Alcom-FT Technical Report Series
ALCOMFT-TR-03-144

Graph Partitioning in Scientific Simulations: Multilevel Schemes versus Space-Filling Curves *

Stefan Schamberger[1] and Jens-Michael Wierum[2]

[1] University of Paderborn, Germany
schaum@upb.de, http://www.upb.de/

[2] Paderborn Center for Parallel Computing, Germany
jmwie@upb.de, http://www.upb.de/pc2/

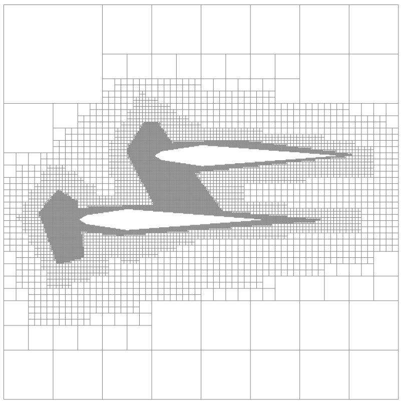
Abstract. Using space-filling curves to partition unstructured finite element meshes is a widely applied strategy when it comes to distributing load among several computation nodes. Compared to more elaborated graph partitioning packages, this geometric approach is relatively easy to implement and very fast. However, results are not expected to be as good as those of the latter, but no detailed comparison has ever been published. In this paper we will present results of our experiments comparing the quality of partitionings computed with different types of space-filling curves to those generated with the graph partitioning package Metis.

Keywords: FEM graph partitioning, space-filling curves

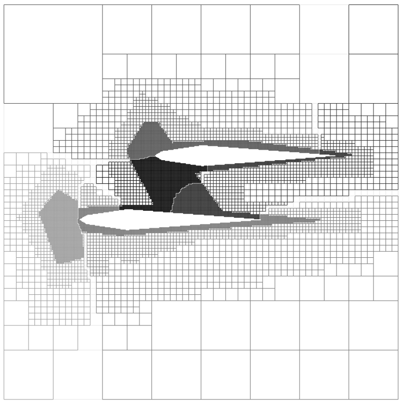
1 Introduction

Finite Elements (FE) are often used to numerically approximate solutions of a Partial Differential Equation (PDE) describing physical processes. The domain on which the PDE has to be solved is discretized into a mesh of finite elements, and the PDE itself is transformed into a set of linear equations defined on these elements [1], which can then be solved by iterative methods such as Conjugate Gradient (CG). Due to the very large amount of elements needed to obtain an accurate approximation of the original problem, this method became a classical application for parallel computers. The parallelization of numerical simulation algorithms usually follows the Single-Program Multiple-Data (SPMD) paradigm: Each processor executes the same code on a different part of the data. This means that the mesh has to be split into P subdomains (where P is the number of processors) and each subdomain is then assigned to one of the processors. Since iterative solution algorithms mainly perform local operations, i. e. data dependencies are defined by the mesh, the parallel algorithm only requires communication at the partition boundaries. Hence, the efficiency depends on two factors: An equal distribution of the data (work load) on the processors and

* This work was partly supported by the German Science Foundation (DFG) project SFB-376 and by the IST Program of the EU under contract number IST-1999-14186 (ALCOM-FT).



A 2D FEM-mesh



A partitioning into 5 parts

Fig. 1. Example: Applying a library to partition the 2D FEM-mesh "biplane.9" into 5 parts

a small communication overhead achieved by minimizing the number of edges between different partitions.

In practice, mainly two distinct approaches are applied to take care of this problem: Advanced partitioning tools based on sometimes quite complicated heuristics and more simplistic methods based on geometric approaches. Comparisons between these approaches have been undertaken and results are presented for example in [2, 3]. However, since these publications consider a large number of partitioning approaches, the presentation of the results are somewhat comprehensive.

Especially space-filling curves, one of the geometric approaches, have not been compared extensively to other methods yet. To better understand what their advantages and disadvantages over elaborated heuristics like multilevel methods are, we will present more detailed results here. The rest of this paper is organized as follows: In the next section we will give a brief overview of the two graph partitioning approaches compared in this paper. In section 3, we define the types of space-filling curves used for our evaluations and also present some of their properties. Section 4 shows how we performed the experiments. The results are presented in section 5.

2 Related Work

Because the graph partitioning problem is known to be NP-complete, a number of heuristics have been developed and implemented in several graph partitioning libraries like Metis [4], Jostle [5], Chaco [6] or Party [7, 8]. They usually follow the multilevel approach: In every level vertices of the graph are matched and a new, smaller graph with a similar structure is generated, until only a small graph, sometimes only with P vertices, is left. The partitioning problem is then solved for this small graph and vertices in higher levels are partitioned according to their representatives in lower levels. Additionally, to improve the partition quality a local refinement phase is applied in every level. In most cases, this refinement is based on the Fiduccia-Mattheyses method [9], a run-time optimized version of the Kerninghan-Lin (KL) algorithm [10]. However, also the Helpful-Set method [11] has been shown to produce good results. But since its current implementation in Party is designed (and reliable) for bisection only, we will restrict our comparison to Metis which uses a KL like algorithm. Figure 1 shows an FEM graph and its partitioning into 5 parts computed with Metis.

Another, widely applied approach to partition a mesh are geometric methods. The one we consider in this paper is based on space-filling curves. The vertices of the FE mesh are sorted by a certain recursive scheme covering the whole domain. Then, the now linear array of vertices is split into equal sized parts, each representing a partition. In contrast to partitioning heuristics, this method only works if vertex coordinates are present. It is also clear, that the quality of the generated partitioning does not excel the quality of the former mentioned elaborated heuristics since except for the coordinates any other information provided by the graph is simply ignored. This can especially be observed if the FE domain has holes which are not handled well by techniques covering the whole coordinate space. On the other hand, not relying on any other information than coordinates can be seen as a big advantage because memory requirements and run-time do decrease a lot. Furthermore, these kinds of algorithms are relatively easy to implement and provide information useful for realizing cache efficient calculations. Therefore, they are often closely coupled with the FE application. Since different kinds of space-filling curves exist, the ones used here are defined in the next section.

3 Partitioning with Space-Filling Curves

Space-filling curves are geometric representations of bijective mappings $M : \{1, \ldots, N^m\} \rightarrow \{1, \ldots, N\}^m$. The curve M traverses all N^m cells in the m-dimensional grid of size N. They have been introduced by Peano and Hilbert in the late 19$^{\text{th}}$ century [12]. An (historic) overview on space-filling curves is given in [13].

3.1 Computation of Space-Filling Curves

In figures 2-4, the recursive construction of space-filling curves is illustrated exemplarily for the Hilbert curve. Figure 2 shows the refinement rule splitting each quadrant into four subparts. The order within the quadrants has the same u-like basic pattern with some subparts being reflected and rotated. A possible algorithm calculating this refinement is sketched in figure 4. The HILBERT-function separates all given nodes in the interval [*first*, *last*[into four sections and processes them recursively. The separators for the four sections are *firstcut*, *midcut*,

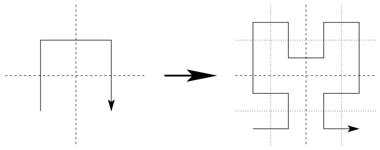


Fig. 2. Refinement rule of the Hilbert curve.

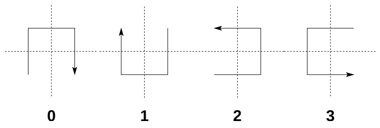


Fig. 3. Enumeration of basic pattern types of the Hilbert curve.

```
HILBERT (first, last, type):

    orient1 = (type < 2) ? x : y
    orient2 = (type < 2) ? y : x
    dir1 = (type%2==0) ? ascend : descend
    dir2 = (type%2==0) ? descend : ascend

    midcut = SPLIT (dir1, orient1, first, last)
    firstcut = SPLIT (dir2, orient1, first, midcut)
    thirdcut = SPLIT (dir2, orient2, midcut, last)

    HILBERT (first, firstcut, (type+2)%4)
    HILBERT (firstcut, midcut, type)
    HILBERT (midcut, thirdcut, type)
    HILBERT (thirdcut, last, 3−type)
```

Fig. 4. Algorithm sketch for the recursive calculation of the Hilbert curve.

and *thirdcut*. The SPLIT-operation sorts all nodes in the specified interval according to an *orientation* (x- or y-axis) and a *direction* (ascend or descend), returning the index which represents the geometric separator. The orientation and direction are determined from the basic pattern *type* of the Hilbert curve to be generated. The given algorithm fits to the enumeration of pattern types printed in figure 3.

An overview on the indexing schemes evaluated in this paper is plotted in figure 7 (top row). In addition to the older indexing schemes of Hilbert, Lebesgue, and Sierpiński we have examined the $\beta\Omega$-indexing. The refinement rules for all curves can be found in [13, 14].

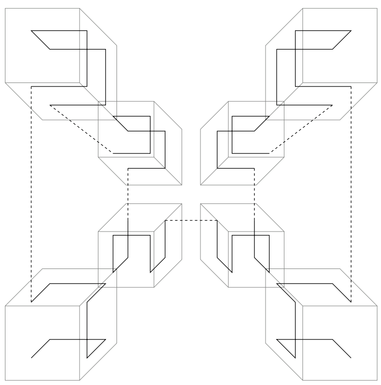


Fig. 5. 3-dimensional Hilbert curve used for evaluations.

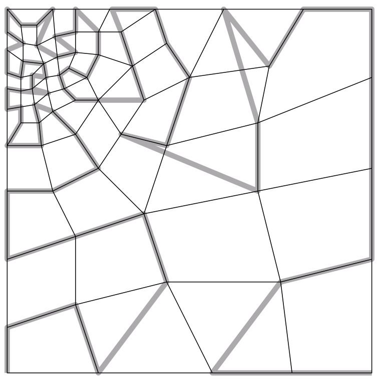


Fig. 6. Hilbert order within an irregular graph.

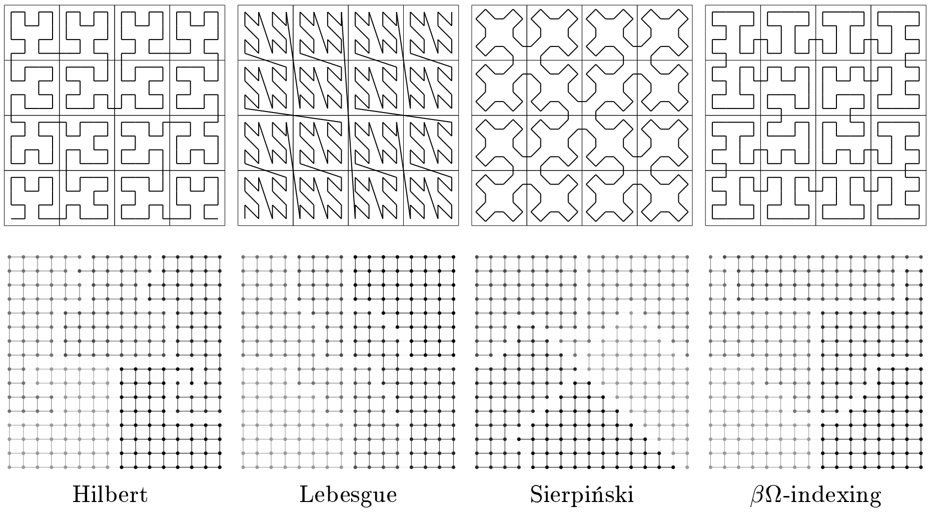


Fig. 7. The four evaluated curves. Top: Structure after 4 refinement steps. Bottom: The induced partitionings on a 16×16-grid.

While the extension to 3-dimensional space is obvious and unique for the Lebesgue curve, there are 1536 structurally different possibilities for curves with the Hilbert property [15]. In this paper the evaluations are based on the 3-dimensional definition sketched in figure 5 showing two refinement steps. For Sierpiński and $\beta\Omega$-indexing, no 3-dimensional versions are known.

3.2 An Example: Partitioning the 16×16-Grid

The bottom row of figure 7 shows the partitioning of a 16×16-grid into 5 parts using space-filling curves. The edge cuts for the four indexing schemes are 65 (Hilbert), 64 (Lebesgue), 66 (Sierpiński), and 58 ($\beta\Omega$-indexing). For comparison, the edge cuts obtained with Metis are 63 (kmetis, direct k-partitioning) and 46 (pmetis, recursive partitioning). Thus, the edge cut for the partitionings based on the indexing schemes is 26 % to 44 % higher than the one of the solution computed by pmetis.

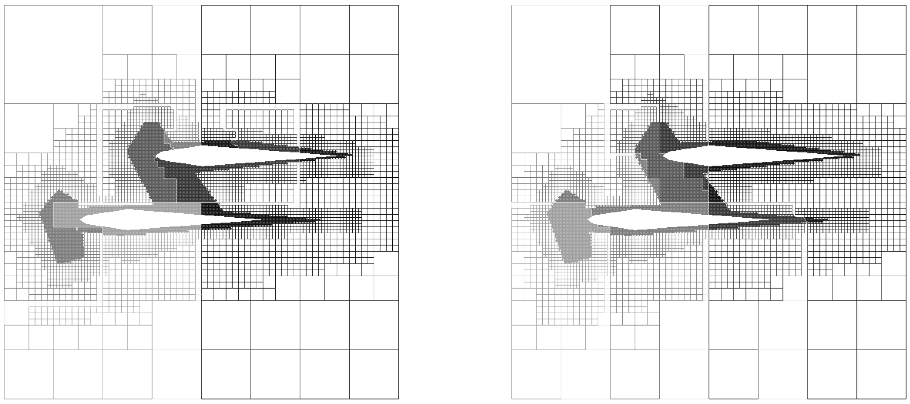
Although this example is not representative for the overall quality (especially for irregular graphs), it shows some of the specific disadvantages of indexing schemes. In case of the Hilbert curve, the endings of the partitions are sometimes spiral. The partitions induced by the Lebesgue curve are not connected even in regular grids.[3] On regular grids, the Sierpiński curve shows a weakness, since here the diagonal geometric separators lead to a high edge cut during recursive

[3] None of the indexing schemes can guarantee connected partitions in irregular graphs, but the probability of disconnected partitions is much higher for the Lebesgue curve for all graphs.

construction, even if the partitions are quite compact. Furthermore, the endings of the partitions are sometimes of a slightly spiral structure. The $\beta\Omega$-indexing is based on the same u-like base pattern as the Hilbert curve but uses different refinement rules to reduce the spiral effects at the partition endings.

3.3 Partitioning Irregular Graphs

For the indexing of the vertices of irregular graphs the space is split recursively until each subsquare (or subcube) contains of at most one vertex. The order of the vertices is given by the order of the subspaces. The Hilbert curve for an irregular graph is presented in figure 6. In the regular grid, the curves of Hilbert, Sierpiński, and the $\beta\Omega$-indexing only connects vertices which are connected in the graph. This observation does not hold for an irregular graph.

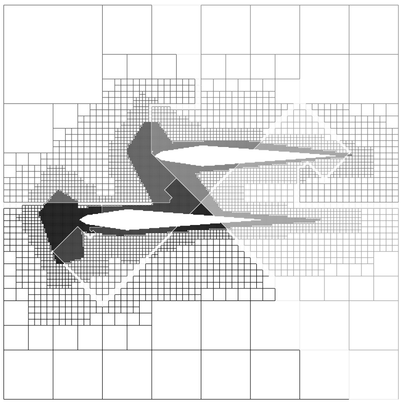


Hilbert Lebesgue

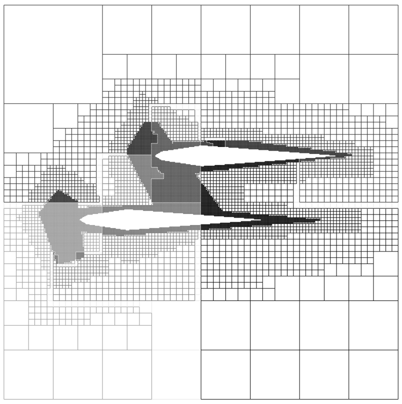
Fig. 8. Partitioning example for *biplane.9* using space-filling curves.

Figures 8 and 9 show the partitionings of the larger irregular graph "biplane.9" (cf. table 1 in section 4) into five parts using the evaluated indexing schemes. The resulting edge cuts are 627 (Hilbert), 611 (Lebesgue), 863 (Sierpiński), and 615 ($\beta\Omega$-indexing). For comparison, the edge cuts obtained using Metis are 299 (kmetis, cf. figure 1) and 302 (pmetis).

Due to the holes in the graph, space-filling curves and all other geometric partitioning heuristics may lead to disconnected partitions. Spiral effects can be observed again for the Hilbert and Sierpiński based partitioning and in a slightly reduced form for the $\beta\Omega$-indexing. On the other hand, for the Lebesgue curve there is a larger number of disconnected partitions. The Sierpiński curve results in the worst partitioning because its recursive definition is based on triangles which fits badly to a graph dominated by axis aligned edges.



Sierpiński



$\beta\Omega$-indexing

Fig. 9. Partitioning example for *biplane.9* using space-filling curves.

3.4 Analytical Results

In [16, 17], it is shown that the partitioning based on connected space-filling curves is "quasi optimal" for regular grids and special types of adaptively refined grids:

$$\text{edge cut} \leq C \cdot \left(\frac{|V|}{P}\right)^{(d-1)/d} \quad , \tag{1}$$

where $|V|$ denotes the number of vertices, P the number of partitions, and d the dimension of the graph. The constant C depends on the type of the curve.

Some constants have been determined for 2-dimensional regular grids in worst case [18]. The quality of a partition based on the Lebesgue curve is bounded by

$$7.348 < 3 \cdot \sqrt{6} - \varepsilon \leq C_{\max}^{\text{Lebesgue}} \leq \frac{384}{\sqrt{2730}} < 7.350 \ . \tag{2}$$

A lower bound for the Hilbert curve is

$$7.442 < 12 \cdot \sqrt{\frac{5}{13}} \leq C_{\max}^{\text{Hilbert}} \ . \tag{3}$$

It follows that the Lebesgue curve is better than the Hilbert curve in worst case analysis for the regular grid, despite its disconnected partitions. Combined with the fact that the Sierpiński curve and the $\beta\Omega$-indexing have larger lower bounds, Lebesgue turns out to be the best of the four evaluated indexing schemes in this case.

In average case, partitions based on the Lebesgue curve are bounded by

$$C_{\text{avg}}^{\text{Lebesgue}} \leq \frac{10}{\sqrt{3}} < 5.774 \ . \tag{4}$$

Upper bounds for the Hilbert curve (5.56) and the $\beta\Omega$-indexing (5.48) can be extracted from experimental results [14]. Compared to the optimal partition, a square with a boundary of $4 \cdot \sqrt{|V|/P}$, the decrease in quality is about 85 % in worst case and 40 % in average case.

4 The Test Environment

4.1 Used Metrics

Several metrics have been proposed to value the quality of results provided by partitioning algorithms. The first and probably most common one is the *edge cut*. Given a graph $G = (V, E)$ and a partitioning $\pi : V \to P$, the edge cut is defined straight forward as

$$\text{edge cut} = |\{(u, v) \in E \mid \pi(u) \neq \pi(v)\}|$$

As described in [19], this metrics has some flaws when applied to FEM partitioning, since it does not model the real communication costs. Therefore, a more exact metrics can be obtained by counting the number of *boundary vertices*, that is those vertices connected via an edge with a vertex from a different partition

$$\text{boundary vertices} = |\{v \in V \mid \exists u \in V : (u, v) \in E \wedge \pi(u) \neq \pi(v)\}|$$

In practice however, the edge cut is still the most widely used metrics. Furthermore, the results obtained for the boundary metrics and other metrics are very similar to the ones obtained based on the edge cut metrics. Thus, we will restrict out presentations to the latter one.

Another important factor for load balancing is the amount of required resources, namely time and memory. Since the goal of the load balancing process is to reduce the overall computation time, it is important that it does only consume very little time itself. To measure time and memory, we implemented a more precise clock and a memory counter into Metis 4.0. The overhead produced thereby has been tested to be negligible. All experiments have been performed on a Pentium III 850 MHz, 512 MB RAM system.

4.2 Evaluated Graphs

As usual for heuristics the obtained results also depend on the input instances. For our experiments, we used a set of established FEM graphs that can be obtained via the Internet [20] and have already been used in other work [5, 7, 21]. Table 1 lists the graphs that we used in this paper to describe our results in more detail.

graph	$\|V\|$	$\|E\|$	$\deg_{\min}$	$\deg_{\text{avg}}$	$\deg_{\max}$	Comments
grid100x100	10000	19600	2	3.96	4	2-dim. grid
airfoil1	4253	12289	3	5.78	9	2-dim. triangle FEM (holes)
biplane.9	21701	42038	2	3.87	4	2-dim. square FEM (holes)
stufe.10	24010	46414	2	3.87	4	2-dim. square FEM
shock.9	36476	71290	2	3.91	4	2-dim. square FEM
dime20	224843	336024	2	2.99	3	2-dim. dual FEM (holes)
pwt	36519	144794	0	7.93	15	3-dim. FEM
ara	62032	121544	2	3.92	4	3-dim. FEM
rotor	99617	662431	5	13.30	125	3-dim. FEM
wave	156317	1059331	3	13.55	44	3-dim. FEM
hermes_all	320194	3722641	4	23.25	56	3-dim. FEM

Table 1. Graphs used in our experiments.

5 Experimental Results

5.1 Quality

The first graph we chose for out test set is a 100×100-grid. Figure 10 displays the edge cut obtained with kmetis and the four types of space-filling curves described in section 3.

With an increasing number of partitions the total edge cut also rises in all cases. The cut size calculated with Metis is the smallest, with some exceptions at powers of 2 where space-filling curves fit perfectly into a grid, followed by $\beta\Omega$-Indexing, Hilbert, Lebesgue and then Sierpinski. The interesting aspect in this figure is, that the gap between Metis and the space-filling curves increases, but on the other hand the relative difference decreases.

This becomes more obvious if the cut sizes achieved with space-filling curves are shown in relation to the ones obtained with Metis (figure 11). Starting with bisection, the results produced with Metis are up to almost twice as good as those obtained with space-filling curves. This head start decreases more or less constantly down to a factor of 1.3, where at the already mentioned powers of 2 space-filling curves are approximately 10 percent better than Metis. Among the space-filling curves the $\beta\Omega$-indexing and Hilbert perform best while Sierpiński and Lebesgue produce partitions with a slightly higher edge cut.

Unfortunately, since the partitioning problem is NP-complete, no optimal solutions are known for large graphs. Thus, if the difference between both methods decreases, it is an open question whether this is due to an improvement of the partitionings induced by space-filling curves or a quality reduction of the results obtained with Metis.

Another way of normalizing the results is shown in figure 12. Here, the obtained edge cut is plotted in relation to the number of vertices of a partition as described in equation 1. As shown in equation 4, the theoretical upper bound of a partitioning induced by the Lebesgue curve is about 5.8 in the average case.

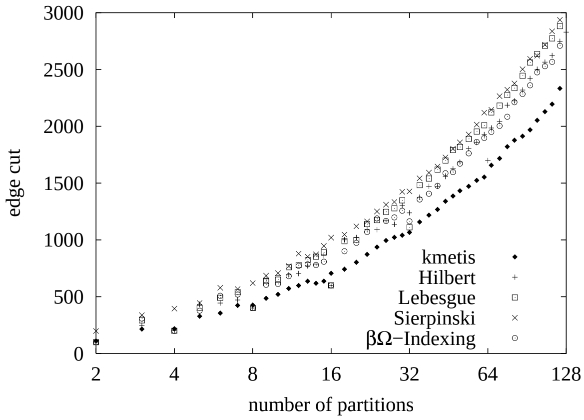


Fig. 10. Total edgecut obtained for "grid100x100" using different space-filling curves

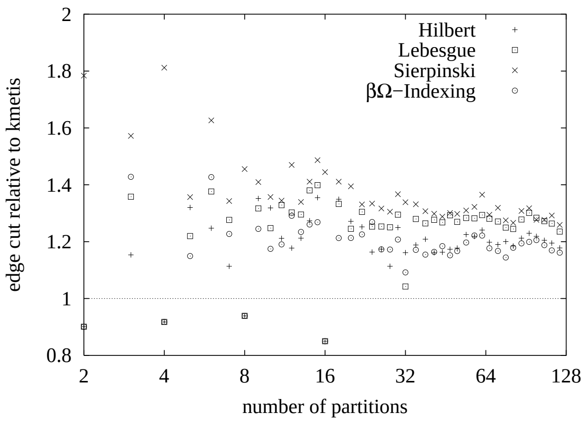


Fig. 11. "grid100x100": Quality of partitionings compared to kmetis.

This bound also holds in this experiment where results get closer to it for higher numbers of partitions.

Figure 13 shows the results for another 2-dimensional graph. In contrast to the grid, the "dime20" consists of an irregular structure and also contains two holes. Therefore, we do not expect as good results as the ones obtained for the grid. While this expectation turned out to be true, this is mainly the case for a small number of partitions. Compared to the grid, the head start of Metis decreases even more with an increasing number of partitions, reaching a factor of 2 in the interesting range of partition sizes. Furthermore, among the space-filling curves the Sierpiński curve performs best for this graph, followed by the $\beta\Omega$-indexing, the Hilbert curve and then Lebesgue, which produces an about 15 percent worse edge cut than the Sierpiński curve.

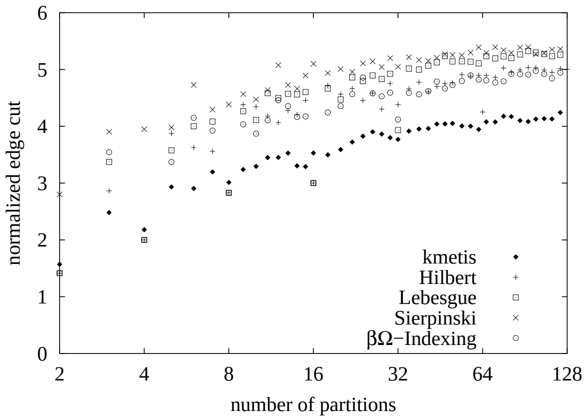


Fig. 12. Edge cut of partitions normalized to volume of partitions ($\sqrt{|V|/P}$).

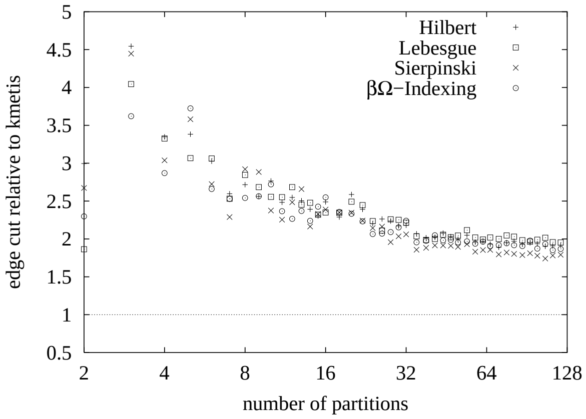


Fig. 13. Quality of partitionings compared to Metis for graph "dime20".

In figure 14, more results obtained applying the Hilbert curve on the other 2-dimensional graphs from table 1 are presented. The same observations made for the "dime20" graph can also be made here. For a small number of partitions Metis outperforms the space-filling curves by quite a large factor, this time ranging up to 3.5. But with an increasing number of partitions, the edge cut obtained using space-filling curves gets closer to the one calculated by Metis. For the graphs included here, a factor of less than 1.5 is reached.

The 3-dimensional extensions of the Hilbert and Lebesgue curves show a similar behavior as their 2-dimensional counterparts. Figure 15 gives the results obtained for the "pwt" graph. While the overall picture is similar to the 2-dimensional ones, the difference between both curves is much larger, with the Hilbert curve producing an up to 7 times worse result than Metis. On the other

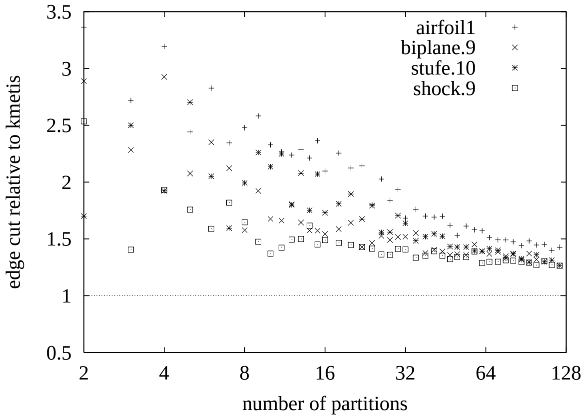


Fig. 14. Quality of partitionings compared to Metis for different 2d graphs.

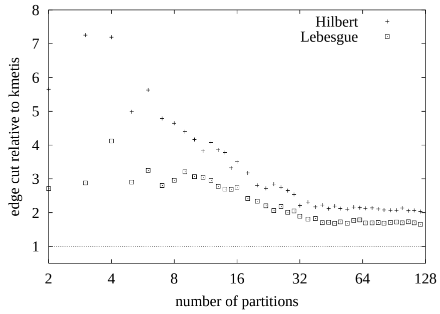


Fig. 15. Quality of partitionings of graph "pwt" compared to Metis.

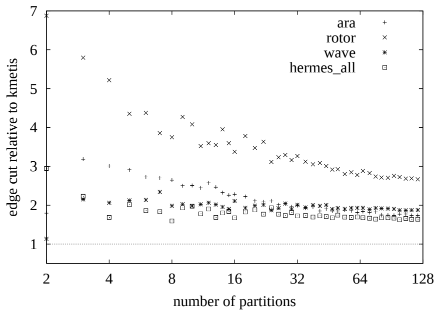


Fig. 16. Quality of partitionings compared to Metis for different 3d graphs using the Hilbert curve.

hand, the Lebesgue scheme does start with only a factor of about 3. Nevertheless, if more then 32 partitions are desired this factor decreases down to 2.2 and 1.8, respectively.

We combined the results from the experiments with the graphs "ara", "rotor", "wave", and "hermes_all" in figure 16, displaying the solution quality obtained by using the Hilbert scheme. The space-filling curves perform quite well again, producing edge cuts less than twice as large as those from Metis from 16 partitions on. An exception to this is the "rotor" graph, where only a factor of 3 to 4 can be achieved with the Hilbert scheme.

Tables 2 and 3 summarize our observations for 16 and 64 partitions, respectively. In the 2-dimensional case and for 16 partitions the Lebesgue curve produces the best results for our graphs. Thus, the discontinued structure of the Lebesgue curve described in 3 results in better partitionings than the connected,

graph	kmetis	pmetis	Hilbert	Lebesgue	Sierpiński	$\beta\Omega$-indexing
grid100x100	660	706	**600**	**600**	1020	**600**
airfoil1	555	574	1204	1215	**1125**	1276
biplane.9	800	812	1253	**1235**	1593	1285
stufe.10	759	723	1251	**1245**	1503	1389
shock.9	1208	1233	1837	**1675**	2050	1736
dime20	1311	1330	3310	**3125**	3176	3390
pwt	2992	2933	10281	**8073**		
ara	4652	4666	10602	**9914**		
rotor	24477	23863	**80523**	96889		
wave	48183	48106	101493	**83661**		
hermes_all	119219	119170	**199542**	256865		

Table 2. Edge cut obtained for 16 partitions.

graph	kmetis	pmetis	Hilbert	Lebesgue	Sierpiński	$\beta\Omega$-indexing
grid100x100	1543	1599	**1699**	1713	2141	1748
airfoil1	1528	1572	2430	2501	**2265**	2408
biplane.9	1906	2023	2867	2888	3229	**2844**
stufe.10	2268	2303	3286	**3036**	3087	3427
shock.9	2902	2889	3873	3915	4355	**3761**
dime20	3655	3670	7465	7510	**6846**	7419
pwt	9015	9310	19458	**15767**		
ara	9034	9405	16546	**16378**		
rotor	52190	53623	**151820**	184376		
wave	94342	97010	180764	**141013**		
hermes_all	241771	249959	**420313**	459234		

Table 3. Edge cut obtained for 64 partitions.

but spiral ones of e.g. the Hilbert scheme. However, in the case of 64 partitions this advantage diminishes and no curve is clearly superior. Due to the different structure of the Sierpiński curve, it either produces very good or very bad results compared to the other curves (e.g. on graphs "airfoil1" and "biplane.9"). The edge cut of the partitionings induced by space-filling curves is about twice as large as the one obtained with Metis. For 64 partitions however, this value decreases down to a factor of 1.6. For 3-dimensional graphs, no space-filling curve performs clearly better than the other, neither for 16 nor for 64 partitions. Compared to Metis, the relative difference here is 2.5 and 2.1 for partition numbers of 16 and 64, respectively.

5.2 Resources

As mentioned before, the goal of load balancing is the reduction of the overall computation time. Therefore, the time spend on partitioning itself should also be minimized. Figure 17 shows the results of our experiments performed on the "dime20" graph. All space-filling curves need much less time for their computation than either kmetis or even the recursive partitioner pmetis. This is even

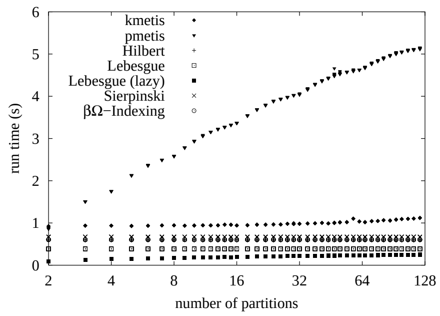


Fig. 17. Run-time of space-filling curves and Metis needed to partition "dime20".

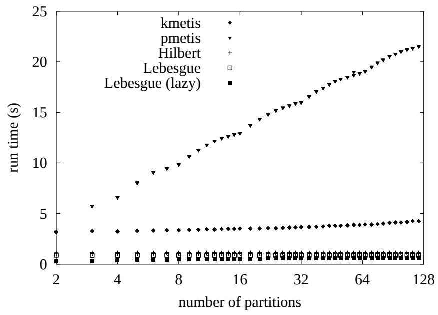


Fig. 18. Run-time of space-filling curves and Metis needed to partition "hermes_all".

more the case if the computation of the ordering is interrupted as soon as all partitionings have been determined, listed as *lazy* in plots 17 and 18. On the other hand, if full ordering information is available, it is easy to decompose the graph into any other given number of partitions very quickly.

Considering the memory consumption, Metis is outperformed even more by all space-filling curves. In case of the "dime20" graph, Metis requires about 42 MByte whereas space-filling curves only consumes 3.5 MByte for the graph and additional 2 KByte for the recursive descending. Partitioning the "hermes_all" graph, this gap even widens to 220 MByte vs. 5 MByte.

6 Conclusions

As expected Metis produces better results concerning the edge cut than space-filling curves do. This is not surprising since space-filling curves only rely on vertex coordinates rather than on any connectivity information between the vertices; the information that edges do provide is simply ignored. However, the gap between the solution quality of both approaches is not too large. In most cases, applying Metis does not result in more than a 30 to 50 percent decrease in edge cut for a decent number of partitions. This factor decreases further with an increasing number of partitions. Moreover, space-filling curves save both, a lot of time and a lot of memory.

Finally, it depends on the application which method is suited best: If memory is a concern, space-filling curves are definitely superior. Looking at run-time we can say, that if some additional communication overhead does not slow down the application too much, space-filling curves have to be considered as a partitioning alternative.

References

1. G. Fox, R. Williams, and P. Messina. *Parallel Computing Works!* Morgan Kaufmann, San Francisco, 1994.
2. Bruce Hendrickson and Karen Devine. Dynamic load balancing in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 184:485–500, 2000.
3. K. Schloegel, G. Karypis, and V. Kumar. Graph partitioning for high performance scientific simulations. In J. Dongarra et al., editor, The Sourcebook of Parallel Computing. Morgan Kaufmann, 2002. to appear.
4. George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.
5. C. Walshaw and M. Cross. Mesh partitioning: A multilevel balancing and refinement algorithm. *SIJSSC: SIAM Journal on Scientific and Statistical Computing, apparently renamed SIAM Journal on Scientific Computing*, 22, 2000.
6. B. Hendrickson and R. Leland. The chaco user's guide — version 2.0, 1994.
7. R. Preis and R. Diekmann. Party - a software library for graph partitioning. *Advances in Computational Mechanics with Parallel and Distributed Processing*, pages 63–71, 1997.
8. Robert Preis. The PARTY graphpartitioning-library – user manual – version 1.99.
9. C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. In *Design Automation Conference*, May 1984.
10. B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, February 1970.
11. R. Diekmann, B. Monien, and R. Preis. Using helpful sets to improve graph bisections. In *Interconnection Networks and Mapping and Scheduling Parallel Computations*, volume 21 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 57–73. AMS Publications, 1995.
12. David Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38:459–460, 1891.
13. H. Sagan. *Space Filling Curves*. Springer, 1994.
14. Jens-Michael Wierum. Definition of a new circular space-filling curve – $\beta\Omega$-indexing. Technical Report TR-001-02, Paderborn Center for Parallel Computing, http://www.upb.de/pc2/, 2002.
15. Jochen Alber and Rolf Niedermeier. On multi-dimensional hilbert indexings. In *Computing and Combinatorics*, number 1449 in LNCS, pages 329–338, 1998.
16. Gerhard Zumbusch. On the quality of space-filling curve induced partitions. *Zeitschrift für Angewandte Mathematik und Mechanik*, 81, SUPP/1:25–28, 2001.
17. Gerhard Zumbusch. Load balancing for adaptivly refined grids. Technical Report 722, SFB 256, University Bonn, 2001.
18. Jan Hungershöfer and Jens-Michael Wierum. On the quality of partitions based on space-filling curves. In *International Conference on Computational Science ICCS*, volume 2331 of *LNCS*, pages 36–45. Springer, 2002.
19. Bruce Hendrickson and Tamara G. Kolda. Graph partitioning models for parallel computing. *Parallel Computing*, 26(12):1519–1534, 2000.
20. C. Walshaw. The graph partitioning archive. http://www.gre.ac.uk/ c.walshaw/partition/.
21. R. Battiti and A. Bertossi. Greedy, prohibition, and reactive heuristics for graph partitioning. *IEEE Transactions on Computers*, 48(4):361–385, April 1999.