sizes grow, the ratios of the corresponding partition perimeters over their optimal values tend to 1. Previous researchers established the asymptotic optimality of a different type of partition based on stripe algorithms, but the cookie cutter algorithm has different asymptotics and has the additional advantage of anticipated generalization to triangular and hexagonal meshes.

1 Introduction

Graph partitioning is a well-known NP-complete problem: partition the set of nodes of a graph into a specified number of equal-size subsets as to minimize the number of arcs between nodes in different subsets (such arcs are called *cut-arcs*). Graph partitioning models certain applied problems. For example, consider the parallel processing problem of dividing assigning a set of

*Supported by NSF grant DMI-0100220

jobs among a collection of parallel processors so as to satisfy load-balancing (each processor has the same number of jobs and the partition is said to be balanced) and so as to minimize interprocessor communication (because the exchange of results between jobs in different processors can slow down the system). To employ the graph analogy, a job is a node, the jobs assigned to processor are a subset, and there is an arc between jobs if and only if the jobs share data. Researchers have proposed many heuristic algorithms for solving graph partitioning, including Kernighan-Lin bisection [9], geometric mesh [5], recursive spectral bisection [10], and graph coarsening [8].

We use the terms *graphs* and *meshes* interchangeably. Partitioning problems for meshes (the focus of this paper) can be reformulated as graph partitioning problems, but in this paper we maintain the original geometric geometric mesh context (described in detail below) because the partition lower bound results and the cookie cutter algorithm are more easily envisioned in that context. The term *domain decomposition* is often used to describe the special case of graph partitioning that arises when the graph is a *grid-graph*, corresponding to square, triangular, or hexagonal grids. This paper focuses on grid-graphs corresponding to square meshes (or rectangular meshes, since they are equivalent from a graph viewpoint), and it appears likely that the results can be extended to triangular and hexagonal meshes.

Square meshes have been investigated by Yackel, Meyer, and Christou [3] and Donaldson [4]. Motivation for considering grid-graphs made of squares comes from the rectangular tables in relational databases [1] and partial differential equations, and finite element applications in which the physical plane domain under investigation is subdivided into square cells. For such graphs, the geometric viewpoint has squares centered at the nodes. See Figure 1. The *domain* is the set of squares centered at the nodes. A *subdomain* is a set of squares with the same label. A subdomain boundary edge is *internal* iff it is shared by 2 subdomains; else, it is *external*. The number of cut-arcs is related to the total subdomain perimeter as follows (see Figure 1):

$$\begin{aligned}\text{total subdomain perimeter} &= 2\#\text{internal edges} + \#\text{external edges}\\ &= 2\#\text{cut-arcs} + \#\text{external edges}\\ &= 2\#\text{cut-arcs} + \text{constant}\end{aligned}$$

Because of the preceding affine relationship between total subdomain perimeter and the number of cut-arcs, minimizing one is equivalent to mini-

Figure 1: Equivalence of a grid-graph (left) and a domain made of squares (right). Minimizing number of cut-arcs in graph partitioning is equivalent to minimizing total subdomain perimeter in domain decomposition.

mizing the other. See Figure 2 for an example of an optimal domain decomposition.

In the next section, we discuss previous research on rectangular domain decomposition with emphasis on geometric approaches. Tilings of such domains with optimal shapes are presented in section 3, and the application of the cookie cutter method with those tilings as input is described in section 4. This paper concludes with a discussion of the applicability of this approach to triangular and hexagonal meshes.

2 Previous research on rectangular domain decomposition

Partitions of square meshes produce subdomains that are collections of squares, so we begin this overview of geometry-based approaches to domain decomposition by discussing the terminology used in combinatorics and in our research to describe such subdomains.

A *polyomino* [6] is a finite edge-connected set of squares on a square grid. See Figure 4. "Polyomino" is a generalization of "domino". The *area* is the

Figure 2: Optimal decomposition of a 22×22 domain into 22 subdomains, each with area 22. The total subdomain perimeter is 440, which is minimal.

number of squares, and the *perimeter* is the number of unit boundary edges.

The relation of polyominos to domain decomposition is very simple; all subdomains produced by the decomposition are connected (unless the domain is disconnected or has some unusual shape, but such cases are not of interest here), so the subdomains are polyominos. Theorem 1 (Polyomino min perimeter formula), which has many proofs [3, 7, 11], gives a lower bound for each subdomain perimeter in a partition.

Theorem 1 (Polyomino min perimeter formula). *The min perimeter of a polyomino with area A is $2\left\lceil \sqrt{4A} \right\rceil$.*

Harary and Harborth [7] and Yang [11] proved Theorem 2 (Polyomino swirl algorithm), which constructs polyominos with min perimeter. It validates the intuitive idea that a polyomino that resembles a square (more specifically, a *quasi-square*, a rectangle whose sides differ by at most 1) has min perimeter.

Theorem 2 (Polyomino swirl algorithm). *See Figure 3. To construct a polyomino with area A and min perimeter, follow the swirl for A squares.*

Ideally, a decomposition will consist of min-perimeter (*optimal*) subdomains, as specified in Theorem 1 (Polyomino min perimeter formula). Thus

Figure 3: Polyomino swirl algorithm. It is possible to arrange the swirl so that it is a spiral (right).

a lower bound for the total subdomain perimeter may be obtained by multiplying this lower bound by the number of subdomains. Specifically, for a decomposition of a domain into S subdomains, each of which has area A, we can apply the min perimeter theorem to obtain the following lower bound for the total perimeter of the decomposition.

Theorem 3 (Total subdomain perimeter lower bound). *The total subdomain perimeter for a domain decomposition with S subdomains, each with area A, is $2S\left\lceil\sqrt{4A}\right\rceil$.*

As will be discussed below, this lower bound is best possible in an asymptotic sense.

However, unless the optimal subdomains are shaped like squares or rectangles (which is not the case unless the subdomain area has the form $A = B^2$ or $A = B(B+1)$), it is seldom possible to decompose a given domain into only optimal subdomains, and the presence of any non-optimal subdomain implies that the theoretical lower bound will not be attained. Using optimal subdomains as a goal, the cookie cutter algorithm uses as input a planar tiling with optimal subdomains, and then reassigns the cells of subdomains near the boundary of the domain (such cells are relatively small in number), while preserving the "internal" optimal subdomains. For very large domains, the number of internal subdomains will be very much larger than the number of boundary subdomains, and a careful analysis of this relationship establishes the asymptotic optimality of the cookie cutter approach.

Figure 4: The 12 polyominos with 5 squares. All have perimeter 12 except for one shaped like a 1×1 square joined to a 2×2 square, which has perimeter 10.

Previous geometry-based research in this area was not based on tilings, but instead focused on stripe decomposition. A domain decomposition is *striped* iff many of the subdomains are arranged in horizontal stripes. For example, the decomposition in Figure 2 is striped. For polyominos, it can be shown that for appropriately chosen stripe heights, cells within stripes may be assigned so that most of the subdomains have near-optimal perimeter. Using such stripe decompositions, Christou and Meyer [2] proved Theorem 4 (Perimeter relative gap). The *perimeter relative gap* is

$$100\% \times \left(\frac{\text{total subdomain perimeter}}{\text{total subdomain perimeter lower bound}} - 1 \right).$$

Theorem 4 (Perimeter relative gap). *Let $S|RC$ (that is, S divides RC) and $S \geq \max(R, C)$. Decompose a rectangular $R\times C$ domain made of squares into S subdomains, each with area $A = RC/S$. There is a stripe decomposition whose perimeter relative gap differs from the lower bound by less than $1/\sqrt{A} + 1/A$.*

Note that in Theorem 4 (Perimeter relative gap), the relative gap goes to 0 as the subdomain area goes to infinity. A drawback of the stripe approach is that it does not yield asymptotic optimality for triangular meshes [11] because stripes cannot be tiled with optimal or near-optimal shapes constructed from triangles. However, the cookie cutter algorithm (discussed later) does not use stripes and is expected to produce asymptotically optimal partitions. For large domains comprised of squares, the cookie cutter algorithm also yields better decompositions than the stripe approach, in that most subdomains have min perimeter. A consequence of this property of the cookie cutter algorithm is that the subdomain area does not have to go to infinity in order that the relative gap go to 0.

3 Tilings of polyominos with min perimeter

Motivation for considering tilings comes from the fact that domain decomposition is essentially a (finite) tiling problem. Whether or not all subdomains can have min perimeter depends heavily on the domain boundary; subdomains touching it must conform to it and so might not be able to have min perimeter. To investigate the best possible situation for domain decomposition, remove these "boundary conditions"; consider tilings of the plane. Then impose the boundary conditions via a cookie cutter and modify assignments near the boundary to achieve a balanced partition.

The next theorem establishes the validity of the first step in the process by showing that, in the square case, the plane may actually be tiled using only optimal shapes. (Tiling issues are more difficult for triangular and hexagonal cells and are discussed in section 5.)

Theorem 5 (Polyomino min perimeter tiling). *For all areas A, there is a polyomino with area A and with min perimeter that tiles the plane.*

Proof. Use Theorem 2 (Polyomino swirl algorithm) to construct a polyomino with min perimeter. Note that all polyominos it constructs have dimensions that differ by at most 1. There are 2 cases:

- Case: The polyomino is a rectangle. Obviously, it tiles the plane.
- Case: The polyomino is not a rectangle. Rotate and reflect the polyomino so that the number of rows is at most the number of columns. Proceed as in Figure 5. Note that putting the indented part of a polyomino over the jutted part of another polyomino creates a space that another polyomino can fit into. □

4 Cookie cutter algorithm

Given a set of bounded domains with suitable properties, the cookie cutter algorithm suitably modifies planar tilings to produce domain decompositions

Figure 5: Tiling the plane with a polyomino with min perimeter that is not a rectangle.

whose perimeters are asymptotically optimal. We first describe the algorithm in detail, and then establish its asymptotic optimality.

Algorithm 1 (Cookie cutter algorithm). *See Figure 6. To decompose a domain, do the following:*

1. *Construct a tiling of the plane with subdomains with min perimeter.*
2. *Place the domain (cookie cutter) onto the tiling (pre-scored cookie dough).*
3. *Some subdomains touch the domain boundary, and some do not. These are* fringe *and* internal *subdomains, respectively. See Figure 6.*
4. *Erase all lines forming the boundary between* 2 *fringe subdomains.*
5. *Partition (in a near-optimal manner) the fringe into subdomains with the desired area.*

The reason the cookie cutter algorithm works well is that the resulting partition consists mostly of internal subdomains with min perimeter and includes only a relatively small number of fringe subdomains that contribute little to the total perimeter. For example, consider decomposition of an $N \times N$ domain into N subdomains, each with area N. It turns out that the total perimeter of internal subdomains perimeter is $\mathcal{O}(N^{3/2})$, the perimeter contribution of the redrawn fringe subdomains is $\mathcal{O}(N)$, and the total subdomain perimeter lower bound is $\mathcal{O}(N^{3/2})$. So if N is large, the perimeter contribution of the redrawn fringe subdomains is negligible. The proof of Theorem 6 (Square asymptotic optimality) is based on these ideas, but is stated for the more general case of rectangular domains.

Figure 6: Cookie cutter algorithm: to decompose a domain, place the domain (cookie cutter) onto a tiling of the plane by subdomains with min perimeter. Delete the original subdomain boundaries in the fringe (shaded region), replacing them by new boundaries that partition the fringe in a near-optimal manner. (In this example, the domain is 28×32, and the subdomain area is 28.)

Figure 7: Redraw the fringe subdomains in Figure 6. The numbers are the lengths of the new subdomain separators. (In this example, $A = 28$, so $\sqrt{28} + 3$ bounds the length of a fringe subdomain separator.)

Theorem 6 (Square asymptotic optimality). *On a square grid, consider a cookie cutter decomposition of an $R \times C$ domain into S subdomains with area $A = RC/S$. The perimeter ratio is*

$$\frac{\textit{total subdomain perimeter}}{\textit{total subdomain perimeter LB}}$$
$$\leq \; 1 + \left(\frac{1}{R} + \frac{1}{C}\right)\left(\frac{3}{2}\sqrt{A} + 11\right) + \frac{2A}{RC} + \frac{6\sqrt{A}}{RC} \xrightarrow{\frac{\sqrt{A}}{R}, \frac{\sqrt{A}}{C} \to 0} 1.$$

Thus the cookie cutter algorithm produces asymptotically optimal domain decompositions, if the number of rows and columns tends to infinity at a rate that appropriately dominates the growth rate of the subdomain size.

Proof. See Figure 6. The following calculations bound the cookie cutter decomposition perimeter and compare it to the total subdomain lower bound.

1. **Fringe width**: $\leq \sqrt{A} + 2$

   Theorem 2 (Polyomino swirl algorithm) constructs polyominos with min perimeter whose sides differ by at most 1. For area A, it constructs a polyomino with sides $\geq \sqrt{A}$. However, the longer side of a polyomino is $\leq \sqrt{A} + 1$. The fringe width can comprise of such a longer side plus an indent of an adjacent polyomino, which adds 1 to the length. See the bottom right corner of the domain in Figure 7 for such a situation.

2. **Fringe outer perimeter**: $2(R + C)$

   This is the boundary of the original domain.

3. **Fringe area**: $\leq 2(R + C)(\sqrt{A} + 2)$

   Multiply fringe outer perimeter by fringe width.

4. **Number of fringe subdomains**: $\leq 2(R + C)\left(\frac{1}{\sqrt{A}} + \frac{2}{A}\right)$

   Divide fringe area by subdomain area. Note that the actual number of fringe subdomains is the integer $S - I$, where I is the not-easily-calculable number of internal subdomains. However, the above bound is sufficient for our purposes.

5. **Fringe subdomain separator length**: $\leq \sqrt{A} + 3$.

   In the ideal case, the separator is straight and has length equal to the fringe width bound $\sqrt{A} + 2$, but sometimes it needs a bend in the middle; such a dogleg adds 1 to the length.

6. **Number of fringe subdomain separators, counting multiplicities**: $\leq 4(R+C)\left(\frac{1}{\sqrt{A}} + \frac{2}{A}\right) + 8$

   For non-corner subdomains, count each one 2 times because it is shared by 2 subdomains; multiply number of fringe subdomains by 2. For a subdomain that contains a corner cell, an additional separator may be needed, so counting multiplicities, an additional constant term with value 8 is added to the separator count.

7. **Total fringe subdomain separator length**:

   $$\leq 4(R+C)\left(1 + \frac{5}{\sqrt{A}} + \frac{6}{A}\right) + 8\sqrt{A} + 24$$

   Multiply fringe subdomain separator length by number of fringe subdomain separators.

8. **Total perimeter added by fringe subdomains**:

   $$\leq 4(R+C)\left(\frac{3}{2} + \frac{5}{\sqrt{A}} + \frac{6}{A}\right) + 8\sqrt{A} + 24$$

   Add fringe outer perimeter to total fringe subdomain separator length.

9. **Total subdomain perimeter lower bound**: $\frac{RC}{A} 2\left\lceil \sqrt{4A} \right\rceil$

   Multiply number of subdomains by subdomain min perimeter.

10. **Total internal subdomain perimeter**: $< \frac{RC}{A} 2\left\lceil \sqrt{4A} \right\rceil$

11. **Cookie cutter decomposition perimeter**:

    $$\leq \frac{RC}{A} 2\left\lceil \sqrt{4A} \right\rceil + 4(R+C)\left(\frac{3}{2} + \frac{5}{\sqrt{A}} + \frac{6}{A}\right) + 8\sqrt{A} + 24$$

Add total fringe length perimeter to total non-fringe internal subdomain perimeter.

12. **Lower bound for total perimeter**: $\frac{RC}{A} 2 \left\lceil \sqrt{4A} \right\rceil$

    Multiply number of subdomains by subdomain min perimeter.

13. **Ratio of cookie cutter decomposition perimeter to total subdomain perimeter lower bound**:

$$
\begin{aligned}
& \leq \frac{\frac{RC}{A} 2 \left\lceil \sqrt{4A} \right\rceil + 4(R+C)\left(\frac{3}{2} + \frac{5}{\sqrt{A}} + \frac{6}{A}\right) + 8\sqrt{A} + 24}{\frac{RC}{A} 2 \left\lceil \sqrt{4A} \right\rceil} \\
& = 1 + \frac{4(R+C)\left(\frac{3}{2} + \frac{5}{\sqrt{A}} + \frac{6}{A}\right) + 8\sqrt{A} + 24}{\frac{RC}{A} 2 \left\lceil \sqrt{4A} \right\rceil} \\
& \leq 1 + \frac{\cdots}{\frac{RC}{A} 2\sqrt{4A}} \\
& = 1 + \frac{\cdots}{\frac{4RC}{\sqrt{A}}} \\
& = 1 + \left(\frac{1}{R} + \frac{1}{C}\right)\left(\frac{3}{2}\sqrt{A} + 5 + \frac{6}{\sqrt{A}}\right) + \frac{2A}{RC} + \frac{6\sqrt{A}}{RC} \\
& \leq 1 + \left(\frac{1}{R} + \frac{1}{C}\right)\left(\frac{3}{2}\sqrt{A} + 11\right) + \frac{2A}{RC} + \frac{6\sqrt{A}}{RC} \quad \square
\end{aligned}
$$

5 Future research

Future goals include generalizing Theorem 6 (Square asymptotic optimality) to triangular and hexagonal meshes. In the case of triangular meshes, a subdomain is a *polyiamond*, a finite edge-connected set of triangles. A *polyhex* is defined similarly, except it consists of hexagons. In contrast to the optimal polyomino case (Theorem 5), there are areas A such that no polyiamond with area A and min perimeter tiles the plane [11]. However, for such areas, we conjecture that polyiamond tilings with near-optimal shapes exist for which the deviation from min perimeter is bounded by a constant that is independent of area. If this conjecture is correct, a suitably-modified version of the cookie cutter algorithm achieves asymptotic optimality, provided that the subdomain areas also tend to infinity (so that bounded deviations from subdomain optimal perimeter become unimportant). For polyhexes, the situation is closer to the polyomino case; for areas A up to 100, we have been able to construct tilings with optimal polyhexes, and we conjecture that such tilings exist for all areas A [11]. If the polyhex tiling conjecture is true, then the cookie cutter asymptotic optimality theorem generalizes to hexagonal meshes in a straightforward manner.

References

[1] P. Berman, B. DasGupta, S. Muthukrishnan and S. Ramaswami, Improved approximation algorithms for rectangle tiling and packing, Proc. 12th ACM-SIAM Symp. on Disc. Alg., 2001, 427–436.

[2] I. T. Christou and R. R. Meyer. Optimal equi-partition of rectangular domains for parallel computation. Journal of Global Optimization, 8:15–34, 1996.

[3] J. Yackel, R. R. Meyer, and I. T. Christou. Minimum-perimeter domain assignment. Mathematical Programming, 78:283–303, 1997.

[4] W. W. Donaldson. Grid-graph partitioning. PhD thesis, University of Wisconsin, Madison, WI, 2000.

[5] J. R. Gilbert, G. L. Miller, and S.-H. Teng. Geometric mesh partitioning: implementation and experiments. SIAM Journal of Scientific Computing, 1995.

[6] S. W. Golomb. Polyominoes: puzzles, patterns, problems, and packings, 2nd ed. Princeton University Press, Princeton, NJ, 1994.

[7] F. Harary and H. Harborth. Extremal animals. Journal of Combinatorics, Information, & System Sciences, 1:1–8, 1976.

[8] G. Karypis, R. Aggarwal, K. Schloegel, V. Kumar, and S. Shekhar. METIS (set of programs for partitioning unstructured graphs and hypergraphs and computing fill-reducing orderings of sparse matrices), `www-users.cs.umn.edu/~karypis/metis/`, 2002.

[9] B. W. Kernighan and S. Lin. An effective heuristic procedure for partitioning graphs. Bell Systems Technical Journal, pages 291–308, 1970.

[10] H. D. Simon and S.-H. Teng. How good is recursive bisection? SIAM Journal of Scientific Computing, 18(5):1436–1445, 1997.

[11] W. C. Yang. Optimal polyform domain decomposition. PhD thesis, University of Wisconsin, Madison, WI, 2003.