Equi-partitioning of Higher-dimensional Hyper-rectangular Grid Graphs

Athula Gunawardena
Department of Mathematical and Computer Sciences
University of Wisconsin-Whitewater
http://www.uww.edu/
gunawara@uww.edu, and
Department of Computer Science, University of Wisconsin-Madison
agunawardena@wisc.edu

Robert R. Meyer
Department of Computer Science, University of Wisconsin-Madison
http://www.cs.wisc.edu/
rrm@cs.wisc.edu

Abstract

A d-dimensional grid graph \( G \) is the graph on a finite subset in the integer lattice \( \mathbb{Z}^d \) in which a vertex \( x = (x_1, x_2, \cdots, x_n) \) is joined to another vertex \( y = (y_1, y_2, \cdots, y_n) \) if for some \( i \) we have \( |x_i - y_i| = 1 \) and \( x_j = y_j \) for all \( j \neq i \). \( G \) is hyper-rectangular if its set of vertices forms \([K_1] \times [K_2] \times \cdots \times [K_d]\), where each \( K_i \) is a nonnegative integer, \( [K_i] = \{0, 1, \cdots, K_i - 1\} \). The surface area of \( G \) is the number of edges between \( G \) and its complement in the integer grid \( \mathbb{Z}^d \). We consider the Minimum Surface Area problem, \( MSA(G, V) \), of partitioning \( G \) into subsets of cardinality \( V \) so that the total surface area of the subgraphs corresponding to these subsets is a minimum. We present an equi-partitioning algorithm for higher dimensional hyper-rectangles and establish related asymptotic optimality properties. Our algorithm generalizes the two dimensional algorithm due to Martin [8]. It runs in linear time in the number of nodes \( O(n) \) when each \( K_i \) is \( O(n^{1/d}) \). Utilizing a result due to Bollobas and Leader [3], we derive a useful lower bound for the surface area of an equi-partition. Our computational results either achieve this lower bound (i.e., are optimal) or stay within a few percent of the bound.

| Article Type | Communicated by | Submitted | Revised |
|--------------|----------------|-----------|---------|
| Regular paper | X. He | April 2005 | January 2007 |

The research of Robert R. Meyer was supported in part by NSF grant DMI-0100220 and DMI-0400294.
1 Introduction

Grid graph equi-partition arises in the context of minimizing interprocessor communication subject to load balancing in parallel computation for a variety of problem classes including the solution of PDEs using finite difference schemes [11], computer vision [10], and database applications [7]. Similar to the general graph partitioning problem [6], arbitrary grid graph partitioning problem is NP-complete [14], and at present one has to depend on effective heuristic algorithms.

In this paper, we focus on partitioning graph analogs of hyper-rectangles and hereafter call our problem the Minimum Surface Area problem for rectilinear grid graphs, denoted by $M\!S\!A(G, V)$, in which the d-dimensional grid graph $G = [K_1] \times [K_2] \ldots \times [K_d]$ is partitioned into subgraphs of cardinality $V$ so that the total surface area of the subgraphs is a minimum. We assume that $V$ divides $K_1 K_2 \ldots K_d$. Figure 1 shows a 25-component equi-partition for the $M\!S\!A([5]^3, 5)$ problem generated by our equi-partitioning algorithm presented in Section 4. Since the surface area of this partition does not equal the lower bound derived in Section 3, its optimality is not guaranteed but its relative optimality gap $[(\text{area-lower bound})(=2)/\text{lower bound}(=500)]$ is 0.04%, and only one component allows any possibility for surface area improvement.

Several effective algorithms for partitioning a 2-dimensional grid graph are available in the literature [2, 4, 5, 8, 15]. Our algorithm generalizes the fast two dimensional algorithm due to Martin [8]. With this generalization, we reduce the problem to several integer knapsack problems (as defined in [12]) and use dynamic programming to solve them.

Those grid graphs that meet the minimum surface area for a given number of vertices draw special interest from research in Combinatorics (i.e., study of Polyominoes) [1], and in Physics (i.e., study of the metastable behavior of the stochastic ising model) [9]. In 1991, Bollabas and Leader [3] derived a class of grid graphs with minimum surface area. In Section 3, utilizing their result, we provide a formula for the minimum surface area for grid graphs with a given cardinality and use this result to find a lower bound for the surface area of a given equi-partition. We present our equi-partitioning algorithm in Section 4.
and give computational results in Section 5 for some 3 and 4-dimensional equipartitions. We also compare our results with the lower bound given in Section 3.

2 Preliminaries

We follow the notation and definitions given in [3]. For the readers’ convenience, we restate them here. A d-dimensional grid graph is the graph on a finite subset in the integer lattice $\mathbb{Z}^d$ in which $x = (x_1, x_2, \ldots, x_n)$ is joined to $y = (y_1, y_2, \ldots, y_n)$ if for some $i$ we have $|x_i - y_i| = 1$ and $x_j = y_j$ for all $j \neq i$. Since grid graphs are determined by their nodes, in much of the discussion below we identify the graph with its nodes. Let $P([n]) = \{0, 1, \ldots, n-1\}$. The binary order on $P([n])$ is a total order on $P([n])$’s $2^n$ elements such that for some $S, T \in P$, $S < T$ if and only if the greatest element of $[n]$ which is in one of $S$ and $T$ but not the other is actually in $T$ (i.e., $\max(S \Delta T) \in T$). This is equivalent to a lexicographical ordering (based on right most differing elements) defined on an indicator mapping in $\mathcal{B}^n$ (the set of binary $n$-tuples) of elements of $P([n])$. The cube order on $[k]^n$ is a total order such that for some $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$, $x < y$ if and only if for some $s \in [k]$, $\{i : x_i = s\} < \{i : y_i = s\}$ in the binary order on $P([n])$ with $\{i : x_i = t\} = \{i : y_i = t\}$ for all $t > s$. For $S \subseteq [k]^n$, the cube order on $S$ is the restriction to $S$ of the cube order on $[k]^n$. We write $x + [k]$ to represent $\{x, x + 1, \ldots, x + k - 1\}$. Let $G$ be a grid graph and $E$ be the set of edges in $\mathbb{Z}^n$. The edge-boundary of $G$, denoted by $\sigma(G)$, is the set $\{(x, y) \in E | x \in G, y \notin G\}$ and the edge-interior of $G$, denoted by $int(G)$, is the set $\{(x, y) \in E | x, y \in G\}$. The surface area of $G$, denoted by $\mathcal{A}(G)$, is the cardinality of $\sigma(G)$ ($\|\sigma(G)\|$) (In three dimensional space this matches the “exposed” surface area of the set of unit hypercubes with centers at the nodes of $G$). In [3], for a given positive integer $m$, bounds have been derived for two optimization problems, minimize $\{\mathcal{A}(G) | \|G\| = m\}$ and maximize $\{|int(G)| | \|G\| = m\}$, when $G \subseteq [k]^n$. In $[k]^n$, $k \geq 3$, these two problems are not equivalent [3]. Since our underlying domain, $\mathbb{Z}^n$, is regular, these two problems are equivalent. Our focus on $\mathbb{Z}^n$ is justified by the applications to database, computer vision, and domain decomposition for partial differential equations [7, 10, 11]. The following proposition states that the cube order may be used to create graphs with minimum surface area. This is a direct corollary to Theorem 15 in [3]. Other versions of proofs of this result can be found in [9], [1] ($d = 3$), and [13].

**Proposition 1** Let $G$ be a d-dimensional grid graph such that $G \subseteq [k]^d$ for some large enough positive integer $k$, and $J$ be the set of first $\|G\|$ elements in the cube order on $[k]^d$. Then $\mathcal{A}(G) \geq \mathcal{A}(J)$.

Let $d$ be a positive integer and $n_d$ be a nonnegative integer. Let $k_d$ be the unique integer such that $k_d^d \leq n_d < (k_d + 1)^d$. Thus, $k_d^d$ is the largest cube whose volume does not exceed $n_d$. Let $m_d$ be the unique integer in $[d]$ such
that \((k_d + 1)^m d^{d-m_d} \leq n_d < (k_d + 1)^{m_d+1}(k_d)^{d-m_d-1}\). Thus, \(m_d\) characterizes a “largest near-cube” which is a hyper-rectangle whose dimensions differ by at most 1. Observe that \(k_1 = n_1\) and \(m_1 = 0\). For an integer \(n_d\), we define the following special grid graph, called the quasi-cube recursively, and we show in Section 3 that quasi-cubes are generated by the cube order.

**Definition 1 (Quasi-Cube)** The \(d\)-dimensional quasi-cube corresponding to \(n_d\) is denoted by \(C_d(n_d)\). For \(d = 1\), \(C_1(n_1) = [n_1]\). For \(d > 1\), \(C_d(n_d)\) is the set

\[
[k_d + 1]_{m_d} \times [k_d]_{d-m_d} \cup P(C_{d-1}(n_{d-1})),
\]

where \(k_d\) and \(m_d\) are unique nonnegative integers as mentioned above, \(n_{d-1} = n_d - (k_d + 1)^{m_d}(k_d)^{d-m_d}\), and \(P(C_{d-1}(n_{d-1}))\) is the set of points,

\[
(x_1, x_2, \ldots, x_{m_d}, k_d, x_{m_d+1}, \ldots, x_{d-1}), \text{ such that } (x_1, \ldots, x_{d-1}) \in C_{d-1}(n_{d-1}).
\]

Essentially the quasi-cube, \(C_d(n_d)\), is obtained by first extracting the “largest possible” \(d\)-dimensional near-cube, then recursively applying the extraction to the residual volumes in successively lower dimensions. (See three dimensional figures in [1].) Since \(k_i\) and \(m_i\) \((d \geq i \geq 1)\) are unique integers, \(C_d(n_d)\) is uniquely defined.

## 3 A Lower Bound for the Total Surface Area of an Equi-partition

We use the quasi-cube to obtain our results in this section. The following lemmas describe some of the properties of the quasi-cube.

**Lemma 1** The quasi-cube has the properties:

\((a)\) \(|C_d(n_d)| = n_d = \sum_{r=1}^{d} (k_r + 1)^{m_r} k_r^{r-m_r}\).

\((b)\) \(A(C_d(n_d)) = 2 \sum_{r=1}^{d} [(r - m_r)(k_r + 1)^{m_r} k_r^{r-m_r-1} + m_r(k_r + 1)^{m_r-1} k_r^{r-m_r}]\).

**Proof:** By Definition 1, \(C_d(n_d) = S_d \cup P(C_{d-1}(n_{d-1}))\) where \(S_d = [k_d + 1]_{m_d} \times [k_d]_{d-m_d}\). According to their definitions, \(S_d\) and \(P(C_{d-1}(n_{d-1}))\) are disjoint and \(|P(C_{d-1}(n_{d-1}))| = |C_{d-1}(n_{d-1})|\). So we have \(|C_d(n_d)| = |S_d| + |C_{d-1}(n_{d-1})|\), and get (a) by applying induction on \(d\).

To prove (b), we consider the surface area of \(C_d(n_d)\). We have \(A(C_d(n_d)) = A(S_d) + A(P(C_{d-1}(n_{d-1}))) - 2[\sigma(S_d) \cap \sigma(P(C_{d-1}(n_{d-1})))].\) Since \(k_{d-1} \leq k_d\), we get \(\sigma(P(C_{d-1}(n_{d-1}))) \subseteq \sigma(S_d)\). Using \(A(S_d) = 2(d - m_d)(k_d + 1)^{m_d} k_d^{d-m_d-1} + 2m_d(k_d + 1)^{m_d-1} k_d^{d-m_d} + A(P(C_{d-1}(n_{d-1}))) = A(C_{d-1}(n_{d-1})) + 2|C_{d-1}(n_{d-1})|\), and \(|\sigma(S_d) \cap \sigma(P(C_{d-1}(n_{d-1}))) = |C_{d-1}(n_{d-1})|\), we get the recurrence relation \(A(C_d(n_d)) = A(C_{d-1}(n_{d-1})) + 2(d - m_d)(k_d + 1)^{m_d} k_d^{d-m_d-1} + 2m_d(k_d + 1)^{m_d-1} k_d^{d-m_d} + A(P(C_{d-1}(n_{d-1})))\).
Let $4$ Equi-partitioning Algorithm

Proof: First we prove that the quasi-cube $C_d(n_d)$ forms a segment of the cube order in dimension $d$. We prove this by induction on $d$. Clearly $C_1(n_1) = \{n_1\}$ is an initial segment of the cube order. Assume $C_{d-1}(n_{d-1})$ is an initial segment of the cube order. By the definition, $C_d(n_d) = S_d \cup P(C_{d-1}(n_{d-1}))$ where $S_d = [k_d]^{m_d} \times [k_d+1]^{d-m_d}$. Let $x = (x_1, x_2, \ldots, x_d)$, where $x_i = k_d - 1$ if $i \leq m_d$ and $x_i = k_d$ if $i > m_d$. Clearly $x \in S_d$ and any element $y$ in the cube order such that $y < x$ is in $S_d$. Since there are $|S_d|$ elements which are less than or equal to $x$ in cube order, $S_d$ is an initial segment of the cube order. We may assume that $C_{d-1}(n_{d-1}) \neq \emptyset$. Otherwise $C_d(n_d) = S_d$ is an initial segment of the cube order. Let $y$ be the element in the cube order such that $y > x$ and there is no other element between $x$ and $y$. Then $y_i = 0$ for $i \neq m_d + 1$, $y_{m_d+1} = k_d$. Clearly $y$ is the lowest element in $P(C_{d-1}(n_{d-1}))$ in the cube order. Our assumption that $C_{d-1}(n_{d-1})$ is an initial segment of the cube order guarantees that $P(C_{d-1}(n_{d-1}))$ forms a complete segment of the cube order starting from $y$. Therefore $C_d(n_d)$ is an initial segment of the cube order. By Proposition 1, we obtain the result.

Theorem 1 Let $C$ be a $d$-dimensional grid graph with $n$ vertices and $n_d$ be a positive integer such that $n_d$ divides $n$. Then the total surface area of an equi-partition of $C$ into subgraphs of cardinality $n_d$ is at least

$$\frac{2n}{n_d} \sum_{r=1}^{d} [(r - m_r)(k_r + 1)^{m_r}k_r^{-m_r-1} + m_r(k_r + 1)^{m_r-1}k_r^{-m_r}]$$

where $k_i$ and $m_i$, $i = 1, 2, \ldots, d$ are calculated as mentioned in the definition of the quasi-cube.

Proof: Since the equi-partition has $n/n_d$ grid graphs and each graph is having $n_d$ vertices, the result follows from Lemma 1 and Lemma 2.

4 Equi-partitioning Algorithm

Let $K_1, K_2, \ldots, K_d, V$ be positive integers and $V$ divides $K_1K_2\ldots K_d$. Here we present a heuristic algorithm for $MSA([K_1] \times [K_2] \times \ldots \times [K_d], V)$ problem. The key idea of the algorithm is to decompose the given hyper-rectangle into a collection of well chosen “towers” of the form $[k_1] \times [k_2] \times \ldots \times [k_{d-1}] \times [K_d]$, in which the tower “bases” $[k_1] \times [k_2] \times \ldots \times [k_{d-1}]$ tile the $[K_1] \times [K_2] \times \ldots \times [K_{d-1}]$ base of the given hyper-rectangle. Associated with each tower is the area of a
particular equi-partition of the tower. In theory, one could construct an “optimal” tiling of the given base by minimizing the total of the surface areas of the corresponding towers, but this problem is comparable in difficulty to the original problem. Our heuristic builds up the base tiling via easily computed partitions of lower dimensions. The main task of the algorithm involves solving 1 + \( K_1 + K_1K_2 + \ldots + K_1\ldots K_{d-2}, d \geq 2 \) integer knapsack problems that represent total areas associated with “tilings” by towers. For \( d = 2 \), there is only one knapsack problem and this case was considered by Martin [8]. The algorithm is implemented in \( d + 1 \) phases.

In Phase-0, we partition each hyper-rectangle, \([k_1] \times [k_2] \times \ldots \times [k_{d-1}] \times [K_d]\) where \( k_i \in [K_i] \ (1 \leq i \leq d - 1)\), into components of size \( V \) and calculate the total surface area for the partition. If \( V < k_1k_2\ldots k_{d-1} \) (in which case a component created by the algorithm would be “badly shaped” by having size 1 in dimension \( d \)) or \( V \) does not divide \( k_1k_2\ldots k_{d-1}K_d \), we will consider the component as an invalid component and assign \( \infty \) as the surface area. Otherwise we calculate the area of the partition of the tower assuming the vertices of each component are assigned by the lexicographical order on \( Z^d \) which can be described as follows. Let \( x = (x_1, x_2, \ldots, x_d) \) and \( y = (y_1, y_2, \ldots, y_d) \) be two \( d \)-tuples in \([k_1] \times [k_2] \times \ldots \times [k_{d-1}] \times [K_d]\). We say \( x > y \) if \( j = \max \{i : x_i - y_i \neq 0\} \) and \( x_j - y_j > 0 \). If \( x > y \) then we assign \( y \) before \( x \). If the component size is \( V \), we may assign the first \( V \) cells in the above total order to the first component and the next \( V \) cells to the second component and carry on the process until we fill the targeted region. In Figure 1, the component “A” represents a \([1] \times [1] \times [5]\) tower with an area equal to 22. The components “V”, “W”, “X,” and “Y” represent a \([2] \times [2] \times [5]\) tower with a total area equal to 80.

For a “tall” tower whose base is appropriately chosen, lexicographic assignment generates components that are good approximations to quasi-cubes and hence near-optimal surface areas are obtained. This property is addressed in the following two theorems. Theorem 2 presents an expression that may be used to compute the total surface area resulting from lexicographic assignment of the nodes in a tower under the assumptions that the tower has an integral number of components and that each component has a “height” of at least one. Theorem 3 establishes under some mild assumptions that the average surface area of the components produced by lexicographic assignment of nodes in “tall” towers with “well-chosen” bases asymptotically approaches minimum surface area. The computational results presented in Section 5 demonstrate this asymptotic optimality property of lexicographic assignment.

**Theorem 2** (Total area for lexicographic assignment) Assume that \( V \) divides \( k_1k_2\ldots k_{d-1}K_d, \ V \geq k_1k_2\ldots k_{d-1}, \) and let \( p \geq 1 \) be the number of components of cardinality \( V \) in the \( d \)-dimensional region \([k_1] \times [k_2] \times \ldots \times [k_{d-1}] \times [K_d]\). Let \( s_j = k_1k_2\ldots k_j \) where \( 1 \leq j \leq d - 1 \). Let \( r_{ij} \) be the \( j \)-dimensional remainder of the \( i \)th component which is calculated by \( r_{ij} = iV \mod s_j, \) where \( 1 \leq i \leq p - 1 \), and \( 1 \leq j \leq d - 1 \). Then the total surface area of an equi-partition of \([k_1] \times [k_2] \times \ldots \times [k_{d-1}] \times [K_d]\) into \( p \) components of cardinality \( V \) using the
the average surface area of some constant \( z > 1 \).

Consider the family of \( \text{Theorem 3} \).

Adding the outer and inner boundary areas gives us our result.

The total surface area of inner boundaries is equal to \( 2(N-1+a_{ij}) \) for each \( 1 \leq i \leq N \) and \( 1 \leq j \leq d \). So the total area of the grid graph \( G \) is \( 2(N-1+a_{ij}) \). when \( i = 1 \) and \( j = d \) since \( a_{ij} = 1 \). When \( j > 1 \),

\[
a_{ij} = \begin{cases} 
1 & \text{if } r_{ij} \neq 0, \\
0 & \text{otherwise.} 
\end{cases}
\]

When \( j > 1 \),

\[
a_{ij} = \begin{cases} 
0 & \text{if } k_j = 1, \\
r_{ij} & \text{if } k_j > 1 \text{ and } r_{ij} < s_{j-1}, \\
 s_j - r_{ij} & \text{if } k_j > 1 \text{ and } s_j - r_{ij} < s_{j-1}, \\
s_{j-1} & \text{otherwise.}
\end{cases}
\]

**Proof:** It is clear that the outer surface area of the grid graph \([k_1] \times [k_2] \times \ldots \times [k_d] \) is \( 2N-1+a_{ij} \). To find the area of the inner boundaries, we need to add the area between the 1st component and the \( (i+1) \)th component for each \( 1 \leq i \leq p-1 \). Since \( V \geq s_{d-1} \), there are \( p-1 \) inner boundaries and each boundary is shared by exactly 2 components. So the total area of the projections of the inner boundaries of the components in the \( d \)th direction is \( 2(p-1)s_{d-1} \). For \( 1 \leq j \leq d-1 \), let \( a_{ij} \) be the common boundary area between \( i \)th component and the \((i+1) \)th component in the \( j \)th direction. The possible cases for \( a_{ij} \) are mentioned in the statement of \( \text{Theorem 2} \). Here we explain two cases and leave the more straightforward cases for the reader’s verification. We can visualize \([k_1] \times [k_2] \times \ldots \times [k_j] \) as \( k_j \) cross sections of \([k_1] \times [k_2] \times \ldots \times [k_{j-1}] \) in the \( j \)th direction. These cross sections are assigned to the \( i \)th component in increasing order of \( j \). When \( r_{ij} < s_{j-1} \), the \( i \)th component is assigned \( r_{ij} \) vertices from the first \([k_1] \times [k_2] \times \ldots \times [k_{j-1}] \) cross section in the \( j \)th direction, and hence \( a_{ij} = r_{ij} \). When \( s_{j} - r_{ij} < s_{j-1} \), the \( i \)th component is assigned \( r_{ij} \) vertices from the last \([k_1] \times [k_2] \times \ldots \times [k_{j-1}] \) cross section in the \( j \)th direction, and hence \( a_{ij} = s_{j} - r_{ij} \). Summing up the contributions in each direction, we get the area between the \( i \)th component and the \((i+1) \)th component as \( 2 \sum_{j=1}^{d-1} a_{ij} \). So the total surface area of inner boundaries is equal to \( 2(p-1)s_{d-1} + 2 \sum_{j=1}^{d-1} \sum_{i=1}^{d-1} a_{ij} \).

Adding the outer and inner boundary areas gives us our result.

**Theorem 3** Consider the family of \( d \)-dimensional towers \( T(N) \) of the form \([k_1(N)] \times [k_2(N)] \times \ldots \times [k_{d-1}(N)] \times [K_d(N)], \) where \( |k_d(N) - N| \leq C \) for some constant \( C \) for \( 1 \leq i \leq d-1 \) and all \( N = 1, 2, \ldots \) and \( K_d(N) = N^z \) where \( z > 1 \). Let \( V = N^d \) and let \( a_1(N^d) \) be the average surface area of the components of \( T(N) \) generated lexicographically under the assumptions of \( \text{Theorem 2} \). Then the average surface area \( a_1(N^d) \) asymptotically approaches the minimum surface area (for volumes of size \( N^d \)), i.e.,

\[
\lim_{N \to \infty} \frac{a_1(N^d)}{a^*(N^d)} = 1
\]
where $a^*(N^d) = 2dN^{d-1}$ is the minimum surface area for volume $V$.

**Proof:** By Theorem 2, $a_l(N^d) = 2dN^{d-1} + \sum_{j=1}^{d-1} s_{d-1}(N) K_d(N) + \sum_{i=1}^{p(N)} 1 \sum_{j=1}^{d-1} a_{ij}(N)$ where $s_{d-1}(N) = k_1(N)k_2(N)\ldots k_{d-1}(N)$ and $p(N) = s_{d-1}(N) \times N^2/N^d$. Since $z > 1$, note that $p(N) \to \infty$ when $N \to \infty$. For sufficiently large $N$, it is easily verified that $a_l(N^d) \leq 2(N+C)^{d-1} + 2 \sum_{j=1}^{d-1} \frac{N^d}{N^d} + d(N+C)^{d-2}$. Thus $\lim_{N \to \infty} \frac{a_l(N^d)}{a^*(N^d)} \leq 1$, but since $a_l(N^d) / a^*(N^d) \geq 1$ for all $N$, the conclusion follows.

4.1 Phase-0 (Surface area computation for towers)

Surface area computations using formula (1) are performed for all “valid” towers.

{Phase-0 }

Define a $(d-1)$-dimensional array $A_{d-1}[K_1][K_2] \ldots [K_{d-1}]$ of surface areas as follows:

$V \leftarrow$ the cardinality of a component.

for $k_1 = 1$ to $K_1$
  for $k_2 = 1$ to $K_2$
    : 
    for $k_{d-1} = 1$ to $K_{d-1}$
      begin 
      if $V$ does not divide $k_1k_2\ldots k_{d-1}K_d$ or $V < k_1k_2\ldots k_{d-1}$
        $A_{d-1}[k_1][k_2] \ldots [k_{d-1}] \leftarrow$ INFINITY
      else
        $A_{d-1}[k_1][k_2] \ldots [k_{d-1}] \leftarrow$ The total surface area given by (1) for the partition of $k_1 \times k_2 \times \ldots \times k_{d-1}K_d$ into components of cardinality $V$ using the lexicographical order
      end if 
    end for 
  end for 
end for 

\end{Phase-0}.

4.2 Phase-$i$ ($1 \leq i \leq d-1$) (Determine optimal base decompositions in successively higher dimensions)

In the $i$th ($1 \leq i \leq d-1$) phase, we solve $K_0K_1K_2\ldots K_{d-1-i}$ (where $K_0 = 1$) integer knapsack problems using dynamic programming. For each $k_i \in [K_i]$, We define the $k_1k_2\ldots k_{d-1-i}$th integer knapsack problem as follows.
minimize $\sum_{j=1}^{K_{d-i}} c_j x_j$

subject to $\sum_{j=1}^{K_{d-i}} h_j x_j = K_{d-i}$

where each $x_j$ is a nonnegative integer variable, $h_j = j$, and $c_j = A_{d-i}[k_1][k_2] \ldots [k_{d-i-1}][j]$ (where $A_{d-i}$ is a $(d-i)$-dimensional matrix (i.e., $i = 1$ case is used in Phase-0)) contains the optimal objective value of the $(k_1 k_2 \ldots k_{d-i-1})$th integer knapsack problem when $i > 1$, and the area of the equi-partition of subdomain $[k_1] \times [k_2] \times \ldots \times [k_{d-i-1}] \times [j] \times [K_d]$ when $i=1$ (from Phase-0).

The above integer knapsack problem determines the combination of hyper-rectangular subdomain heights in direction $(d-i)$ that minimizes the total surface area as expressed by the objective function. The value of $x_j$ represents how many subdomains of height $j$ in the $(d-i)$ direction (i.e., $[k_1] \times [k_2] \times \ldots \times [k_{d-i-1}] \times [j] \times [K_d]$) are in the optimal solution. The reader can observe that Phase-$i$ knapsack problems use the results of Phase-$(i-1)$ knapsack problems as their area measure objective coefficients $c_j$. Initially, Phase-0 area values are used as $c_j$ values for Phase-1. In Figure 1, the towers $\{A\}$, $\{B, C\}$, and $\{D, E\}$ represent a solution to a Phase-1 knapsack problem which partitions a $[2] \times [1] \times [5]$ hyper-rectangle. These Phase-1 solutions have produced the final Phase-2 solution.

begin { Phase-$i$ ($1 \leq i \leq d - 1$) }
  if ($i = d-1$) (i.e. the final knapsack problem)
    \begin{verbatim}
    A_0 ← The solution of the final knapsack problem (i.e. the optimal
    objective value and a corresponding solution \{(h_j, x_j)|x_j \neq 0\})
    call Phase-$d$
    return
    \end{verbatim}
  else
    Define a $d - i - 1$ dimensional array $A_{d-i-1}[K_1][K_2] \ldots [K_{d-i-1}]$.
    for $k_1 = 1$ to $K_1$
      for $k_2 = 1$ to $K_2$
        \ldots
        for $k_{d-i-1} = 1$ to $K_{d-i-1}$
          begin
            $A_{d-i-1}[k_1][k_2] \ldots [k_{d-i-1}] ←$ The solution of the $(k_1 k_2 \ldots k_{d-i-1})$th
            Knapsack problem (i.e. the optimal objective value and a
            corresponding solution \{(h_j, x_j)|x_j \neq 0\})
          end
        \end{verbatim}
      \end{verbatim}
    \end{verbatim}
  end
end { Phase-$i$ }
Our equi-partitioning algorithm is feasible when the above Phase-\((d-1)\) algorithm which solves the last integer knapsack problem is feasible. The following theorem guarantees the feasibility of Phase-\((d-1)\) integer knapsack problem.

**Theorem 4** If \(V\) divides \(K_1K_2\ldots K_d\) then Phase-\((d-1)\) integer knapsack problem is feasible.

**Proof:** Since \(V\) divides \(K_1K_2\ldots K_d\), we can find positive integers \(p_i\) \((1 \leq i \leq d)\) such that \(p_i\) divides \(K_i\) and \(V = p_1p_2\ldots p_d\) (i.e., use the fundamental theorem of algebra). Now we can write \([K_i] = \bigcup_{j=0}^{n_i}(jp_i + [p_i])\), where \(n_i = (K_i/p_i) - 1\). We consider the \((p_1p_2\ldots p_{d-1})\)th knapsack problem in Phase-\(i\) \((i < d - 1)\) where \(c_j = A_{d-i}[p_1][p_2]\ldots[p_{d-1-1}][j]\). This problem has a feasible solution \(x_{p_{d-1}} = n_{d-1}\), and \(x_j = 0\) if \(j \neq p_{d-1}\) with the objective value \(c_{p_{d-1}}n_{d-i}\) if \(c_{p_{d-1}}\) is finite. Now we show that \(c_{p_{d-1}} = A_{d-i}[p_1][p_2]\ldots[p_{d-1-1}][p_{d-1}]\) is finite for each \(i\) \((1 \leq i \leq d - 1)\) by induction on \(i\). In our base case \((i = 1)\), since \(V\) divides \(p_1p_2\ldots p_{d-1}K_d\) and \(p_1p_2\ldots p_{d-1} \leq V\), we get a finite value for \(c_{p_{d-1}} = A_{d-1}[p_1][p_2]\ldots[p_{d-1}][d]\) (from Phase-\(0\)) which is the total surface area of the partition of the hyper-rectangle \([p_1] \times [p_2] \times \ldots [p_{d-1}] \times K_d\). We assume \(c_{p_{d-1}}\) is finite for \((i < d - 1)\). Then \((p_1p_2\ldots p_{d-1-1})\)th knapsack problem in Phase-\(i\) \((i < d - 1)\) is feasible as mentioned before and \(c_{p_{d-1}}\) gets its finite optimal objective value. Therefore, by induction, \(c_{p_{d-1}}\) is finite for all \(i\) \((1 \leq i \leq d - 1)\). Since \(c_{p_{d-1}}\) is finite, Phase-\((d-1)\) knapsack problem is feasible with a feasible solution of \(x_{p_1} = n_1\), and \(x_j = 0\) if \(j \neq p_1\) with the objective value \(c_{p_1}n_1\). \(\square\)

### 4.3 Phase-\(d\) (Assignment)

Let \(x = (x_1, x_2, \ldots, x_n)\) and \(h = (h_1, h_2, \ldots, h_n)\) represent the solution of a knapsack problem that partitions the \(j\)th direction such that \(\sum_{i=1}^{n_j} h_ix_i = K_j\) and each \(x_i > 0\). Now we can partition \([K_j]\) such that \([K_j] = \bigcup_{i=1}^{n_j}(k_{ji} + [H_{ji}])\)

where \(n_j = \sum_{i=1}^{n_j} x_i, k_{ji} = \sum_{p=1}^{q-1} x_p h_p + (i-1)h_j\) and \(H_{ji} = h_j\) if \(i\) satisfies \(1 + \sum_{p=1}^{q-1} x_i \leq i \leq \sum_{p=1}^{q} x_i\).

begin Phase-\(d\)

count ← 0

proc ← 1

for \(i_d = 0\) to \(K_d - 1\)

begin

Select the solution \((x, h)\) stored in \(A_0\) from Phase-\((d-1)\) which partitions 
\([K_1]\)

for \(i_1 = 1\) to \(m_1\)

begin

Select the solution \((x, h)\) stored in \(A_1[H_{1i_1}]\) from Phase-\((d-2)\) which partitions 
\([K_2]\)

for \(i_2 = 1\) to \(m_2\)

begin

:
4.4 Analysis of the Equi-partitioning Algorithm

First we show that our algorithm solves the $\text{MSA}([K_{1}] \times [K_{2}] \times \ldots \times [K_{d}], V)$ problem in $O(n) \ (n = K_{1}K_{2} \ldots K_{d})$ time if each $K_{i}$ is in $O(n^{1/d})$. In Phase-0, we do $n/K_{d}$ assignments and each assigned value takes at most $O(n^{1/d})$ time to calculate (Theorem 2). Since $K_{d} = O(n^{1/d})$, we need $O(n)$ time for Phase-0. In Phase-$i$ ($0 \leq i \leq d - 1$), we solve $K_{0}K_{1} \ldots K_{d-i-1}$, ($K_{0} = 1$), integer knapsack problems using dynamic programming which takes $O(K_{d-i}^{2/d}) = O(n^{2/d})$ time for each problem [12]. Hence we solve Phase-$i$ in $O(n^{(d-i+1)/d})$ time with the worst case $i = 1$ giving $O(n)$. It is clear that the final assignment phase (Phase-$d$) also takes $O(n)$ time. Therefore the algorithm runs in $O(n)$ time when each $K_{i}$ is $O(n^{1/d})$.

Let us define a gap of a feasible solution for the $\text{MSA}$ problem as the difference between a lower bound and an upper bound for the total area of
the partition. The percent gap (gap%) is the gap as a percentage of the given lower bound. Although the above algorithm is always feasible when \( V \) divides \( K_1K_2\ldots K_d \), there are examples where the algorithm may not perform well (i.e., large gap%). For example, consider \( \text{MSA}(\lceil P \rceil^4, P^2) \) problem where \( P \) is a prime. In this problem every valid domain in Phase-0, \( [k_1] \times [k_2] \times [k_3] \times [P] \), has exactly one of \( k_1, k_2 \) or \( k_3 \) equal to \( P \). So the area of each component is \( \Omega(P^4/3) \). The lower bound given in Theorem 3.2 for the area of a component is \( O(P) \). Hence gap% is \( \Omega(P^{1/3}) \) which is undesirable for large \( P \). We can modify Phase-0 and Phase-\( d \) of our algorithm to avoid these undesirable gaps. In Phase-0, when \( V \) does not divide \( k_1k_2\ldots k_{d-1}K_d \) (i.e., partition contains a partial component), the approach of Theorem 2 can be used to obtain an upper bound for the area of the partition. Instead of assigning \( \infty \) to \( A_{d-1}[k_1][k_2]\ldots[k_{d-1}] \), we assign \( U + 2s_{d-1} \) where \( U \) is the total area expression in Theorem 2 assuming \( p = [k_1k_2\ldots k_{d-1}K_d/V] \). In the final phase, Phase-\( d \), we first assign labels for the full components and then work with the remaining partial components. We get at most one inner boundary inside a partial component justifying our upper bound \( U + 2s_{d-1} \) in Phase-0.

5 Computational Results

The algorithms were coded in C and ran in 3.0GHz PCs with Linux OS.

We present computational results for selected three and four dimensional problems. Figure 2 shows the results for \( \text{MSA}([M]^3, M) \) (1 \( \leq M \leq 1000 \)). The percentage of solutions that are at the lower bound (i.e., optimal solutions) is 10.3%, and within 4% of gap is 95.3%. Figure 3 shows the execution times for the cases given in Figure 2. Due to memory limitation for larger problems, we have excluded the final assignment phase. So the running times in the graph are for obtaining the solutions without actual assignments.

Figure 4 shows the results for \( \text{MSA}([M]^4, M^2) \) (1 \( \leq M \leq 400 \)). The percentage of solutions that are at the lower bound (i.e., optimal solutions) is 15.0%, and within 4% of gap is 77.75%. Here we used the modified algorithm as mentioned in Section 4.4. Figure 5 shows the execution times for the cases given in Figure 4.

6 Conclusions and Future Research

We have developed a heuristic that runs in linear time in the size of the graph (for well shaped hyper-rectangles) that generates asymptotically optimal equi-partitions of hyper-rectangles, and demonstrated this performance for graphs of size up to \( 10^{10} \).

When the area of an equi-partition does not meet the lower bound, it is very challenging to decide whether the partition is optimal. As an example, one may consider the equi-partition for the \( \text{MSA}(\lceil 5 \rceil^3, 5) \) problem shown in Figure 1 which is the smallest non trivial problem in this family. This partition has a surface
Figure 2: Gap% versus the number of vertices \((M^3)\) for \(MSA([M]^3, M)\) \((1 \leq M \leq 1000)\)

Figure 3: Execution time versus the number of vertices \((M^3)\) for \(MSA([M]^3, M)\) \((1 \leq M \leq 1000)\)
Figure 4: Gap\% versus the number of vertices ($M^4$) for $MSA([M]^4,M^2)$ ($1 \leq M \leq 400$)

Figure 5: Execution time versus the number of vertices ($M^3$) for $MSA([M]^4,M^2)$ ($1 \leq M \leq 400$)
area of 502, bound gap of 2, and with only one non-optimal component which is shown with the letter ‘A’. Is this an optimal solution? Constraint programming may allow optimality verification for smaller problems such as this.

Other interesting extensions to this research include relaxing the divisibility assumptions (i.e., modified algorithm), and improving the assignments obtained by more complex procedures for bad components as suggested in [5] for the two-dimensional case.

**Acknowledgments**

The authors thank Winston Yang for bringing the references [1] and [3] to their attention.
References

[1] L. Alanzo and R. Cerf. The three dimensional polyominoes of minimal area. *Electronic Journal of Combinatorics*, 3(1):1–39, 1996.

[2] S. L. Bezrukov and B. Rovan. On partitioning grids into equal parts. *Computers and Artif. Intell.*, 16:153–165, 1997.

[3] B. Bollabas and M. Leader. Edge isometric inequalities in the grid. *Combinatorica*, 11(1):299–314, 1991.

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

[5] W. W. Donaldson. *Grid-graph partitioning*. Ph.D. Thesis, Department of Computer Science, University of Wisconsin-Madison, Madison, WI, 2000.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[7] S. Ghandeharizadeh, R. R. Meyer, G. Schultz, and J. Yackel. Optimal balanced partitions and a parallel database application. *ORSA Journal on Computing*, 4:151–167, 1993.

[8] W. Martin. Fast equi-partitioning of rectangular domains using stripe decomposition. *Discrete Appl. Math.*, 82:193–207, 1998.

[9] E. J. Neves. A discrete variational problem related to ising droplets at low temperatures. *Journal of Statistical Physics*, 80:103–123, 1995.

[10] R. J. Schalkoff. *Digital Image Processing and Computer Vision*. John Wiley & Sons, Inc., 1989.

[11] J. Strikwerda. *Finite Difference Schemes and Partial Difference Equations*. Wadsworth & Brooks, 1989.

[12] L. A. Wolsey. *Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Inc., 1998.

[13] J. Yackel. Partitioning grid graphs with partial cubes. *private communication*.

[14] J. Yackel. *Minimum-perimeter tiling for optimization of parallel computation*. Ph.D. Thesis, Department of Computer Science, University of Wisconsin-Madison, Madison, WI, 1993.

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