Fast & Effective Lossy Compression Algorithms for Scientific Datasets

Jeremy Iverson[1], Chandrika Kamath[2], and George Karypis[1]

[1] University of Minnesota, Minneapolis MN 55455, USA
[2] Lawrence Livermore National Laboratory, Livermore CA 94550, USA

Abstract. This paper focuses on developing effective and efficient algorithms for compressing scientific simulation data computed on structured and unstructured grids. A paradigm for lossy compression of this data is proposed in which the data computed on the grid is modeled as a graph, which gets decomposed into sets of vertices which satisfy a user defined error constraint ϵ. Each set of vertices is replaced by a constant value with reconstruction error bounded by ϵ. A comprehensive set of experiments is conducted by comparing these algorithms and other state-of-the-art scientific data compression methods. Over our benchmark suite, our methods obtained compression of 1% of the original size with average PSNR of 43.00 and 3% of the original size with average PSNR of 63.30. In addition, our schemes outperform other state-of-the-art lossy compression approaches and require on the average 25% of the space required by them for similar or better PSNR levels.

1 Introduction

The process of scientific discovery often requires scientists to run simulations, analyze the output, draw conclusions, then re-run the simulations to confirm or expand hypothesis. One of the most significant bottlenecks for current and future extreme-scale systems is I/O. In order to facilitate the scientific process described above, it is necessary for scientists to have efficient means to output and store data for offline analysis. To facilitate this, data compression is turned to, to create reduced representations of the resulting data for output, in such a way that the original result data can be reconstructed off-line for further analysis.

Straightforward approaches for scientific data compression exist in lossless techniques designed specifically for floating-point data. However, due to the high variability of the representation of floating-point numbers at the hardware level, the compression factors realized by these schemes are often very modest [4, 10]. Since most post-run analysis is robust in the presence of some degree of error, it is possible to employ lossy compression techniques rather than lossless, which are capable of achieving much higher compression rates at the cost of a small amount of reconstruction error. As a result, a number of approaches have been investigated for lossy compression of scientific simulation datasets including classical [7] and diffusion wavelets [3], spectral methods [5], and methods based on the techniques used for transmission of HDTV signals [2]. However, these approaches are either applicable only to simulations performed on structured grids or have high computational requirements for *in situ* data compression applications.

In this paper we investigate the effectiveness of a class of lossy compression approaches that replace the actual values associated with sets of grid-nodes with a constant value whose difference from the actual value is bounded by a user-supplied error tolerance parameter. We develop approaches for obtaining these sets by considering only the nodes and their values and approaches that constrain these sets to connected subgraphs in order to further reduce the amount of information that needs to be stored. To ensure that these methods are applicable for *in situ* compression applications, our work focuses on methods that have near-linear complexity and are equally applicable to structured and unstructured grids. We experimentally evaluate the performance of our approaches and compare it against that of other state-of-the-art data compression methods for scientific simulation datasets. Over our benchmark suite, our methods obtained compression of 1% of the original size with average PSNR of 43.00 and 3% of the original size with average PSNR of 63.30. Our experiments show that our methods achieve compressed representations, which on average, require 50%–75% less space than competing schemes at similar or lower reconstruction errors.

2 Definitions and Notations

The methods developed in this paper are designed for scientific simulations in which the underlying physical domain is modeled by a grid. Here we assume that the grid topology is fixed and thus can be compressed and stored separately from the data which is computed on it. Each node of a grid has one or more values associated with it that correspond to the quantities being computed in the course of the simulation. The grid can be either structured or unstructured. A *structured grid* is a collection of elements which have an implicit geometric structure. That structure is a basic rectangular matrix structure, such that in $\mathbb{R}^3$, the nodes can be indexed by a triplet (x, y, z). Thus, the grid topology can be described simply by the number of nodes in each of the three dimensions. An *unstructured grid* has no implicit structure. Since there is no implicit structure, the topology is described by identifying the elements which each node belongs to.

In this work, we model these grids via a *graph* $G = (V, E, L)$. The set of vertices V, models the nodes of the grid for which values are computed. The set of edges E, models the connectivity of adjacent nodes. Two nodes are adjacent if they belong to the same element in the grid. The set of vertex-labels L, models the values computed at each node of the grid such that l_i stores the value computed for node v_i. In this work we assume there is only one value being computed for each node of the grid.

An *ϵ-bounded set-based decomposition* of G is a partitioning of its set of vertices into non-overlapping sets $\{V_1, \ldots, V_k\}$ such that for each V_i, $\forall v_q, v_r \in V_i,\ |l_q - l_r| \leq \epsilon$ (i.e., each set contains vertices whose values differ at most by ϵ). When the induced subgraph $R_i = (V_i, E_i)$ of G is connected, the set V_i will also be referred to as a *region* of G. When all sets in an ϵ-bounded set-based decomposition form regions, then the decomposition will be referred to as an *ϵ-bounded region-based decomposition* of G. Given a set of vertices V_i, the average value of its vertices will be referred to as its *mean value* and will be denoted by $\mu(V_i)$. Given a region V_i, its *boundary vertices* are its subset of vertices $B_i \subseteq V_i$ that are adjacent to at least one other vertex not in V_i, and its *interior vertices* are the subset of vertices $I_i \subseteq V_i$ that are adjacent only to vertices in V_i. Note that $I_i \cup B_i = V_i$.

3 Related Work

Most of the work on lossy compression of scientific datasets has focused on compressing the simulation output for visualization purposes. The most popular techniques in this area are based on wavelet theory [7] that produces a compression-friendly sparse representation of the original data. To further sparsify this representation, coefficients with small magnitude are dropped with little impact on the reconstruction error [8, 9]. Due to the nature of the wavelet transform, classical wavelet methods apply only to structured grids. An alternative to wavelet compression is Adaptive Coarsening (AC) [11]. AC is an extension of the adaptive sub-sampling technique first introduced for transmitting HDTV signals [2], which is based on down-sampling a mesh in areas which can be reconstructed within some error tolerance and storing at full resolution the others. In [12], the authors use AC to compress data on structured grids and compare the results to wavelet methods. Even though AC can potentially be extended for unstructured grids [11], current implementations are limited to structured grids.

Another approach is spectral compression that extends the discrete cosine transform used in JPEG, from 2D regular grids to the space of any dimensional unstructured grids [5]. This method uses the Laplacian matrix of the grid to compute topology aware basis functions. The basis functions serve the same purpose as those in the wavelet methods and define a space where the data can be projected to, in order to obtain a sparse representation. Since the Laplacian matrix can be defined for the nodes of any grid, this method is not limited to structured grids. However, deriving the basis functions from the Laplacian matrix of large graphs is computationally prohibitive. For this reason, practical approaches first use a graph partitioning algorithm to decompose the underlying graph into small parts, and each partition is then compressed independently using spectral compression [5]. Finally, another approach, introduced in [3], is diffusion wavelets. The motivation for diffusion wavelets is the same as that of spectral compression, and is used to generate basis functions for a graph. However, instead of using the eigenvectors of the Laplacian matrix to derive these basis functions, diffusion wavelets generate them by taking powers of a diffusion operator. The advantage of diffusion wavelet is that its basis functions capture characteristics of the graph at multiple resolutions, while spectral basis functions only capture global characteristics.

4 Methods

In this work we investigated the effectiveness of a lossy compression paradigm for grid-based scientific simulation datasets that replaces the values associated with a set of nodes with a constant value whose difference from the actual values is bounded. Specifically, given a graph $G = (V, E, L)$ modeling the underling grid, this paradigm computes an ϵ-bounded set-based decomposition $\{V_1, \ldots, V_k\}$ of G and replaces the values associated with all the nodes of each set V_i, with its mean value $\mu(V_i)$. This paradigm bounds the point-wise error to be no more than ϵ, whose actual value is explicitly controlled by the users based on their subsequent analysis requirements. Since the values associated with the nodes tend to exhibit local smoothness [1], these value substitutions increase the degree of redundancy, which can potentially lead to better compression.

Following this paradigm, we developed two classes of approaches for obtaining the ϵ-bounded set-based decomposition of G. The first class focuses entirely on the vertices of the grid and their values, where the second class also takes into account the connectivity of these vertices in the graph. In addition, we developed different approaches for encoding the information that needs to be stored on the disk in order to maximize the overall compression. The description of these algorithms is provided in the subsequent sections.

In developing these approaches, our research focused on algorithms whose underlying computational complexity is low because we are interested in being able to perform the compression *in-situ* with the execution of the scientific simulation on future exascale-class parallel systems. As a result of this design choice, the algorithms that we present tend to find sub-optimal solutions but do so in time that in most cases is bounded by $O(|V| \log |V| + |E|)$.

4.1 Set-Based Decomposition

This class of methods derives the ϵ-bounded set-based decomposition $\{V_1, \ldots, V_k\}$ of the vertices by focusing entirely on their values. Towards this end, we developed two different approaches. The first is designed to find the decomposition that has the smallest cardinality (i.e., minimize k), whereas the second is designed to find a decomposition that contains large-size sets.

The first approach, referred to as *SBD1*, operates as follows. The vertices of G are sorted in non-decreasing order based on their values. Let $\langle v_{i_1}, \ldots, v_{i_n} \rangle$ be the sequence of the vertices according to this ordering, where n is the number of vertices in G. The vertices are then scanned sequentially from v_{i_1} up to vertex v_{i_j} such that $l_{i_j} - l_{i_1} \leq \epsilon$ and $l_{i_{j+1}} - l_{i_1} > \epsilon$. The vertices in the set $\{v_{i_1}, \ldots, v_{i_j}\}$ satisfy the constraint of an ϵ-bounded set and are used to form a set of the set-based decomposition. These vertices are then removed from the sorted sequence and the above procedure is repeated on the remaining part of the sequence until it becomes empty. It can be easily shown that the above greedy algorithm will produce a set-based decomposition that has the smallest number of sets for a given ϵ.

The second approach, referred to as *SBD2*, utilizes the same sorted sequence of vertices $\langle v_{i_1}, \ldots, v_{i_n} \rangle$ but it uses a different greedy strategy for constructing the ϵ-bounded sets. Specifically, it identifies the pair of vertices v_{i_q} and v_{i_r} such that $l_{i_r} - l_{i_q} \leq \epsilon$ and $r - q$ is maximized. The vertices in the set $\{v_{i_q}, \ldots, v_{i_r}\}$ satisfy the constraint of an ϵ-bounded set and are used to form a set of the set-based decomposition. The original sequence is then partitioned into two parts: $\langle v_{i_1}, \ldots v_{i_{q-1}} \rangle$ and $\langle v_{i_{r+1}}, \ldots, v_{i_n} \rangle$, and the above procedure is repeated recursively on each of these subsequences. Note that the greedy decision in this approach is that of finding a set that has the most vertices (by maximizing $r - q$). It can be shown that SDB2 will lead to a decomposition whose maximum cardinality set will be at least as large as the maximum cardinality set of SBD1 and that the cardinality of the decomposition can be greater than that of SDB1's decomposition.

Decomposition Encoding We developed two approaches for encoding the vertex values derived from the ϵ-bounded set-based decomposition. In both of these approaches, the encoded information is then further compressed using standard lossless compression methods such as GZIP, BZIP2, and LZMA.

The first approach uses scalar quantization and utilizes a pair of arrays Q and M. Array Q is of size k (the cardinality of the decomposition) and $Q[i]$ stores the mean value $\mu(V_i)$ of V_i. Array M is of size n (the number of vertices) and $M[j]$ stores the number of the set that vertex v_j belongs to. During reconstruction, the value of v_j is given by $Q[M[j]]$. Since for reasonable values of ϵ, $k \ll n$, the number of distinct values in M will be small, leading to a high degree of redundancy that can be exploited by the subsequent lossless compression step. We will refer to this approach as *scalar quantization encoding* and denote it by *SQE*.

The second approach encodes the information by sequentially storing the vertices that belong to each set of the decomposition. Specifically, it uses three arrays Q, S, and P, of sizes k, k, and n, respectively. Array Q is identical to the Q array of SQE and array S stores the number of vertices in each set (i.e., $S[i] = |V_i|$). Array P is used to store the vertices of each set in consecutive positions, starting with those of set V_1, followed by V_2, and so on. The vertices of each set are stored by first sorting them in increasing order based on their number and then representing them using a differential encoding scheme. The smallest numbered vertex of each set is stored as is and the number of each successive vertex is stored as the difference from the preceding vertex number. Since each vertex-set will likely have a large number of vertices, the differential encoding of the sorted vertex lists will tend to consist of many small values, and thus increase the amount of redundancy that can be exploited by the subsequent lossless compression step. We will refer to this approach as *differential encoding* and denote it by *DE*.

Vertex Ordering To achieve good compression using the above encoding schemes, vertices which are close in the vertex ordering should have similar values. Towards this end, we investigate three vertex orderings which are as follows. The first is the original ordering of the nodes, that is often derived by the grid generator and tends to have a spatial coherence. The second ordering is a breadth first traversal of the graph starting from a randomly selected vertex. The third ordering is a priority first traversal, in which priority is given to those vertices which are adjacent to the most vertices which have been previously visited. Arranging the vertices according to their visit order is intended to put together in the ordering vertices that are close in the graph topology. Due to the local smoothness of values, vertices that appear close in the ordering will share similar values.

4.2 Region-Based Decomposition

This class of methods derives an ϵ-bounded set-based decomposition $\{V_1, \ldots, V_k\}$ by requiring that each set V_i also forms a region (i.e., its induced subgraph of G is connected). The motivation behind this region-based decomposition is to reduce the amount of data that needs to be stored by only writing information about V_i's boundary vertices and a select few of its interior vertices. During reconstruction, by taking advantage of V_i's connectivity, its non-saved interior vertices can be identified by a depth- or breadth-first traversal of G starting at the saved interior vertices and terminating at its boundary vertices. The set of vertices visited in the course of this traversal will be exactly those in V_i. From this discussion, we see that the amount of compression that can be achieved by this class of methods is directly impacted by the number of boundary vertices that must be stored. Thus, the region identification

approaches must try to reduce the number of boundary vertices. Towards this end, we developed three different heuristic approaches whose description follows.

The first approach, referred to as *RBD1* , is designed to compute a decomposition that minimizes the number of regions. The motivation behind this approach is that by increasing the average size of each region (due to a reduction in the decomposition's cardinality), the number of interior vertices will also increase. RBD1 initially sorts the vertices in a way identical to SBD1, leading to the sorted sequence $s = \langle v_{i_1}, \ldots, v_{i_n} \rangle$. Then, it selects the first vertex in the sequence (v_{i_1}), assigns it to the first region V_1, and removes it from s. It then proceeds to select from s a vertex v_{i_j} that is adjacent to at least one vertex in V_1 and $l_{v_{i_j}} - l_{v_1} \leq \epsilon$, inserts it into V_1, and removes it from s. This step is repeated until no such vertex can be selected or s becomes empty. The above algorithm ensures that V_1 is an ϵ-bounded set and that the subgraph of G induced by V_1 is connected. Thus, V_1 is a region and is included in the region-based decomposition. The above procedure is then repeated on the vertices remaining in s, each time identifying an additional region that is included in the decomposition. Note that unlike the algorithm for SBD1, the above algorithm does not guarantee that it will identify the ϵ-bounded region-based decomposition that has the minimum number of regions.

The second approach, referred to as *RBD2* , is designed to compute a decomposition that contains large regions, as the regions that contain a large number of vertices will also tend to contain many interior vertices. One way of developing such an algorithm is to use the greedy approach similar to that employed by SBD2 to repeatedly find the largest region from the unassigned vertices and include it in the decomposition. However, due to the region's connectivity requirement, this is computationally prohibitive. For this reason, we developed an algorithm that consists of two steps. The first step is to obtain an ϵ-bounded set-based decomposition $\{V_1, \ldots, V_k\}$ using SBD1. The second step is to compute an ϵ-bounded region-based decomposition of each set V_i. The union of these regions over $V_1, \ldots, V_k$ is then used as the region-based decomposition computed by RBD2. This two-step approach is motivated by the following observation. One of the reasons that prevents RBD1 from identifying large regions is that it starts growing each successive region from the lowest-valued unassigned vertex and does not stop until all of the unassigned vertices adjacent to that region have values that will violate the ϵ bound. This will tend to fragment subsequent regions as the are constrained by the initial vertices that have low values. RBD2, by forcing RBD1's region identification algorithm to stay within each set V_i, prevents this from happening and as our experiments will later show, lead to a decomposition that has smaller number of boundary vertices and better compression.

Finally, the third approach, referred to as *RBD3* , is designed to directly compute a decomposition whose regions have a large number of interior vertices. It consists of three distinct phases. The first phase identifies a set of *core* regions that contain at least one interior vertex, the second phase expands these regions by including additional vertices to them, and the third phase creates non-core regions. Let V' be the subset of vertices of V such that $\forall v \in V', v \cup \text{adj}(v)$ is an ϵ-bounded set, where $\text{adj}(v)$ is the set of vertices adjacent to v. A core region, V_i, is created as follows. An unassigned vertex $v \in V'$ whose adjacent vertices are also unassigned is randomly selected and $v \cup \text{adj}(v)$ is inserted into V_i. Then the algorithm proceeds to identify

an unassigned vertex $u \in V'$ such that: (i) it is connected to at least one vertex in V_i, (ii) all the vertices in $\text{adj}(u) \setminus V_i$ are also unassigned, and (iii) $V_i \cup \{u\} \cup \text{adj}(u)$ is an ϵ-bounded set. If such a vertex u exists, then u and $\text{adj}(u) \setminus V_i$ are inserted into V_i. If no such vertex exists, then V_i's expansion stops. The above procedure is repeated until no more core regions can be created. Note that by including u and its $\text{adj}(u) \setminus V_i$ vertices into V_i, we ensure that u becomes an interior vertex of V_i. During the second phase of the algorithm, the vertices that have not been assigned to any region are considered. If a vertex v can be included to an existing region while the resulting region remains an ϵ-bounded set, then it is assigned to that. Finally, the third phase is used to create additional regions containing the remaining unassigned vertices (if they exist), which is done using RBD1.

Decomposition Encoding As discussed earlier, the region-based decomposition allows us to reduce the storage requirements by storing only the boundary vertices along with the interior vertices that are used as the *seeds* of the (depth- or breadth-first) traversals. For each region V_i, the set of seed-vertices I_i^s is determined as follows. An interior vertex is randomly selected, added to I_i^s, and a traversal from that vertex is performed terminating at V_i's boundary vertices. If any of V_i's interior vertices has not been visited, then the above procedure is repeated on the unvisited vertices, each time adding an additional source vertex into I_i^s. In most cases, one seed vertex will be sufficient to traverse all the interior vertices, but when regions are contained within other regions, multiple seed vertices may be required. Also, in the cases in which V_i consists of only boundary vertices, I_i^s will be empty.

An additional storage optimization is possible, as there is no need to store the boundary vertices for all the regions. In particular, consider a region V_i and let $\{V_{i_1}, \ldots, V_{i_m}\}$ be the set of its adjacent regions in the graph. We can then identify V_i by performing a traversal from the vertices in I_i^s that terminates at the boundary vertices of V_i's adjacent regions. All the vertices visited during that traversal (excluding the boundary vertices) along with I_i^s will be exactly the vertices of V_i. Thus, we can choose not to store V_i's boundary vertices as long as we store the boundary vertices for all of its adjacent regions. In our algorithm, we choose the regions whose boundary information will not be stored in a greedy fashion based on the size of their boundaries. Specifically, we construct the region-to-region adjacency graph (i.e., two regions are connected if they contain vertices that are adjacent to each other), assign a weight to the vertex corresponding to V_i that is equal to $|B_i|$ (i.e., the size of its boundary), and then identify the regions whose boundary information will not be stored by finding a maximal weight independent set of vertices in this graph using a greedy algorithm.

Given the above, we can now precisely describe how the region-based decomposition is stored. Let $\{V_1, \ldots, V_k\}$ be the ϵ-bounded region-based decomposition, $B_1, \ldots, B_k$ be the sets of boundary vertices that need to be stored (if no boundary information is stored for a region due to the earlier optimization, then the corresponding boundary set is empty), and $I_1^s, \ldots, I_k^s$ be the sets of internal seed-vertices that have been identified. Our method stores five arrays, Q, N_I, N_B, I_I, and I_B. The first three arrays are of length k, I_I is of length equal to the total number of seed vertices ($\sum_i |I_i^s|$), and I_B is of length equal to the total number of boundary vertices ($\sum_i |B_i|$). Array Q stores the mean values of each region, whereas arrays

Table 1. Information about the various datasets.

Dataset	$\|V\|$	$\|E\|$	$\mu(V)$	Grid Type	Dataset	$\|V\|$	$\|E\|$	$\mu(V)$	Grid Type
d1	486051	4335611	0.9958	unstruct.	d5	31590144	94562224	0.0176	unstruct.
d2	589824	1744896	0.5430	struct.	d6	41472000	123926400	0.2107	struct.
d3	1936470	15399496	0.9874	unstruct.	d7	100663296	300744704	4.5644	struct.
d4	16777216	50102272	163.70	struct.					

N_I and N_B store the number of seed and boundary vertices of each region, respectively. Array I_I stores the indices of the regions in consecutive order starting from I_1^s, whereas array I_B is used to store the boundary vertices of each region in consecutive positions starting from B_1. These indices are stored using the same differential encoding approach described in Sect. 4.1 and like that approach, the results of this encoding are further compressed using a standard lossless compression method.

5 Experimental Design & Results

Datasets We evaluated our algorithms using seven real world datasets obtained from researchers at UMN and our colleagues at NASA and LLNL. These datasets correspond to fluid turbulence and combustion simulations and contain both structured and unstructured grids. Their characteristics are shown in Table 1.

Evaluation Methodology & Metrics We measured the performance of the various approaches along two dimensions. The first is the error introduced by the lossy compression and the second is the degree of compression that was achieved. The error was measured using three different metrics: (i) the root mean squared error (RMSE), (ii) the maximum point-wise error (MPE), and (iii) the peak signal-to-noise ratio (PSNR). The RMSE is defined as

$$RMSE = \sqrt{\frac{1}{|V|}\sum_{i=1}^{|V|}|l_j - \hat{l}_j|^2}, \tag{1}$$

where l_j is the original value of vertex v_j and $\hat{l}_j$, is its reconstructed value. The MPE is defined as

$$MPE = \max(|l_1 - \hat{l}_1|, ..., |l_n - \hat{l}_n|), \tag{2}$$

which is the ℓ_∞-norm of the point-wise error vector. The MPE measure is presented in tandem with RMSE to identify those algorithms which achieve low RMSE, but sustain high point-wise errors. Finally, the PSNR is defined as

$$PSNR = 20 \cdot \log_{10}\left(\frac{\max(x_1, ..., x_n)}{RMSE}\right), \tag{3}$$

which is a normalized error measure; thus, facilitating comparisons of error between datasets with values that differ greatly in magnitude. The compression effectiveness was measured by computing the compression ratio (CR) of each method, which is defined as follows:

$$CR = \frac{\text{compressed size}}{\text{uncompressed size}}. \tag{4}$$

The wavelet and spectral methods were implemented in Matlab®. The spectral method uses METIS [6] as a pre-processing step to partition the graph before compressing. The adaptive coarsening implementation was acquired from the authors

Table 2. Set-based compression results.

Dataset	Method	k	Original		BFT		PFT	
			SQE	DE	SQE	DE	SQE	DE
d1	SBD1	22	2.39E-02	3.58E-02	6.32E-02	4.29E-02	5.97E-02	4.43E-02
	SBD2	32	2.48E-02	3.53E-02	6.35E-02	4.31E-02	5.99E-02	4.27E-02
d2	SBD1	19	2.51E-03	7.66E-03	4.47E-02	7.35E-02	4.26E-03	1.81E-02
	SBD2	28	3.47E-03	1.04E-02	5.91E-02	8.44E-02	5.72E-03	2.15E-02
d3	SBD1	33	1.27E-02	2.34E-02	6.65E-02	3.33E-02	6.53E-02	3.33E-02
	SBD2	47	1.22E-02	2.21E-02	6.52E-02	3.21E-02	6.40E-02	3.16E-02
d4	SBD1	33	2.63E-03	2.98E-03	2.18E-02	1.85E-02	3.28E-03	5.62E-03
	SBD2	38	3.01E-03	3.15E-03	2.30E-02	1.90E-02	3.67E-03	6.29E-03
d5	SBD1	45	3.22E-03	4.00E-03	2.90E-02	2.29E-02	1.60E-02	1.84E-02
	SBD2	64	3.13E-03	3.71E-03	2.89E-02	2.11E-02	1.60E-02	1.68E-02
d6	SBD1	17	9.30E-03	1.90E-02	1.90E-02	2.28E-02	1.03E-02	1.97E-02
	SBD2	29	9.87E-03	2.03E-02	2.19E-02	2.35E-02	1.09E-02	2.11E-02
d7	SBD1	40	2.82E-02	6.01E-02	3.80E-02	6.79E-02	3.08E-02	6.15E-02
	SBD2	56	2.85E-02	6.05E-02	3.85E-02	6.83E-02	3.11E-02	6.20E-02

of [12] and modified to provide the statistics necessary for these experiments. All algorithms described in Sect. 4 were implemented in C++. Finally, for the lossless compression of the decomposition encodings, we used LZMA compression (7-zip's implementation) as it resulted in better compression than either GZIP or BZIP2. In addition, the same LZMA-based compression was applied to the output of the spectral and wavelet-based compressions. Note that AC does not need that because it achieves its compression by coarsening the graph and reducing the data output.

6 Results

Our experimental evaluation is done in two parts. First, we select a fixed set of values for RMSE and compare the various algorithmic choices for the set- and region-based decomposition approaches in terms of their compression ability. Second, we compare the compression performance of the best combinations of these schemes against that achieved by other approaches for two different levels of lossy compression errors.

6.1 Set-Based Decomposition

Table 2 shows the compression performance achieved by SBD1 and SBD2 for the different datasets across the different vertex ordering and decomposition encoding schemes described in Section 4.1. These results show that SBD1 tends to perform somewhat better than SBD2 and on average, it requires 5% less storage for each specific combination of decomposition encoding and vertex ordering scheme. This can be attributed to the fact that the cardinality of its decomposition is often considerably lower than SBD2's (shown in the column labeled "k"), which tends to outweigh the benefits achieved by the few larger sets identified by SBD2.

Comparing the performance of the decomposition encoding schemes (SQE and DE), we see that SQE performs considerably better across both decomposition methods and ordering schemes. On the average, SQE requires only 75% of the storage of DE. These results suggest that when compared to scalar quantization, the differential encoding of the vertices in each set is not as effective in introducing redundancy in the encoding, which in turn reduces the compression that can be obtained by the lossless LZMA compression. Finally, comparing the performance of the three vertex ordering schemes, we see that the original ordering leads to greater compression than either of the other two. As discussed in Section 4.1, this ordering

Table 3. Region-based compression results.

Dataset	Method	|R|	NB	Original	BFT	PFT
d1	RBD1	152	153567	3.97E-02	2.94E-02	4.36E-02
	RBD2	174	105705	2.71E-02	1.91E-02	2.97E-02
	RBD3	1100	104181	3.93E-02	3.37E-02	4.05E-02
d2	RBD1	852	399593	1.19E-02	6.22E-02	2.02E-02
	RBD2	1016	393802	1.19E-02	5.92E-02	1.99E-02
	RBD3	4776	360609	3.80E-02	9.99E-02	5.47E-02
d3	RBD1	312	312220	1.73E-02	2.03E-02	1.33E-02
	RBD2	361	248773	1.31E-02	1.57E-02	1.05E-02
	RBD3	1667	271304	1.81E-02	2.00E-02	1.51E-02
d4	RBD1	51210	1503191	6.30E-03	1.65E-02	1.11E-02
	RBD2	60301	1500247	5.67E-03	1.55E-02	9.99E-03
	RBD3	94704	1511615	1.22E-02	2.29E-02	1.85E-02
d5	RBD1	23025	5832563	5.17E-03	1.81E-02	1.11E-02
	RBD2	29358	3883703	3.55E-03	1.12E-02	6.76E-03
	RBD3	107479	3910251	1.37E-02	2.58E-02	1.82E-02
d6	RBD1	69875	11435672	3.62E-02	4.83E-02	4.59E-02
	RBD2	95618	5007376	1.87E-02	2.25E-02	2.09E-02
	RBD3	281600	5112178	2.92E-02	3.81E-02	3.90E-02
d7	RBD1	358720	73369522	1.02E-01	1.33E-01	1.35E-01
	RBD2	357382	36819989	5.60E-02	6.62E-02	6.23E-02
	RBD3	2055003	38992553	9.48E-02	1.19E-01	1.25E-01

NB: number of boundary nodes after applying storage optimization discussed in Sect. 4.1

utilizes information from the underlying grid geometry, and as such it has a higher degree of regularity, leading to better compression. With respect to the other two methods, we see that PFT tends to perform better than BFT.

6.2 Region-Based Decomposition

Table 3 shows various statistics of the decompositions computed by RBD1, RBD2, and RBD3 for the different datasets and their compression performance for the three vertex ordering schemes. In terms of the number of regions into which G is decomposed, we see that RBD1 results in the least number of regions, whereas RBD3 identifies a considerably greater number of regions (often 2–7 times more regions than RBD1). We also see that RBD2 only identifies slightly more regions than RBD1 (about 18% more on average). In terms of the number of boundary vertices that need to be stored by each decomposition, we see an inversion of the previous results. RBD2 and RBD3 produce the smallest boundary sets, typically being within about 5% of each other, whereas RBD1 produces boundary sets which are considerably larger, in some cases, more than twice the size of those required by RBD2 and RBD3. These results suggest that the region identification heuristics employed by RBD2 and RBD3 are quite effective in minimizing the total number of boundary vertices, even though they find more regions.

In terms of compression performance, we see that across all datasets RBD2 results in the lowest compression ratio. On the average, RBD2 requires only 70% of the storage of RBD1 and 56% of RBD3. Contrasting this with the number of boundary vertices identified by each approach, we see that there is a direct correlation, based on the size of the boundary vertex set, between RBD1 and RBD2 in terms of which approach results in lower compression ratio and by how much. RBD3 does not share in this correlation, due to its significantly higher number of regions.

6.3 Comparison with Other Methods

In our last set of experiments, we compare the performance of the best-performing combinations of the set- and region-based decomposition approaches (SBD1 with

Table 4. Comparison of scientific data compression algorithms for two different rmse.

info		high error tolerance				low error tolerance			
Dataset	Algorithm	RMSE	PSNR	MPE	CR	RMSE	PSNR	MPE	CR
	SBD1	6.30E-03	4.64E+01	1.89E-02	2.39E-02	6.66E-04	6.60E+01	1.86E-03	**7.80E-02**
d1	RBD2	6.28E-03	4.65E+01	1.89E-02	**2.52E-02**	6.38E-04	6.63E+01	2.12E-03	1.28E-01
	Spctrl	6.37E-03	4.63E+01	1.11E-01	4.00E-02	3.90E-03	5.06E+01	7.14E-02	1.05E-01
	SBD1	2.92E-02	3.60E+01	7.33E-02	**2.51E-03**	2.50E-03	5.74E+01	7.71E-03	**1.27E-02**
	RBD2	2.88E-02	3.61E+01	8.02E-02	5.02E-03	1.91E-03	5.97E+01	7.71E-03	6.57E-02
d2	Wvlt	3.10E-02	3.55E+01	2.34E-01	2.00E-02	2.59E-03	5.70E+01	2.38E-02	1.15E-01
	Spctrl	3.17E-02	3.53E+01	7.34E-01	4.50E-02	7.04E-03	4.84E+01	3.56E-01	1.30E-01
	AC	3.31E-02	3.49E+01	1.50E-01	1.86E-02	6.80E-03	4.87E+01	7.19E-01	5.17E-02
	SBD1	5.22E-03	4.88E+01	1.91E-02	**1.27E-02**	4.79E-04	6.96E+01	2.05E-03	**3.56E-02**
d3	RBD2	5.18E-03	4.89E+01	1.93E-02	1.33E-02	4.54E-04	7.00E+01	2.07E-03	4.33E-02
	Spctrl	5.27E-03	4.87E+01	2.14E-01	4.50E-02	3.31E-03	5.28E+01	1.35E-01	1.00E-01
	SBD1	2.36E+01	4.70E+01	1.63E+02	**2.63E-03**	2.43E+00	6.68E+01	1.34E+01	**1.02E-02**
	RBD2	2.05E+01	4.83E+01	1.65E+02	6.30E-03	2.00E+00	6.85E+01	1.36E+01	3.51E-02
d4	Wvlt	2.47E+01	4.66E+01	6.86E+02	7.50E-03	2.64E+00	6.61E+01	4.87E+01	2.50E-02
	Spctrl	2.57E+01	4.63E+01	1.78E+03	3.50E-02	3.92E+00	6.26E+01	3.59E+02	1.95E-01
	AC	2.30E+01	4.73E+01	3.01E+03	2.15E-02	-	-	-	-
	SBD1	4.97E-04	4.59E+01	1.78E-03	**3.22E-03**	5.43E-05	6.51E+01	1.28E-04	**1.42E-02**
d5	RBD2	4.88E-04	4.61E+01	1.76E-03	4.47E-03	5.32E-05	6.53E+01	1.69E-04	4.96E-02
	Spctrl	5.84E-04	4.45E+01	4.56E-02	5.00E-03	5.87E-05	6.45E+01	8.74E-03	6.50E-02
	SBD1	1.21E-02	3.82E+01	5.70E-02	9.30E-03	1.05E-03	5.94E+01	4.87E-03	**2.96E-02**
	RBD2	1.20E-02	3.82E+01	5.71E-02	1.28E-02	8.75E-04	6.10E+01	4.87E-03	1.74E-01
d6	Wvlt	9.48E-03	4.03E+01	1.56E-01	**5.00E-03**	1.05E-03	5.94E+01	1.17E-02	5.50E-02
	Spctrl	1.60E-02	3.57E+01	6.37E-01	**5.00E-03**	1.05E-03	5.94E+01	4.32E-02	6.50E-02
	AC	1.82E-02	3.46E+01	1.50E-01	1.11E-02	-	-	-	-
	SBD1	2.72E-01	4.27E+01	5.50E-01	2.82E-02	2.74E-02	6.26E+01	6.37E-02	**8.70E-02**
	RBD2	2.70E-01	4.28E+01	7.41E-01	3.43E-02	2.17E-02	6.47E+01	7.99E-02	5.16E-01
d7	Wvlt	2.76E-01	4.26E+01	2.75E+00	**1.00E-02**	3.05E-02	6.17E+01	2.00E-01	1.60E-01
	AC	2.76E-01	4.26E+01	1.00E+00	1.82E-02	-	-	-	-

bold indicates the lowest CR for a given dataset and error tolerance

SQE encoding and original vertex ordering, and RBD2 with original vertex ordering) against wavelet compression (Wvlt), spectral compression (Spctrl), and adaptive coarsening (AC). Among these techniques, the wavelet compression and adaptive coarsening can only be applied to structured grids and are only presented for the d2, d4, d6, and d7 datasets. Also, due to its high computational requirements, we were not able to obtain results for the spectral compression for the largest problem (d7). In addition to these schemes, we also experimented with diffusion wavelets [3]. However, we obtained poor compression and we omitted those results.

Table 4 shows the results of these experiments for two different compression levels, labeled "high error tolerance" and "low error tolerance". These compression levels result in RMSEs and MPEs that differ by approximately an order of magnitude, and were obtained by experimenting with the parameters of the various schemes so that to match their RMSEs for each of the datasets. However, for AC we were unable to achieve the desired RMSEs at all error tolerance levels. In the case that we could not achieve a desired RMSE, the results were omitted.

The results show that on average, our algorithms compress the simulation datasets to 2–5% of their original size. Compared with just lossless compression only, which results in storage costs of 40–80% of the original size, this is a big improvement. The results also show that for all but two experiments, SBD1 performs the best and that on average it required only 36% of the storage of the next best algorithm. For unstructured grids it requires on average 25% of the storage of Spctrl whereas for structured grids it requires on average 48% and 38% of the space of Wvlt and AC, respectively. Moreover, we see that as the amount of allowable error is lowered, the performance gap between SBD1 and the other methods grows. In addition, for unstructured grids, RBD2 performs the second best overall and requiring 61% of

the space required by the Spctrl on average. We also see that due to the ϵ constraint placed on the our methods, they consistently result in MPE values which are much lower than those of the competing algorithms. These results suggest that in the context of grid-based simulation, SBD1 and RBD2 are consistently good choices for compression, providing low point-wise and global reconstruction error, high compression ratio, and low computational complexity.

7 Conclusion

In this paper, we introduced a paradigm for lossy compression of grid-based simulation data that achieves compression by modeling the grid data via a graph and identifying vertex-sets which can be approximated by a constant value within a user provided error constraint. Our comprehensive set of experiments showed that for structured and unstructured grids, these algorithms achieve compression which results in storage requirements that on average, are up to 75% lower than that other methods. Moreover, the near linear complexity of these algorithms makes them ideally suited for performing *in situ* compression in future exascale-class parallel systems.

References

1. Baldwin, C., Abdulla, G., Critchlow, T.: Multi-resolution Modeling of Large Scale Scientific Simulation Data. Proceedings of the twelfth international conference on Information and knowledge management - CIKM '03 p. 40 (2003)
2. Belfor, R.A.F., Hesp, M.P.A., Lagendijk, R.L., Biemond, J.: Spatially Adaptive Subsampling of Image Sequences. IEEE Transactions on Image Processing 3(5), 492–500 (Jan 1994)
3. Coifman, R., Maggioni, M.: Diffusion wavelets. Applied and Computational Harmonic Analysis 21(1), 53–94 (Jul 2006)
4. Engelson, V., Fritzson, D., Fritzson, P.: Lossless Compression of High-volume Numerical Data from Simulations. Data Compression Conference pp. 574–586 (2000)
5. Karni, Z., Gotsman, C.: Spectral Compression of Mesh Geometry. Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '00 pp. 279–286 (2000)
6. Karypis, G.: METIS˜5.0: Unstructured graph partitioning and sparse matrix ordering system. Tech. rep., Department of Computer Science, University of Minnesota (2011)
7. Mallat, S.: A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(7), 674–693 (Jul 1989)
8. Muraki, S.: Approximation and Rendering of Volume Data Using Wavelet Transforms. Proceedings Visualization '92 pp. 21–28 (1992)
9. Muraki, S.: Volume Data and Wavelet Transforms. IEEE Computer Graphics and Applications 13(4), 50–56 (Jul 1993)
10. Ratanaworabhan, P., Ke, J., Burtscher, M.: Fast Lossless Compression of Scientific Floating-Point Data. Data Compression Conference (DCC'06) pp. 133–142 (2006)
11. Shafaat, T.M., Baden, S.B.: A Method of Adaptive Coarsening for Compressing Scientific Datasets. In: Applied parallel computing: State-of-the-Art in Scientific and Parallel Computing, 8th Intl. Workshop, Proc. PARA '06. pp. 1–7 (2006)
12. Unat, D., Hromadka III, T., Baden, S.B.: An Adaptive Sub-sampling Method for In-memory Compression of Scientific Data. 2009 Data Compression Conference pp. 262–271 (Mar 2009)