Dynamic Structured Grid Hierarchy Partitioners Using Inverse Space-Filling Curves[*]

Johan Steensland

6th February 2001

Abstract

This paper discusses partitioning of dynamic structured grid hierarchies, occuring in structured adaptive mesh refinement (SAMR) applications. When a SAMR method is executed on a parallel computer, the work load will change dynamically. Thus, there is need for dynamic load balancing. Inverse space-filling curve partitioning (ISP) is appealing for load balancing in parallel SAMR, because of its speed. In this paper, ISP is considered as *part* of a partitioning approach, which combines structured and unstructured techniques. More precisely, various design decisions for the structured partitioning are considered. Different design choices lead to graphs with different properties. The main objective is to investigate how these differences affect the behavior of ISP. The paper contributes by (a) identifying certain design choices as being advantageous, and (b) presenting four new partitioning algorithms that correspond to these design decisions.

Keywords: Space-filling curves; Parallel/distributed algorithms; Dynamic partitioning/load-balancing; SAMR.

1 Introduction

This paper discusses partitioning of dynamic structured grid hierarchies. Such hierarchies occur in the context of structured adaptive mesh refinement (SAMR). When SAMR is executed on a parallel computer, the work load will change dynamically, as new grid points are inserted or removed. Thus, there is need for dynamic load balancing. This can be achieved by dynamically repartitioning and remapping the grid hierarchy. For this to be a viable approach, the repartitioning algorithm has to be very fast. For this reason, inverse space-filling curve partitioning (ISP) is appealing for load balancing in parallel SAMR. There are

[*]This research was supported by the Swedish Foundation for Strategic Research via the program of Industrial Computational Mathematics.

several reports of work where ISP was used in that context with satisfactory results [1, 2, 3, 4].

In this paper, ISP is considered as *part* of a partitioning approach, which combines structured and unstructured techniques. First, the grid hierarchy is partitioned with structured techniques, i.e., techniques that cut the hierarchy into rectangular blocks. The resulting block graph can then be partitioned by means of a general graph partitioning algorithm. Consequently, each partition will be a collection of blocks.

Now, assume that we take this approach and use ISP for the partitioning of the block graph. How, then, should the initial *structured* partitioning be carried out in order to yield a block graph with properties suitable from the point of view of ISP? This is the question to be addressed here.

More precisely, we consider various design decisions for the structured partitioning. Different design choices lead to graphs with different properties. The main objective is to investigate how these differences affect the behavior of ISP. As a result, we identify certain design choices as being advantageous. Finally, we present four new partitioning algorithms that correspond to these design decisions.

2 Partitioning SAMR Applications

In Section 2.1, we give a brief introduction to SAMR and dynamic grid hierarchies. In Section 2.2, we classify existing partitioners, including the algorithms presented in this paper, for SAMR applications. Section 2.3 discusses design choices for partitioners, and their implications on the properties of the resulting block graph.

2.1 Introduction to SAMR

The numerical solution to a partial differential equation (PDE), is obtained by computing an approximate solution for discrete points. A standard way of discretizing the domain is to introduce a structured uniform grid. The unknowns of the PDE are then computed at each discrete grid point.

The resolution of the grid determines local and global error. Moreover, it dictates memory requirement and number of arithmetical operations required to obtain the solution.

For simulations where the solution to the PDE is ”well behaved”, it might be possible to find a uniform resolution of the grid that is satisfactory with respect both to solution accuracy and computing resources. However, in cases where the solution contains shocks, discontinuities, or steep gradients, it might be impossible to meet demands both on solution accuracy and computing resources with a uniform grid. Moreover, due to solution dynamics in time-dependent problems, it is difficult to estimate minimum grid resolution to obtain acceptable solution accuracy.

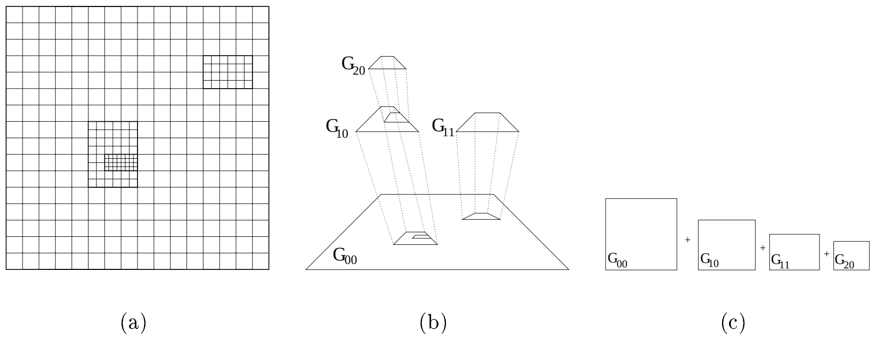


Figure 1: Three views of a structured adaptive mesh refinement (SAMR) example. Sub figure (a) shows the grid, sub figure (b) the Berger dynamic grid hierarchy, and sub figure (c) shows a linear representation of all patches involved in the computation. The sub indices are refinement level and patch ID, respectively.

Adaptive mesh refinement (AMR) techniques provide a means for concentrating additional resolution and computational effort to demanding regions in the computational domain. In the case of structured grids, refinement proceeds recursively so that regions requiring higher resolution are flagged, and finer-resolution grids are overlaid on these regions, forming a dynamic grid hierarchy (Figure 1). This is called structured AMR (SAMR).

2.2 Classification of SAMR Partitioners

A dynamic grid hierarchy [5] stemming from a SAMR application, can be partitioned in two fundamentally different ways.

1. **Patch-based partitioning** [6, 7]. Distribution decisions are made independently for each newly created patch. A patch may be kept on the local processor or entirely moved to another processor. If the patch is considered too large, it may be split. Figure 2.

2. **Domain-based partitioning** [5, 8]. Domain-based partitioners partition the physical domain, rather than the grids themselves. The domain is partitioned along with all contained grids on all refinement levels. Figure 3-4.

Unstructured partitioning techniques based on graphs as input, could be used for both approaches. The dynamic grid hierarchy in Figure 1 b could be viewed

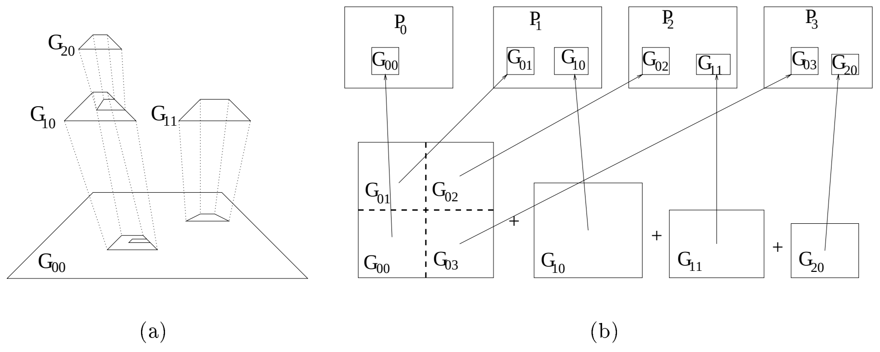


Figure 2: In patch-based partitioning, distribution decisions are made independently for each newly created patch. A patch may be kept on the local processor or entirely moved to another processor. If the patch is considered too large, it may be split. An example of such a patch is G_{00} in sub figure (a), which is split into $G_{00} - G_{03}$ in sub figure (b).

as a graph, where the grid patches are the vertices and the communication requirements between grids constitute the edges. For example, iterative tree-balancing [5] may be applied to this graph, hence forming a patch-based method.

Another way of creating a graph from the dynamic grid hierarchy is the following. Let each node in the graph correspond to a computational point in the coarsest grid, and let the edges indicate communication, hence forming a *grid graph*. This can be implemented as a matrix with entries corresponding to workload.

The values in the matrix are set to reflect the cost, i.e., the computational effort, associated to advancing the solution one time-step *on the coarsest level* of the grid. Thus, the workloads of all overlaid, refined grids are projected on, and accumulated in, the coarse grid points and their corresponding matrix entries.

The algorithms presented in Section 4 all use the approach discussed in Section 1. They create a block graph by attacking the grid graph. Hence, they clearly fall into the category of domain-based techniques. They partition the matrix of workloads into blocks, thus forming a block graph. This graph is mapped onto the processors by use of a space-filling curve.

There are three main advantages for the domain-based approach. First, there is the elimination of "depth-wise" communication. Overlaid grids belonging to different refinement levels have to communicate. The communication is of restriction and prolongation type. If all cuts by the partitioner are orthogonal to the dimension of the expanding and contracting dynamic grid hierarchy (i.e., domain-based partitioning), then this communication is eliminated. This tends to increase scalability. Second, partitioning the workload matrix, instead of

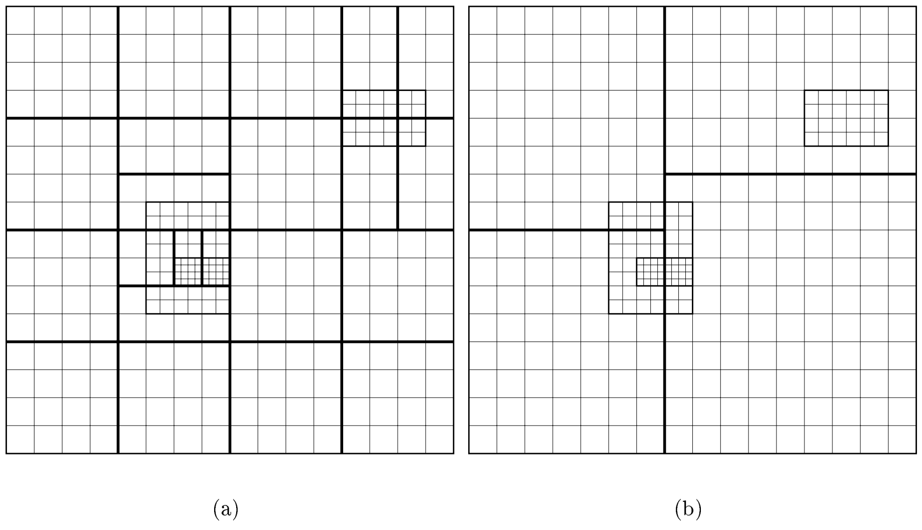


Figure 3: Resulting block graphs from two domain-based partitioning strategies, illustrating different design decisions as described in Section 2.3 and summarized in Table 1. Both strategies can produce both heterogeneous and homogeneous blocks. However, the relations between the blocks are regular in (a), but irregular in (b).

explicitly attacking the dynamic grid hierarchy graph, is normally a less complex task. The matrix is a complexity-static two-dimensional entity, while the complexity of the grid hierarchy is dynamic. Hence, given an efficient mapping scheme that maps the grid hierarchy onto the matrix, the complexity of the partitioning of the matrix does not depend on the complexity of the grid hierarchy. (The same idea is explored e.g., in PLUM [9], a software infrastructure for unstructured AMR, where a dual-graph corresponding to the coarsest mesh is partitioned). Third, in parallel SAMR, execution is performed level by level in a sequential manner. That is, the solution on all grids at refinement level l have to be advanced before the solution on any overlaid grid at level $l+1$ can be advanced. Thus, for patch-based partitioning where processors may host grids belonging to a certain refinement level solely, these processors will be idle while execution is on-going on grids at other levels.

Hybrid partitioners combine the two approaches described above. Partitioning is performed in two steps. The first step uses domain-based techniques to generate meta-partitions that are mapped to a group of processors. The second step uses a combination of domain and patch based techniques to optimize the distribution of each meta-partition within its processor group. For a discussion

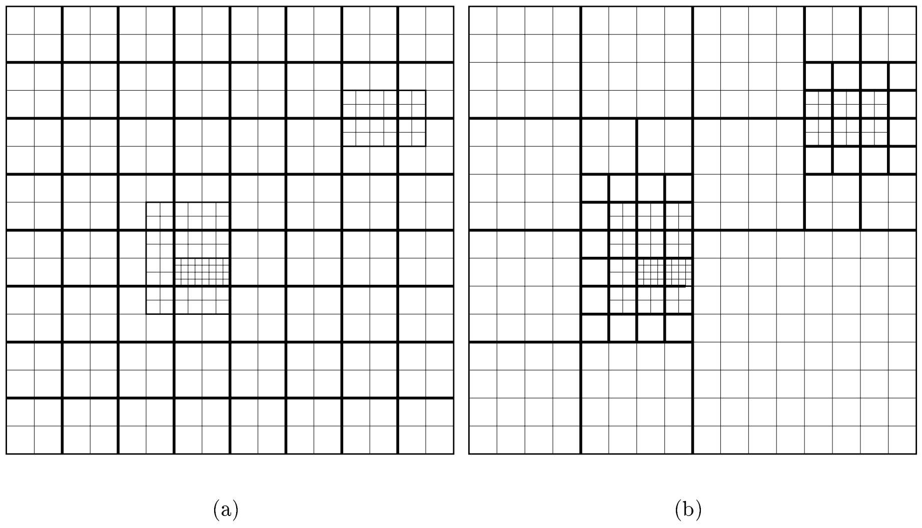


Figure 4: Resulting block graphs from two more domain-based partitioning strategies, illustrating different design decisions as described in Section 2.3 and summarized in Table 1. Sub figure (a) illustrates a strategy that produces blocks with equal size and hence regular relations, most likely both homogeneous and heterogeneous. Sub figure (b) illustrates a strategy that has the requirement to obtain homogeneous blocks exclusively. The relations between the blocks are regular, but the sizes may vary drastically.

on strategies for composite grids, see [10].

2.3 Design Decisions and Implications on the Block Graph

We now focus on the interrelation between the two phases of our partitioning approach. As illustrated in Figures 3-4, different structured partitioning algorithms lead to block graphs with different properties. The main objective of the present paper is to discuss how well-suited these block graphs are for being partitioned with ISP. Will some design choices give graphs that are more "ISP-friendly" than others? Before we give this question a more precise formulation, we will briefly mention some important design decisions, and their impact on the block graph.

The two main design choices for a domain-based *structured* partitioner are (a) where to place the cuts, and (b) those affecting granularity. Design decisions for (a) include (but are not limited to) (1) cuts at the geometrical mid-point, (2) cuts at the optimal point with respect to load balance, and (3) cuts aligned to refined patches for creating blocks homogeneous in terms of refinement level. That is, the entire block has uniform resolution. Design decisions for (b) include

(1) a pre-defined fixed number of blocks, (2) an approximate granularity, and (3) granularity will be determined by some other objective. As an example of the latter, we might want all blocks to be homogeneous in terms of workload. Then we split the domain (with arbitrary selection of (a)) until we achieve this objective. The objective and the input dynamic grid hierarchy determines the granularity.

Applying a combination of choices for (a) and (b) results in a block graph. In this graph, we can identify two fundamentally different kinds of blocks: *Homogeneous* (HO) and *heterogeneous* (HE) blocks in terms of refinement level in the sense described above. The relations between blocks can be categorized into blocks with *equal size* (ES) and block with *equal workload* (EW). Further, the relations could be *regular* or *irregular* (I/R). The relations between blocks are defined as regular if the domain is always split (recursively) at its geometrical midpoint, and irregular otherwise. The communication pattern for irregular block relations is potentially more complex. Moreover, the relations could be of the type *no restrictions* (NR), where sizes, aspect ratios and workloads are arbitrary.

Table 1 shows some examples of combinations of design choices (a) and (b) and corresponding objectives and block types and block relations. Note that type/relation corresponds to the algorithm *objective.* That is, equal workload in the heterogeneous case, for example, does not necessarily mean that all blocks have *exactly* the same weight. Rather, all blocks are supposed to have roughly the same weight, according to the partitioning objective (and the success of the algorithm).

(a)	(b)	Figure	Objective	Type/Relation
1	2,3	3 (a)	Equal workload, $\approx kp$ blocks	HE: EW, R, trivial
2	1	3 (b)	p blocks with equal workload	HE/(HO): EW, I
1	1	4 (a)	High granularity	HE/HO: ES, trivial
1	3	4 (b)	Homogeneous blocks	HO: NR, ES, EW, R
3	3	-	Homogeneous, fewer blocks	HO: NR, I

Table 1: Combinations of design choices (a) where to place the cuts, and (b) the desired granularity, and their implications on the block graph. NR=no restriction; EW=equal weight; ES=equal size; R=regular; I=irregular. These terms are explain in Section 2.3.

3 The SAMR Block-Graphs and the Heuristics of ISP

3.1 The problem to be considered

We have now identified a number of block-graph properties, resulting from various design choices for the structured partitioning. The question of interest is: how do these properties relate to ISP?

In short, ISP operates in three steps [3]: (1) *Indexing.* Based on its geometrical position, each vertex is labeled with an index (a real or integer number). The index is determined by a space-filling curve (SFC). (2) *Sorting.* The vertices are sorted with respect to their indices, forming a linear list. (3) *Coloring.* Consecutive portions of the sorted list of vertices are assigned to the processors.

Parashar et al [1] discovered that, given a binary recursive splitting scheme with cuts at the geometric mid-point, step (1) and (2) could be avoided by letting the SFC dictate the orientation of the cuts in the partitioning. We use this technique, and extend it to handle arbitrary binary recursive rectilinear cuts, i.e., cuts not aligned to the geometrical mid-point.

Partitioning a graph with the only objectives *(a)* optimize load balance, and *(b)* minimize edge cut, is an NP-hard problem. As a consequence, real-life partitioners (with even more objectives) rely on heuristics. Typically, they optimize one metric at the expense of others.

The heuristics behind ISP consist of two assumptions:

1. A graph node will communicate more with its geographically nearest neighbors than with nodes far away.

2. An SFC preserves proximity between nodes. That is, if two nodes are geographically close in the graph, then the SFC will give them nearby indices.

For our block graphs, it is clear that Assumption 1 is normally valid in a global sense. However, it is also of interest to consider the local situation, close to partition boundaries. Consider Figure 5. There, we have a reference block A, and two of its nearest neighbor blocks, B and C. According to Assumption 2, the ISP algorithm will tend to map A closer to C than to B, since the geographical distance between A and C is shorter. If Assumption 1 is locally valid, then the volume of communication is larger between A and C than between A and B. Consequently, if it turns out that a partition boundary should be placed between B and C, then it is likely, under these assumptions, that A will be placed in the same partition as C, which would lead to a smaller edge cut.

The focus of our investigation with thus be *the local validity of Assumption 1.*

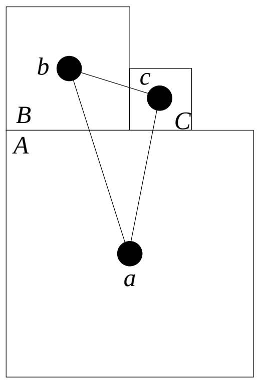


Figure 5: A reference block A, and two of its nearest neighbor blocks, B and C. According to Assumption 2, the ISP algorithm will tend to map A closer to C than to B, since the geographical distance between A and C is shorter.

3.2 The Local Validity of Assumption 1

To investigate the local validity of Assumption 1, we will consider a number of representative scenarios, all following the general pattern of Figure 5. In each scenario, the blocks A, B, and C represent a different set of the properties identified in Section 2.3. We will investigate to what extent Assumption 1 is valid in these scenarios, i.e., whether it is true that if the reference block A is closer to C than to B, then there is more communication between A and C than between A and B. If some scenarios fulfill the assumption better than others, then the corresponding design choices will be beneficial for ISP.

The scenarios differ with respect to the properties identified in Section 2.3. However, there are additional properties, such as the aspect ratio of the blocks, that can vary within each single scenario. Thus, in each study of a specific scenario, we will consider more than one case, the different cases representing different settings of such additional properties.

As a consequence of the multi-case scenario, we will not be able to conclude that a certain scenario entirely fulfills (or does not fulfill) Assumption 1. For a certain scenario, the normal situation will be that the assumption is fulfilled in some cases but not in others. Thus, we introduce a metric for evaluating *to what extent* the scenario, considering all cases together, fulfills the assumption.

A Metric for the Local Validity of Assumption 1

To measure the degree of local validity of Assumption 1 in the scenarios outlined by Figure 1, we study graphs containing only three vertices (corresponding to the three blocks A, B, and C). These “triangle graphs” are intended to constitute cases of the scenario to be studied.

We start by defining the triangle graphs. Let the operation $a \oplus b$ denote $(a+b)$ modulo 3.

Definition 1 *A triangle graph $G = (V, E, W, D)$ is a triangle with vertices $v_j \in V$ in $\mathbb{R}^n$. The edges $e_j = (v_j, v_{j\oplus 1}) \in E$ have weights $w_j \in W$. The Euclidean distance between vertices connected by e_j is $d_j \in D$.*

In order to set up the triangle graph for a certain case, the vertices of the graph are defined as the geometric midpoints of the corresponding blocks. Assuming a refinement factor of 2 and a two-dimensional case, we define edge weight as $s \sum_{r=0}^{l} 4^r$ where s is the length of the common side, and l is the refinement level of the block with the coarsest level bordering the side s. In cases where there is no edge between B and C, this is reflected in the triangle graph by introducing a "fuzzy edge" with weight zero. (As an example, this would be needed in a scenario where the three blocks where taken from a uniform block graph).

Definition 2 *A reference point is a vertex in a triangle graph such that both its incident edges have weights greater than zero.*

We will study the edges incident to the reference point. If the one having the largest weight is also associated with the shortest distance, then Assumption 1 is locally valid for that particular triangle graph. Formally, we define the function *ALVA* (Assumption 1 Locally VAlid) as follows:

Definition 3 *Let V be the set of vertices of a triangle graph, and let $R \subseteq V$ be the set of reference points. The boolean function $ALVA : R \mapsto \{0, 1\}$ returns 1 if Assumption 1 is locally valid taking the input vertex as a point of reference, and 0 otherwise.*

$$ALVA(v_j) = \begin{cases} 1 & \textit{if } Sign(w_j - w_{j\oplus 2}) = Sign(1/d_j - 1/d_{j\oplus 2}) \\ 0 & \textit{otherwise} \end{cases},$$

where the function Sign is extended to return plus, minus, or equal.

We can determine the *degree of local validity of Assumption 1* for a certain scenario, by accumulating the *ALVA* values for the corresponding cases.

Definition 4 *For $1 \leq i \leq n$, let G_i denote a triangle graph. Further, let S_G be the set of all G_i:s, let k_i be the number of reference points in G_i, and let k be the total number of reference points in S_G. Then, the degree Γ of local validity of Assumption 1 for S_G is defined as*

$$\Gamma(S_G) = \frac{1}{k} \sum_{i=1}^{n} \sum_{j=1}^{k_i} ALVA(v_j \in G_i).$$

A value of $\Gamma = 1$ means that Assumption 1 is locally valid for all cases of the scenario. If $\Gamma = 0$, then Assumption 1 is always locally invalid for the scenario. In general, the value of Γ will be somewhere in between those two extremes.

We now proceed to evaluate Γ for a number of representative scenarios.

The Trivial Case

The trivial case is the result of homogeneous blocks with equal size, equal workload, and regular relations, see Figure 6. Applying our metric to this computational graph, we realize trivially that $\Gamma = 1$. That is, short distances translate perfectly into more communication. The same result holds for quadratic blocks. Though a trivial case, the result may lead us to believe that it is possible to establish positive coding even for more complicated grid structures and split operations.

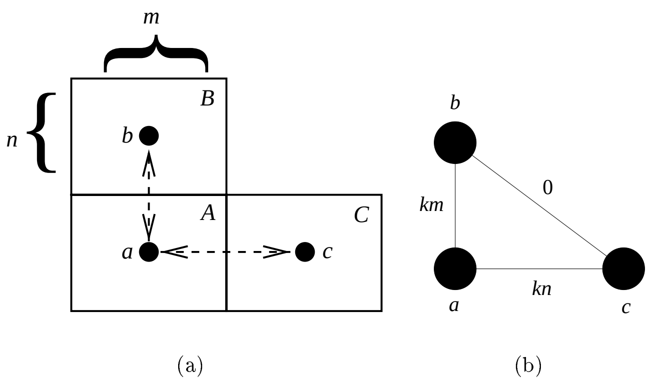
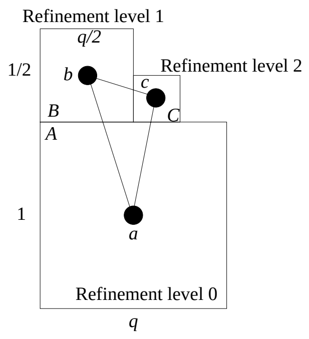


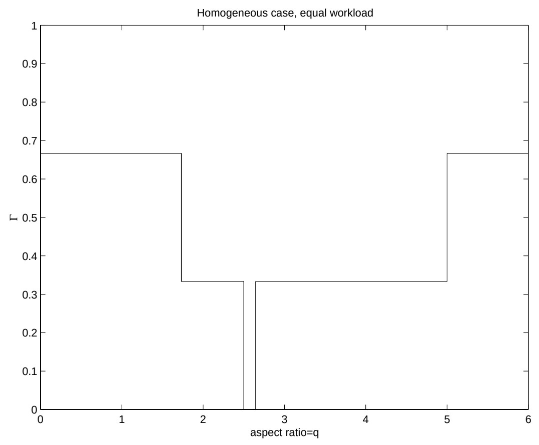
Figure 6: The trivial case (a) and a corresponding triangle graph (b). Here, Assumption 1 is locally valid.. More communication is going on between A and B than between A and C since $m > n$ and communication is proportional to the side length. Moreover, a and b are closer to each other than a and c are.

Scenario 1 (HO EW)

This scenario describes homogeneous blocks with equal workload. Three blocks are positioned according to Figure 7. The values of Γ are derived as a function of the aspect ratio q. Only one triangle graph was used for each q, since other interesting cases are reflections and rotations of the original positions. For a perfect square, $\Gamma = 2/3$, but as the aspect ratio increases, Γ decreases (see Figure 7). Interestingly, $\Gamma = 0$ for $2.5 < q < \sqrt{7}$. After $q = \sqrt{7}$, Γ increases again.



(a) A perfect square. $\Gamma = 2/3$



(b) Γ as a function of q

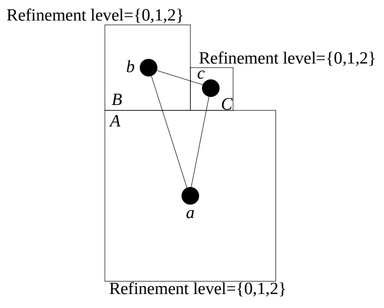
Figure 7: Scenario 1. Homogeneous blocks with equal workload. For a perfect square, $\Gamma = 2/3$, but as aspect ratio increases, Γ decreases. Interestingly, $\Gamma = 0$ for $2.5 < q < \sqrt{7}$. After $\sqrt{7}$, Γ increases again.

Scenario 2 (HO NR)

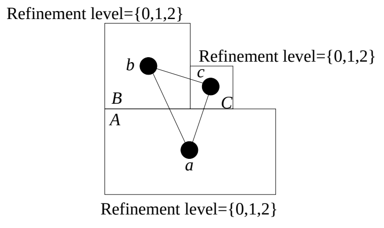
In Scenario 2, we study homogeneous blocks without any correlation between a block's size and its workload. A block can have any workload and any size. We study this case by examining two particular sub cases, namely (a) structured quadratic form, and (b) the largest block is a rectangle, see Figure 8. For both cases, the refinement levels were permuted among the blocks, resulting in 18 triangle graphs for each sub case. For (a) $\Gamma = 1/3$ and for (b) $\Gamma = 2/9$. Interestingly, if all blocks were to assume the same rectangular shape as the biggest block, then Γ goes back up to 1/3.

Scenario 3 (HO ES)

This scenario describes homogeneous blocks with equal size. Three blocks are positioned like block A, B and C in Figure 6. The values of Γ are derived as a function of the aspect ratio q. The refinement levels are permuted among the blocks, resulting in 18 triangle graphs for each value of q. In this case, $\Gamma = 1$ for all q except in the interval $1/5 < q < 5$, where $\Gamma = 2/3$ (except for $q = 1$). The worst result is for $q = 1$, i.e., the quadratic case. This is because the communication requirement is often different for the neighboring blocks, but the distances are the same due to the quadratic form.



(a) $\Gamma = 1/3$



(b) $\Gamma = 2/9$

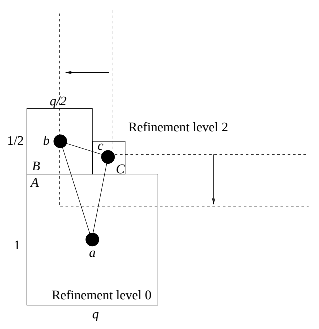
Figure 8: Scenario 2. Homogeneous blocks with no restrictions. Sub case (a) is a structured quadratic form, where $\Gamma = 1/3$. Sub case (b) the largest block (A) is rectangular, where $\Gamma = 2/9$. Hence, both cases exhibit negative coding.

Scenario 4 (HE EW R)

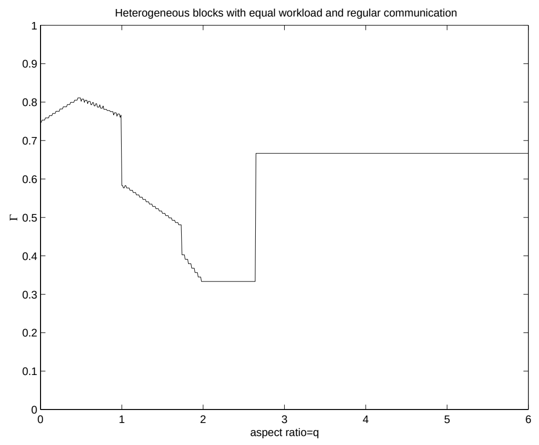
Scenario 4 describes heterogeneous blocks with equal workload and regular communication pattern. Three blocks are positioned as in Figure 9. The values of Γ, again, are derived as a function of the aspect ratio q. The most obvious way these block positions are created, is when a patch with higher refinement level is positioned somewhere as indicated by the figure. Due to the equal workload objective, the blocks are split as in the figure. In this case, a grid patch with refinement level 2 is allowed to move freely within the bounds given in Figure 9 in such a way that all positions have the same probability. Hence, there are infinitely many triangle graphs for each q. This informal integral is discretized and computed numerically. This case exhibits good behavior in general.

Scenario 5 (HE EW I)

Scenario 5 describes heterogeneous blocks with equal workload and irregular communication pattern. Three blocks are positioned as indicated by Figure 10. The three delimiters d_1, d_2, and d_3 are allowed to assume any position ($0 < d_1 < q$, $0 < d_2, d_3 < 1$) in such a way that all positions have the same probability. The amount of communication between two blocks A and B with the shared side s is approximated by $2s/(R_A + R_B)$, where R is the area of a block. Since this scenario assumes equal workload, we know that a smaller block has higher density (in terms of workload) than a large block. But we do not know how much (if any) of the finer resolution grids reside on the border. Therefore, we approximate the communication as being proportional to the (reciprocal of the) average of the areas. This informal integral was computed numerically. This



(a) A freely moving grid patch



(b) Γ as a function of q

Figure 9: Scenario 4. Heterogeneous blocks with equal workload and regular relations (see Section 2.3). This case displays much better behavior in general compared to the homogeneous case with equal workload.

case exhibits extremely high values of Γ.

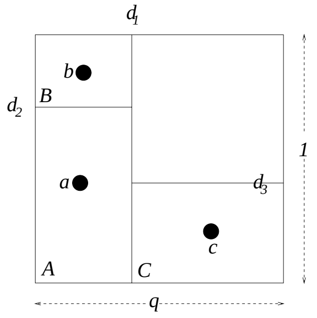
Scenario 6 (HE ES)

In Scenario 6, we study heterogeneous blocks of equal size. The scenario can be viewed in Figure 4 (a). In short, there are too many unknowns for deriving Γ in a more precise way. We can not assume anything about the number of refined grid patches or their aspect ratio and so on. However, it seems reasonable to justify a rough estimate $\Gamma \approx 0.5$, taken over the set of *all* possible grids, refinement patterns and so forth, in the following way.

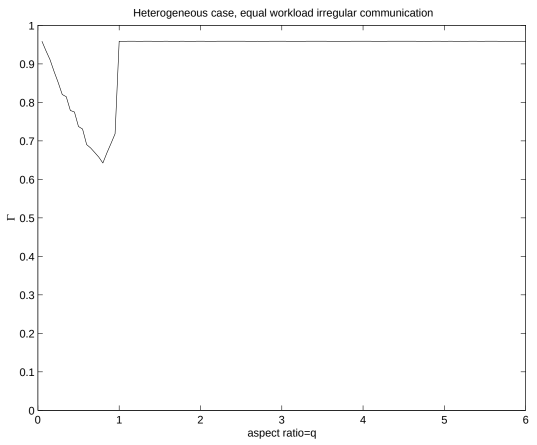
In this scenario, Assumption 1 is valid if none of the refinement patches that defines the heterogeneity touches the borders between the blocks. This would translate to the trivial case.

First, we assume that aspect ratio $q = 1$. Assumption 1 is *only* locally valid if the trivial case applies. The distances are exactly the same, requiring the amount of communication being the same, too.

Now, we assume $q \neq 1$. The trivial case *plus* some additional cases make Assumption 1 locally valid. These additional cases occur when when refined patches (by touching the correct block border) enhances the already advantageous situation created by the trivial case. The probability for this must be higher



(a)



(b) Γ as a function of q

Figure 10: Scenario 5. Heterogeneous blocks with equal workload and irregular communication (see Section 2.3). This scenario displays much better behavior in general compared to the other scenarios.

than the opposite, for two reasons: (1) The refined patches would first have to cancel the locally valid trivial case before it could turn it into a case not locally valid. (2) A refined patch touches the longer side with greater probability than a shorter side. More communication is already going on at the longer side.

3.3 Discussion

As pointed out in Section 3.1, Assumption 1 is trivially valid *in a global sense* for our block graphs. That is, a block will communicate only with blocks in a geographically close neighborhood, and there is no communication with blocks far away. Cases where this is not true are conceivable, for example if the grid has an exceptionally high aspect ratio. However, in most cases, Assumption 1 can be expected to hold in this global sense. This is the reason why ISP is considered a feasible alternative for these graphs.

However, locally, it is not certain that Assumption 1 holds. That is, among the geographically closest blocks, the communication volume is not necessarily related to proximity. Our investigation in Section 3.2 of local scenarios led to the results summarized in Table 2. They show that some scenarios have a higher degree of local validity of Assumption 1. The corresponding design decisions are:

- Blocks of equal size.

Scenarios	Γ for $0 < q < 6$
Trivial case	1
Homogeneous, equal workload	mostly <0.5
Homogeneous, no restrictions	<0.5
Homogeneous, equal size	>0.5
Heterogeneous, equal workload, R	mostly >0.5
Heterogeneous, equal workload, I	>0.5
Heterogeneous, equal size	0.5

Table 2: Summary of values for Γ for different cases of blocks. I=irregular; R=regular

- Heterogeneous blocks with equal workload, and regular block relations
- Heterogeneous blocks with equal workload, and irregular block relations

In Section 4, we present four new partitioning algorithms, which correspond to these advantageous design choices.

Table 2 also includes the trivial case, i.e., when all blocks have the same level of refinement, and are of equal size. This is the result of a HPF-type block partitioning of a uniform grid. It can be noted that for a SAMR application with strongly localized refinements large areas of the block graph will correspond to this trivial case. Since Assumption 1 is valid for the trivial case, it can be concluded that ISP can be expected to behave well for applications with such refinement patterns.

It could be noted that Table 2 is not complete in the sense that it does not list all possible combinations of blocks and relations between them. The case *HO, NR, I* was not further studied, because it contains too many unknowns for making meaningful derivations. The case *HE NR* does not make sense to investigate, since it does not correspond to any partitioning objective we could think of.

4 Four New Algorithms

4.1 Description

We now proceed with the presentation of four new partitioning algorithms. All four are domain-based and represent the overall partitioning approach discussed in Section 2.2. The algorithms proposed here correspond to the design decisions that are advantageous according to the investigation in Section 3.2.

In the following, we will describe the four algorithms in some detail. Then, we will discuss some experiences with using these algorithms for real-life applications. These experimental results agree with what would be expected on the basis of our theoretical investigations above.

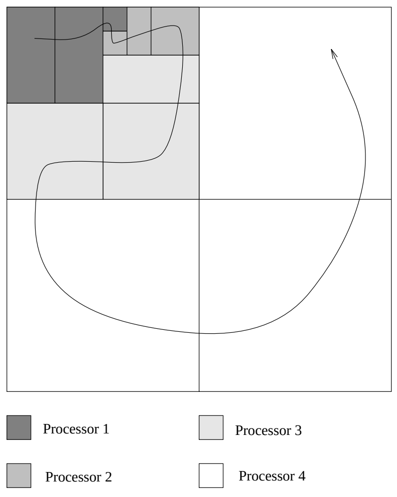


Figure 11: The G-MISP scheme is a result of the combination of design decisions a=1 and b=2,3 described in Section 2.3. It strives to create an internal list of blocks, each having approximately the same weight (workload).

Geometric multilevel inverse space-filling curve partitioning (G-MISP) with variable grain size

The G-MISP scheme is a result of the combination of design decisions a=1 (cuts at geometrical mid-point) and b=2,3 (approximate granularity, in part determined by other objective) described in Section 2.3. This algorithm produces mainly heterogeneous blocks with equal workload and regular relations, and blocks belonging to the trivial case. It is a multilevel algorithm that starts out by viewing the matrix of workloads as a one-vertex graph. The G-MISP algorithm strives to create an internal list of blocks, each having approximately the same weight (workload) (b=3). It successively refines the graph where needed. This is done by recursively splitting the (current) matrix as long as the accumulated workload is greater than some threshold. Each cut is made at the geometrical mid point (a=1). The parameter "atomic unit" acts as an upper bound of the granularity. Therefore, the resulting partitioning granularity is a function of the atomic unit, the threshold, and the current state of the dynamic grid hierarchy. Load balance is targeted by trivially assigning the same number of blocks to each processor. The idea is to generate a granularity fine enough (b=2) to make this procedure result in acceptable load balance, see Figure 11.

The algorithm is designed for speed through simplicity, and for giving acceptable communication patterns. The speed comes at the expense of load balance.

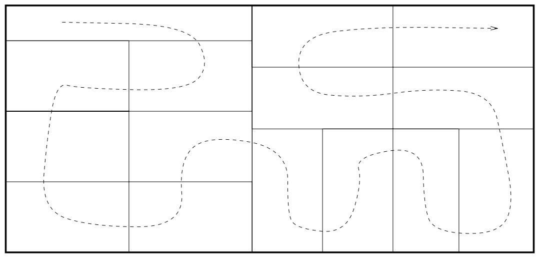


Figure 12: The pBD-ISP algorithm partitions the domain into p parts. Load balance is asserted in each cut. In the figure $p = 15$ and a uniform grid is used for simplification

Geometric multilevel inverse space-filling curve partitioning with sequence partitioning (G-MISP+SP) and variable grain size

The G-MISP+SP is a "smarter" variant of the G-MISP technique. It uses sequence partitioning [11] [1] to assign blocks from the one-dimensional list to processors, instead of just assigning the same number of blocks to each processor. As a result, load balance improves. The scheme is however computationally more expensive.

The p-way binary dissection inverse space-filling curve partitioning (pBD-ISP)

The pBD-ISP scheme is the result of the combination of design decisions a=2 (cuts for achieving optimal load balance) and b=1 (fixed number of blocks). This algorithm produces mainly heterogeneous blocks with equal workload and irregular relations. It is a generalization of the binary dissection algorithm discussed in [12], extending this scheme in two ways. First, the domain is partitioned into p partitions *without restrictions on p.* The binary dissection is performed in such a way that each split divides the load as evenly as possible, taking the available numbers of processors into account. That is, if there are e.g., 15 processors available, the cut is made such that 8/15 *of the load* is in one part and 7/15 is in the other part. Second, the orientation of the cuts is determined by the Hilbert space-filling curve, hence incorporating unstructuredness into the scheme. The idea is illustrated in Figure 12.

[1] Sequence partitioning is the problem to partition a sequence of n numbers into k intervals ($k < n$) by finding $k - 1$ delimiters such that the sum of the numbers in the interval with the largest sum is minimized.

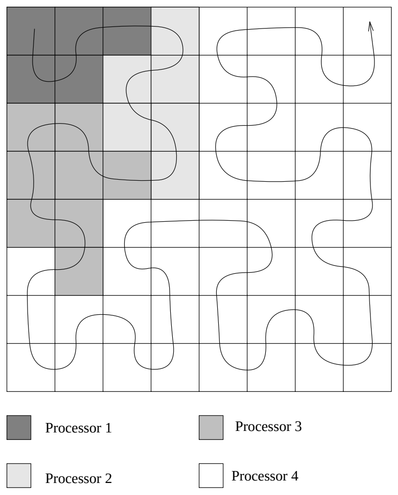


Figure 13: The SP scheme is the result of the combination of design decisions a=1 and b=1. The domain is subdivided into $p * k$ (b=1) equally sized blocks (a=1). These blocks are ordered according to the SFC to form a one-dimensional list, which is then partitioned using sequence partitioning.

"Pure" sequence partitioning with inverse space-filling curves (SP) and variable grain size

The SP scheme is the result of the combination of design decisions a=1 (cuts at geometrical mid-point) and b=1 (fixed number of blocks). This algorithm produces heterogeneous and homogeneous blocks with equal size, and blocks belonging to the trivial case. The domain is subdivided into $p * k$ (b=1) equally sized blocks (a=1) by a generalization of the parameterized binary dissection (BD) algorithm described in [13]. The BD algorithm is extended in two ways. First, SP is a dual-level algorithm, enabling different parameter settings for each level. In the first level, the domain is split into k blocks. In the second level, all k blocks are further divided into p blocks each. Second, the orientation of the cuts is determined by the Hilbert SFC. In the case of SAMR grid hierarchies, blocks generated will have different workloads due to adaptation of the grid. These blocks are ordered according to the SFC to form a one-dimensional list, which is then partitioned using sequence partitioning [11]. SP is potentially a fine granularity partitioning scheme, leading to good load balance at the expense of increased communication, overhead, and a higher computational cost.

4.2 Discussion

The four algorithms presented above have been implemented as parts of the parallel SAMR partitioning library Vampire [14]. This library has been used in experimental studies comparing several partitioning algorithms for a variety of applications.

The partitioning requirements for adaptive applications depend on the current state of the application and the computing environment. Therefore, it is of little consequence to discuss the absolute “goodness” of a particular partitioning technique. Hence, characterization of partitioning behavior should be based on information such as if the application and/or computer system cause communication or computation to dominate execution time, or if the application has high activity dynamics, hence calling for frequent repartitioning, or if the adaptation pattern is scattered or localized.

Our investigations in Section 3 lead us to the following conclusions. Due to their speed, ISP-friendliness, and limited ability to create good load balance, G-MISP and pBD-ISP, would be suitable for communication dominated applications requiring frequent repartitioning. Due to their relatively larger complexity, their ISP-friendliness *and* ability to create good load balance, G-MISP+SP and SP would be suitable for neutral applications requiring repartitioning less frequently. These conclusions are largely confirmed by the experimental studies in [15] and [16].

Moreover, an algorithm corresponding roughly to blocks and relations of types *HO EW* and *HO NR* was included in the studies in [15]. According to our investigations in Section 3, this algorithm would probably induce more communication, since it is not as ISP-friendly as the ones presented in this paper. This was confirmed in the experimental study. However, the particular algorithm created partitionings that were beneficial from other aspects.

5 Summary and Conclusions

Different design choices affect the local validity of one of the assumptions underlying ISP. Four specific combinations of design choices were found to be particularly advantageous from this perspective. As a result, we suggest four new partitioning algorithms, which correspond to these design choices. Generally, it was found that block graphs with heterogeneous blocks have better properties than block graphs with homogeneous blocks.

Acknowledgments

The author would like to thank Michael Thuné, Manish Parashar, Jarmo Rantakokko, and Göran Hamrin for fruitful discussions. Special thanks to Henrik Steensland for proof-reading all calculations and program listings.

References

[1] M. Parashar and J. C. Browne. On partitioning dynamic adaptive grid hierarchies. In *Proceedings of the 29th Annual Hawaii International Conference on System Sciences*, 1996.

[2] M. Parashar, et al. A common data management infrastructure for adaptive algorithms for PDE solutions. In *Proceedings of Supercomputing 1997*, 1997.

[3] Chao-Wei Ou and Sanjay Ranka. Parallel remapping algorithms for adaptive problems. Technical report, Center for Research on Parallel Computation, Rice University, 1994.

[4] John R. Pilkington and Scott B. Baden. Partitioning with spacefilling curves. Technical Report CS94-34, CSE, UCSD, March 1994.

[5] Mausumi Shee, Samip Bhavsar, and Manish Parashar. Characterizing the performance of dynamic distribution and load-balancing techniques for adaptive grid hierarchies. In *Proceedings IASTED International conference of parallel and distributed computing and systems*, 1999.

[6] Scott Kohn. SAMRAI homepage, structured adaptive mesh refinement applications infrastructure. http://www.llnl.gov/CASC/SAMRAI/, 1999.

[7] Stephen J. Fink, Scott B. Baden, and Scott R. Kohn. Flexible communication mechanisms for dynamic structured applications. In *Proceedings of IRREGULAR '96*, 1996.

[8] Jarmo Rantakokko. *Data Partitioning Methods and Parallel Block-Oriented PDE Solvers*. PhD thesis, Uppsala University, 1998.

[9] Leonid Oliker and Rupak Biswas. PLUM: Parallel load balancing for adaptive unstructured meshes. *Journal of Parallel and Distributed Computing*, 52(2):150–177, 1998.

[10] M. Thuné. Partitioning strategies for composite grids. *Parallel Algorithms and Applications*, 11:325–348, 1997.

[11] Bjorn Olstad and Fredrik Manne. Efficient partitioning of sequences. *IEEE Transactions on Computers*, 44(11):1322–1326, 1995.

[12] Marsha J. Berger and Shahid Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Trans. on Computers*, 85:570–580, 1987.

[13] S. H. Bokhari, T. W. Crockett, and D. M. Nicol. Binary dissection: variants & applications. Technical Report ICASE Report No. 97-29, NASA Langley Research Center, Hampton, VA, 1997.

[14] Johan Steensland. Vampire homepage,
http://www.caip.rutgers.edu/~johans/vampire/. 2000.

[15] Johan Steensland, Sumir Chandra, Manish Parashar, and Michael Thuné. Characterization of domain-based partitioners for parallel SAMR applications. In *Proceedings of PDCS2000*. IASTED, 2000.

[16] Johan Steensland, Stefan Söderberg, and Michael Thuné. A comparison of partitioning schemes for blockwise parallel SAMR applications. In *Proceedings of PARA2000*, volume 1947 of *LNCS*, 2000.