25th International Congress on Modelling and Simulation, Darwin, NT, Australia, 9 to 14 July 2023
mssanz.org.au/modsim2023

Efficiently computing DGGS cell representations of spatial geometries

Ross Petridis, Jonathan Yu, Ben Leighton and Simon Cox

CSIRO Environment, Clayton, Victoria, Australia
Email: ross.petridis@csiro.au

Abstract: Traditional analysis of geospatial data is limited to those with expertise in Geographic Information Systems (GISs) and access to GIS tools. Furthermore, these systems require computationally demanding mathematical operations which creates an increasingly significant computational barrier to geospatial analysis. Especially so as modern geospatial practices occur at a higher volume with more detailed geometries. However, many complex operations can be replaced with simpler set theory operations. This is achieved with Discrete Global Grid Systems (DGGS). A DGGS tessellates the earth with cells (and their unique identifiers) of a predefined and arbitrarily small size, therefore facilitating the representation of geospatial regions from sets of cell identifiers to any desired resolution. When a vector formatted geospatial polygon is mapped onto a DGGS grid and represented by a set of cells, simpler set theory can be used between it and other geospatial regions also represented by a set of cells. Topological relationships can be inferred from these sets of cell identifiers alone as each are uniquely identified across the tessellating grid and between cell levels. This paper concerns the rHEALPIX DGGS (Gibb, 2016), however, results can be generalized to computing cell representations in other DGGSs.

A prerequisite to exploiting the efficiencies enabled by DGGS cell representations of geospatial regions is to first compute them from the traditionally vector formatted geometries. There are many algorithmic strategies to compute a DGGS cell representation of a vector formatted polygon, each with contrasting advantages and disadvantages. This paper investigates the optimal algorithmic strategy given a particular vector formatted polygon and their features. The investigation will involve a comparison between two contrasting algorithmic strategies that have been implemented using the rHEALPIX DGGS engine (Manaaki Whenua, 2020). The algorithms are run on a diverse set of test data for a range of desired resolutions. It was ultimately found that:

- For a geometry with a given ‘complexity’, there exists a desired DGGS resolution at which, for all resolutions finer, the top-down algorithms will be faster than the brute force point-in-poly algorithms, and for any resolution coarser, the point-in-poly brute force algorithms will be faster.
- For any query resolution, there exists a polygon with a ‘complexity’ for which any polygons more complex, the point-in-poly algorithms are faster than the top-down self-guiding algorithms, and any less complex, the top-down methods are superior.

In this paper, Koch Snowflakes of varying iterations were used to obtain a test dataset of varying ‘complexity’. Ultimately, this paper can provide readers with guidance towards selecting the optimal algorithm for their vector formatted geometry conversions into DGGS cells. This is useful as incorrect algorithm choice can result in exponentially more computation in the order of hours or days.

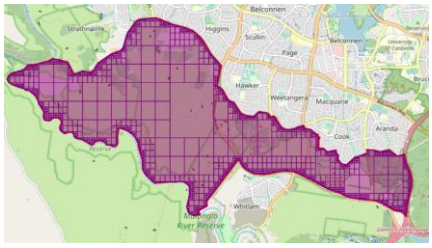
For future work, there are opportunities to refine the calculations of the ‘complexity’ metric for a geometry, as well as to quantify, perhaps heuristically, the ‘complexity’ at which one algorithm becomes better or worse than the other. Additionally, a machine learning model could be trained to predict which algorithm is better suited to convert a particular geometry given its geometric properties and the query resolution. This would only be useful insofar as the time to compute said geometric properties is insignificant to the time to compute the DGGS cell representation.

Keywords: *DGGS, rHEALPIX, geospatial, efficiency, algorithm*

1. INTRODUCTION

Traditional analysis of geospatial data is limited to users with expertise in Geographic Information Systems (GISs) and access to GIS tools. Such analysis can be computationally demanding as complex mathematical operations are required, the effects of which are amplified by modern geospatial practices of increased volume and resolution. However, using Discrete Global Grid Systems (DGGSs) such as the rHEALPIX DGGS (Gibb 2016), improvements in efficiency and accessibility of geospatial querying and analysis can be made as the requirement for traditional GIS tooling and intense computation is lessened.

By tessellating the earth with cells, DGGSs facilitate the cell-based representation of geospatial regions such as those depicted in Figure 1. However, where traditional rasters have a fixed precision (e.g., 5km x 5km cell), DGGSs support an arbitrary spatial precision through the potentially infinite subdivision of each cell into a set of sub cells that, when combined, represent the same region of the parent cell. Finer cells allow for more accurate representation of the original regions and each nested resolution is referred to as a 'level', with higher numbered levels for finer scales (Figure 2).



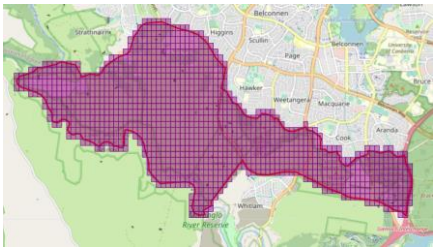


Figure 1a. (left) A fine hierarchical cell representation of a region in Canberra. **Figure 1b.** (right) A coarser constant cell-resolution representation of the same region.

Cells are identified using string identifiers (IDs). In the rHEALPIX DGGS, the first level (largest parent cells) are each defined by 6 letters: N, O, P, Q, R, S (Gibb 2016). Cells N and S contain the projection of the North and South caps, respectively. Meanwhile, O, P, Q, R, and S are assigned left to right (Figure. 3). From there, 9 children cells resulting from the subdivision of a parent cell are defined using the parent cell ID and appending an integer between 0 and 9 (for a subdivision by 9). Further children of these children are defined in a similar way, and there is no limit to this recursive subdivision which enables the definition of infinitely fine cells. This facilitates easy derivation of parent cells from sub cells by truncating a cell ID. For example, the parent of cell with ID "R142" is "R14". Similarly, the children of cell "R123" are cells with IDs "R123**1**" , "R123**2**", … , "R123**9**". Topological relationships can be inferred from sets of cell identifiers alone as each are uniquely identified across the grid and between cell levels.

Efficiency improvements in geospatial analysis are achieved after converting vector-based geometries/polygons into DGGS cell representations. This cell representation enables simpler set theory operations implemented as database look ups to perform queries such as "what bush fire burn regions intersect with what local government areas". Such database look ups oppose (the traditionally required) complex mathematical operations performed on the raw vector representations of polygons to answer the same question. Furthermore, cell counts to represent a geometry can be minimized to improve efficiency without loss of accuracy by utilizing coarser cells to represent larger open areas. See figure 1a vs figure 1b.

There are various implementations of DGGSs, with the key differences between them being the cell shape used to partition and tesselate the Earth. For example, the rHEALPIX DGGS (Gibb 2016) uses quadrilaterals of equal area, while Uber Engineering's H3 (2018) uses hexagons. Different shapes provide different computational advantages. This report concerns the rHEALPIX DGGS which is used in the standard Geoscience Australia DGGS (2016) services.

Resolution	Cell area (m^2)	Nominal cell edge	Number of cells
0	85.11E+12	9,225 km	6
1	9.456E+12	3,075 km	54
2	1.051E+12	1,025 km	486
..			
11	2,712	54 m	188.3E+09
12	301	18 m	1,694.6E+09
13	33.5	6 m	15,251E+09
14	3.72	2 m	137.26E+12

Figure 2. Approximate cell area, nominal cell edge length, and number of cells for a selection of grid resolutions of the Nside = 3 WGS84 ellipsoidal rHEALPIX grid hierarchy (Gibb 2016).

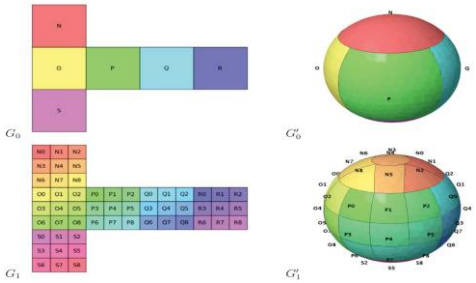


Figure 3. The first two tessellating cell grid levels for the entire globe (Gibb 2016).

Cell-based representations compare unfavorably to vector-based representations when attempting to capture the detail of complex regions. Fine resolutions may be required to satisfactorily represent geometries, which can result in high cell-counts, and exponentially larger computation times for most spatial applications. However, the use of cells of mixed levels significantly reduces the cell-count compared with a single resolution, so may allow much more efficient application support.

A prerequisite to harnessing the aforementioned efficiencies enabled by DGGS cell representations is the transformation of the (traditionally) vector format geospatial polygons into their DGGS cell representations. This paper will discuss approaches to this computation problem. There are currently two contrasting algorithmic strategies implemented on top of the rHEALPIX DGGS engine for computing the DGGS cell representations (to a desired cell level) of an input vector formatted polygon. It is hypothesized that each algorithm will favor different characteristics of the inputted polygons, for example, the size, shape, hole density, complexity of the boundaries and more. Firstly, the top-down method which can produce a hierarchical representation (Figure 1a). Secondly, the point-in-poly method which produces a constant cell level representation. There algorithm logic is illustrated in Figures 4 and 5, respectively.

The ability of the Top-down methods (Figure. 4) to terminate search early in regions where coarse cells can fit is advantageous because of a potentially more efficient generation of candidate cells, whereas the point-in-poly method is restricted to generating candidate cells at the finest desired resolution throughout the entire polygon. Meanwhile, the biggest advantage of the point-in-poly method (Figure. 5) is only requiring one predicate check per generated candidate cell, whereas the top-down methods must perform two. A predicate check is a question of "does cell intersect with poly?", or "is cell within poly?", which are needed to produce cell representations. Ultimately though, only the top-down methods can naturally output hierarchical (mixed cell resolution) representations without post processing, the point-in-poly methods are limited to producing cell ID sets with cells of a constant resolution. If a hierarchical set of cell IDs is required from the output of a point-in-poly approach, this would require additional post-processing time which is not taken into account for the timings discussed below. This process could be achieved by iteratively scanning through the set of cell IDs, and replacing a complete set of nine children by their one parent. Noting that by their definition, a set of 9 children cells represent the same spatial region as their 1 parent cell.

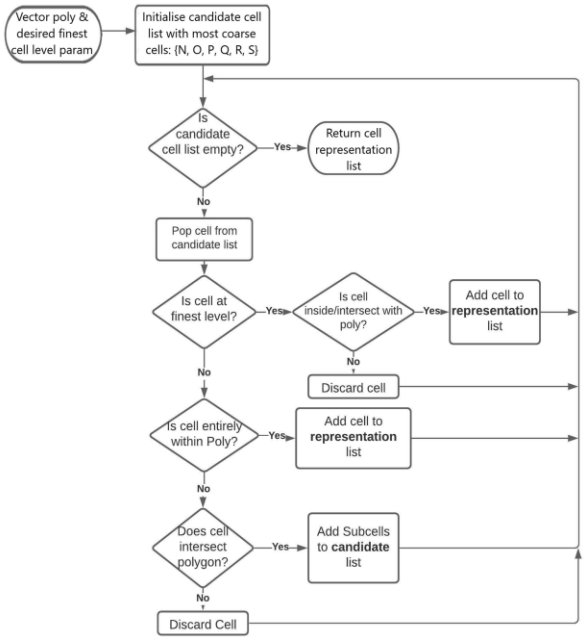


Figure 4. Top-down, self-guided algorithm.

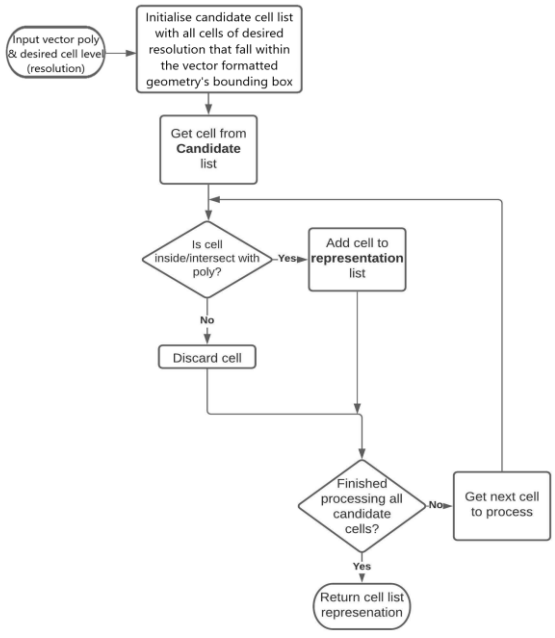


Figure 5. Point-in-poly, brute force style algorithm.

Conclusively, the following is hypothesized:

1. Top-down algorithms will outperform point-in-poly algorithms for transformations of geometries and resolutions that will allow for a high use of coarser cells. E.g., suburb border with large open areas.
2. Point-in-poly approach will favour geometries and resolutions that do not allow for many coarse cells in the representation. E.g.,, highly detailed interiors of polygons that require fine cells to be used everywhere to capture the detail.

2. METHODOLOGY AND ALGORITHMIC STRATEGIES OVERVIEW

To investigate the optimal method to compute DGGS cell representations, the performance of two algorithmic strategies will be analyzed. The first uses a top-down (self-guided) hierarchical decomposition which has been implemented by CSIRO (2022) and Surround (2020). The second is a Point-in-poly (exhaustive) approach, implemented by Geoscience Australia (2016) which outputs a constant cell level representation. See Figure 6 for a comparison of raw outputs.

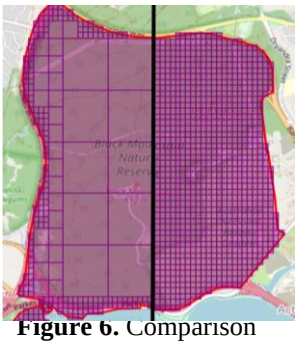


Figure 6. Comparison between hierarchical (Left) and constant resolution (Right) decompositions.

Algorithm performance will be measured using the time to compute the cell representations for test data containing a diverse set of spatial geometries. Additionally, the desired resolution will be altered to examine the scaling of each algorithm with respect to this parameter. Tests were run on a PC using an AMD Ryzen 7 4800HS CPU with typical speed of 3.8Ghz.

The following data sets containing geometries were utilized in testing.

1. The first 16 ABS Statistical Areas Level 2 (SA2) for the State of Victoria (ABS, 2021).
2. Koch Snowflakes (Fractals) of size 50km width and of iterations 3-7 inclusive (Hine, n.d.).

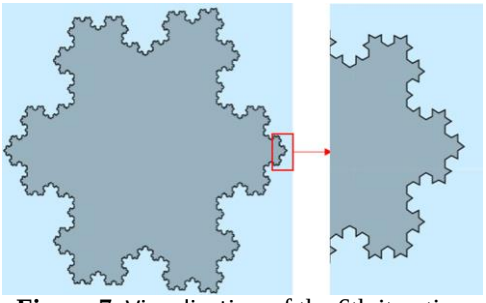


Figure 7. Visualization of the 6th iteration of a Koch Snowflake.

The first dataset of SA2s was chosen as a sample of realistically queried geometries. SA2s have a population between 3000-25000, so are approximately the size of suburbs or localities, but have a variety of different sized geometries, though with simple interiors. Testing with this dataset is intended to reveal if one algorithm is generally more efficient than another for these kinds of statistical geographies. The first 16 spatial geometries were used to aid with time and computing resource constraints, particularly for computing representations to high resolutions. The second data set contains Koch Snowflake Fractals of iterations 3-7 inclusive, see Figure 7 for an example. As iterations increase, Koch Snowflakes are defined with an increasing perimeter and number of points defining this perimeter - meanwhile, the area approaches a constant value. Therefore, this dataset was chosen to provide insight into how algorithmic performance scales with increasing border complexity.

3. RESULTS & DISCUSSION

3.1. SA2 Results

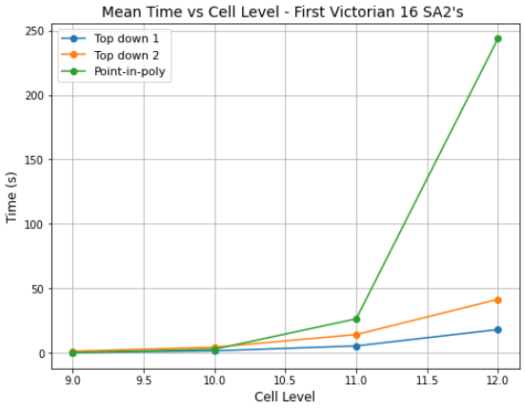


Figure 8. Mean Time to compute vs cell resolution.

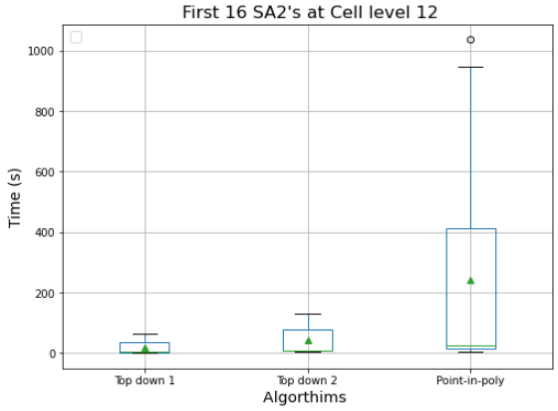


Figure 9. time to compute box plots per strategy.

Figure 8 presents the algorithm times to compute cell representations of the first 16 SA2's, plotted against the increasing resolution, or "Cell Level", (Petridis et al. 2022). Note that these geometries do not contain holes in interiors. It illustrates that the top-down (hierarchical) algorithms scaled more favorably (with respect to

increasing resolution) than the point-in-poly methods - albeit both scaled exponentially. This was likely a result of top-down methods requiring additional computation (for every increase in resolution) only around the perimeter of the geometries where it is necessary to capture additional detail of the polygon to the desired resolution. Whereas in the center of the polygons, the algorithm could terminate computation early as coarser cells were used to represent those larger areas. Meanwhile, the point-in-poly methods were constrained to calculating all cells at the finest resolution throughout the entire area of the polygons and not just the perimeters. A corollary to favorable scaling with increasing resolution is favorable scaling with increasing polygon input size. This is suggestable because a resolution is considered "fine" with respect to the size of the polygon. This notion is reflected in the boxplots of Figure 9 as the point-in-poly boxplot illustrates a larger spread of results compared to the top-down method, reflecting its performance being more sensitive to changes in area (size) than the top-down methods. Noting that there was a large variance in the areas for the first 16 SA2.

3.2. Koch Snowflakes (Fractals) Results

Whilst the top-down approaches scaled more favorably with increasing resolution resulting in them outperforming brute-force methods for increasing resolutions, there are some polygon features for which they scaled less favorably than the point-in-poly methods.

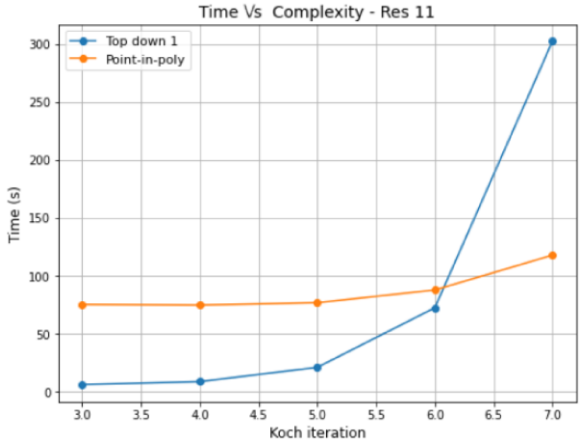


Figure 10. Koch snowflakes - time Vs 'complexity'

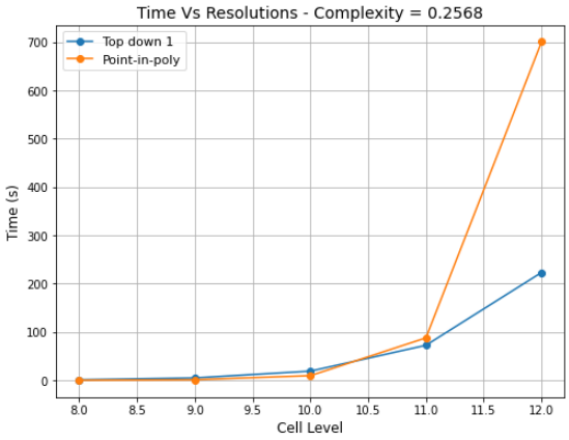


Figure 11. Koch Snowflakes – Time Vs Resolution

It was found that the point-in-poly approach scaled more favorably when tested against increasing input complexity and a constant input (area) size, see Figure 10. This is demonstrated by the tests run on a variety of different Koch Snowflake iterations of increasing border complexities. A likely reason is, for each candidate cell generated, the Top-down (hierarchical) methods must complete up to at least 2 predicate checks for cell containment or cell overlap with poly. Meanwhile, the brute force point-in-poly method only checks for containments once per candidate cell. For complex geometries like fractals with high 'complexities' along the perimeter, the portion of time spent completing predicate checks started to increase. As a result, because the top-down methods were completing more checks per cell, Fractals defined by an exponentially increasing number of points begun to favor point-in-poly approaches. This is especially so if the time taken per predicate check scales exponentially too as the number of points increases, potentially outweighing the benefits of a more efficient generation of candidate cells by the top-down methods since these methods require more predicate checks per cell. As Koch Snowflakes have large open areas allowing for good use of coarse cells by top-down algorithms yet are favored by the point-in-poly approaches as their complexity increases, this partly disproves the hypothesis that polygons that allow for the use of coarse cells with be favored by the top-down approaches. Ultimately, given a sufficiently complex perimeter, the point-in-poly approach computed the DGGS cell representation faster. However, keeping the perimeter complexity constant, an increasing cell representation resolution eventually favored the top-down strategy again (Figure. 11).

3.3. Complex Interiors

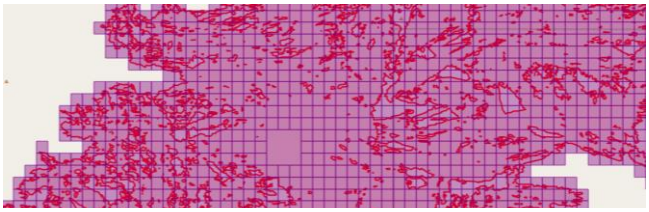


Figure 12. Geometry represented by a cell level that would be best computed via a point-in-poly method.

Furthermore, it makes logical sense that the point-in-poly methods are best suited to polygons with many holes and complex interiors (relative to the query resolution). Indeed, for the pictured geometry (Figure 12) represented using a resolution of 8, the point-in-poly methods will outperform the Top-down methods. This is due to minimal coarser cells being able to be utilized because of many small holes preventing perfect containment. Meanwhile, the top-down (hierarchical) method must still exhaustively attempt (unsuccessfully) to terminate search in an area earlier with coarser cells.

However, if the desired query resolution were increased enough, the top-down approaches would soon be able to leverage their ability to represent the geometries with coarser cells (than the desired cell-level resolutions) in the centre and eventually become the optimal algorithm again. This ultimately leads to the following conclusions:

1. For a geometry with given 'complexity', there exists a resolution at which, for all resolutions finer, the Top-down algorithms will be faster than the brute force point-in-poly algorithms. And for any resolution coarser, the point-in-poly brute force algorithms will be faster.
2. For any query resolution, there exists a polygon with a 'complexity' value for which any polygons more complex, the point-in-poly algorithms are faster than the top-down self-guiding algorithms, and any less complex, the top-down methods are superior.

3.4. Defining Complexity and future work

There are many ways to define the 'complexity' of an input geometry, or input transformation request including the desired resolution. Here, the Koch fractal iteration is used as a proxy for how complex the input geometry for cell-based representation is. However, complexity should also concern the interior of the polygons and the cell resolution as this can drastically affect the ability for top-down strategies to exploit their hierarchical decomposition and terminate computation early in certain areas of a polygon. Where there are many holes (relative to the resolution), the top-down methods waste a lot of time testing for coarser cell containment to only end up using the finest cells necessary to capture the detail, meanwhile the point-in-poly methods will begin at this level resulting in them not only producing a more efficient candidate cell list (as the top-down also test all resolutions above the target resolution) but also using less total predicate checks. The University of Connecticut (UoC) (n.d.) lists lots of different measures for properties of geospatial regions that could be combined to measure the overall 'complexity' or used in a decision process to predict the optimal algorithm choice. Therefore, in future, a complexity metric or decision process can be designed to quantify the crossover point (or boundary) more accurately at which one algorithm becomes more efficient than the other, as the complexity of the polygon or desired relative resolution changes. Such a metric could make use of multiple properties defined by UoC (n.d.) as well as including the resolution of the query. The properties could be calculated quickly (relative to entire computation time) on the fly prior to algorithm choice and combined via heuristic function or fed through a Machine Learning Model that has been trained to predict which algorithm is better suited to perform the DGGS cell transformation for a particular geometry given its characteristic properties and the desired cell resolution.

4. CONCLUSION

Ultimately, it was found that there is a use case for each algorithmic strategy as describe below.

- For a geometry with given 'complexity', there exists a resolution at which, for all resolutions finer, the top-down algorithms will be faster than the brute force point-in-poly algorithms. For any resolution coarser, the point-in-poly brute force algorithms will be faster to compute.
- For a particular query resolution, there exists a polygon 'complexity' for which polygons more complex are best converted by point-in-poly methods, and polygons less complex are best converted using top-down algorithms.

The top-down (hierarchical output) method for calculating DGGS cells for a geometry is not universally better than the point-in-poly method. This was shown in the case of the Koch Snowflakes test dataset, especially as their iteration increases. Where for each next iteration, the border is exponentially more complex, but area approaches a constant limit. With regards to geometries with many holes and coarser query resolutions preventing the use of coarse cells by hierarchical top-down algorithmic strategies, it is likely more efficient to use point-in-poly methods which will not waste computations attempting (and failing) to use coarse cells.

However, given a geometry of any complexity (including exterior and interior complexity), if the desired resolution for the transformation is increased enough, the top-down methods will be more efficient as result of their more efficient generation of candidate cells. A benefit of their ability to use coarser than the finest desired cell resolution to represent interiors.

Ultimately, the correct algorithm choice can be difficult as the fineness of a resolution is determined with respect to the size of a geometry and its features. Hence, future work could be to enhance the complexity metric ascribed to a geometry for transformation such that it considers its geometric properties more broadly as well as the desired resolution. Further, the training of a Machine Learning model could be used to predict which algorithm is better suited for a transformation given the desired query resolution and a list of features or attributes of the inputted polygon concerning its complexity and other geometric properties.

REFERENCES

Australian Bureau of Statistics (ABS), 2021, "ABS Maps", https://dbr.abs.gov.au/absmaps/index.html?xmin=15445593.186192056&ymin=-4777067.285358091&xmax=16867321.912296265&ymax=-3977841.717608328&bottomlayer=ASGS2021:STE&toplayer=ASGS2021:SA2. Accessed between 22/11/2021 and 14/01/2022.

CSIRO, 2022, "DGGSForPoly", GitHub, https://github.com/CSIRO-enviro-informatics/DGGSForPoly. Accessed between 22/11/2021 and 14/01/2022.

Geoscience Australia (GA) , 2020, "AusPIX: An Australian Government Implementation of the rHEALPIX DGGS in Python", GitHub, https://github.com/GeoscienceAustralia/AusPIX_DGGS. Accessed between 22/11/2021 and 14/01/2022.

Gibb, R.G., 2016, "The rHealPIX Discrete Global Grid System", Institute of Physics, Proceedings of the 9th Symposium of the International Society for Digital Earth (ISDE). Halifax, Nova Scotia, Canada. 5th-9th of October, 2015.

Jason Hine, (n.d.), "Koch Snowflake Polygon GeoJSON Generator", https://codepen.io/jhine/pen/LLpEEy. Accessed on 16/03/2023

Manaaki Whenua – Landcare Research, 2020, "rhealpixdggs-py", GitHub, https://github.com/manaakiwhenua/rhealpixdggs-py

Petridis, Ross; Cox, Simon; Yu, Jonathan; Leighton, Ben (2022): rHEALPIX Experimental Results for contrasting algorithms. v1. CSIRO. Data Collection. https://doi.org/10.25919/5h7w-7715. Accessed on 10/02/2022.

Surround, 2020, "CellZoneFromPoly", GitHub, https://github.com/manaakiwhenua/rhealpixdggs-py/blob/master/rhealpixdggs/conversion.py. Accessed between 22/11/2021 and 14/01/2022.

Uber Engineering, 2018, "H3: Uber's Hexagonal Hierarchical Spatial Index", https://eng.uber.com/h3/. Accessed between 22/11/2021 and 14/01/2022.

University of Connecticut (UoC), n.d., "Shape Metrics Tool", http://clear.uconn.edu/tools/Shape_Metrics/. Accessed on 10/02/2022.