frontiers

Supplementary Material: How Nonassociative Geometry Describes a Discrete Spacetime

1 TECHNICAL PRELIMINARIES

The code was written and executed using the *Mathematica 10* package on a 64-bit desktop PC with an i7-6700 processor and 16GB of RAM. *Mathematica* was used mainly for familiarity with its code and capabilities, and many built-in functions were also of use.

2 ALGORITHM

2.1 Preparation of initial vertices

Each instance of the algorithm begins with the same four initial points. These four vertices form an equilateral tetrahedron with geometric center at the origin and of sides l, with a vertex situated at the "north pole". Which vertex is selected as the north pole is initially arbitrary, but it must be maintained as such throughout the algorithm. The vertices are stored in the form of a list with coordinate triads, (r_i, θ_i, ϕ_i).

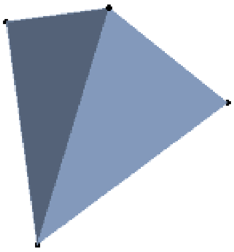

Figure S1: The initial vertex set define a symmetric tetrahedron centered at the origin, with a vertex at the coordinate corresponding to the north pole (here the figure is rotated to provide perspective).

2.2 Non-associative combining of vertices

The radius R functions as the independent variable in the algorithm. In the random-growth model, each new value for R is a random increase or decrease of one unit l per iteration, $R_{i+1} = R_i \pm l$, with the condition that R must always remain above the initial value (i.e., R behaves as a random walk with a barrier). Once a new radius is defined, it is used in the mapping of vertices to the complex plane such that vertices are turned to the form $\zeta = R\tan(\theta/2)e^{i\varphi}$. Thus, the vertices are now stored as a list of complex numbers rather than coordinate triads. Once in this form, each vertex is combined with all others using the non-associative operation

$$\zeta_{pq} = \frac{\zeta_p + \zeta_q}{1 - \zeta_p^* \zeta_q / R^2}. \tag{S1}$$

A second combination is performed after reversing the order of the list, ensuring that all combinations of $p - q$ and $q - p$ are carried out (the non-associative operation is also non-commutative). In some cases duplicate vertices can occur, and these are removed from the list. As preparation for the next step of the algorithm the new vertices are converted back to spherical coordinates. The coordinate angles for a new vertex ζ_{pq} arising from the combination of two other vertices, ζ_p and ζ_q, can be recovered from

$$\theta_{pq} = 2 \arctan \left[\frac{1}{R} \left| \frac{\zeta_p + \zeta_q}{1 - \zeta_p^* \zeta_q / R^2} \right| \right], \tag{S2}$$

and

$$\varphi_{pq} = \arg(\zeta_p + \zeta_q) - \frac{i}{2} \ln l(\zeta_p \zeta_q), \tag{S3}$$

where

$$l(\zeta_p \zeta_q) = \frac{1 - \zeta_p \zeta_q^* / R^2}{1 - \zeta_p^* \zeta_q / R^2}. \tag{S4}$$

2.3 Vertex deletion: Sorted versus Unsorted sets

After the fist step described above, we end up with both new and old vertices in the same set, just with a different radius. We propose to identify the variable l with the Planck length, and set it to $l = l_p = 1$. In addition to defining the step size in R, l works as a minimum allowed distance between any two vertices. The non-associative operation creates many more points than can fit into a spherical region complying with this length restriction. Therefore, we perform a purge of vertices from the list of old and newly generated points, starting with the northernmost point. This can be interpreted either as a fusion or a suppression of vertices that do not have enough space. Beginning with the north vertex ensures its preservation, which is important since the operation S1 is defined in reference to it.

We compare the distance of each vertex v_i to all others, one by one, and delete the vertices v_j that are at a Euclidean distance $||v_i - v_j|| < 1$ in 3D space. Then we move on to the next point on the list and repeat the procedure. As the list gets shorter each sweep progresses more quickly than the previous one, but this is still the most time-consuming phase of the algorithm.

We predicted that, having saturated the spherical surfaces with many vertices in each iteration, the average cell area would converge to that of the minimal cell, $\sim 0.433l = 0.433$. However, despite fitting hundreds of thousands of points into a radius of order l, the average area remained at approximately twice the expected value. A numerical experiment revealed that the order in which vertices were generated and fed into the deletion algorithm was not optimal for generating the smallest possible areas.

We can illustrate how this happens by generating many points randomly on a sphere and taking snapshots of the vertex deletion process at various points. When vertices were deleted in the same order as they where generated, we found a pattern as in figure S2. The vertices were produced in a chaotic fashion, so that their deletion proceeded chaotically as well. During the deletion, when a vertex v_i is reached and its neighbors are deleted if they lie below the allowed distance, it is left at the center of an otherwise empty region delimited by its closest surviving neighbors. However, if the neighbors of another vertex v_j that is close to v_i are checked later on, v_j's own deletion process may end up removing some of v_i's closest neighbors, leaving it surrounded by more distant vertices later on. The overall result is that the 'optimal' distribution of vertices over the sphere (and therefore the optimal tiling of the sphere with nearly equilateral triangular cells) is broken by the chaotic order of the deletion process.

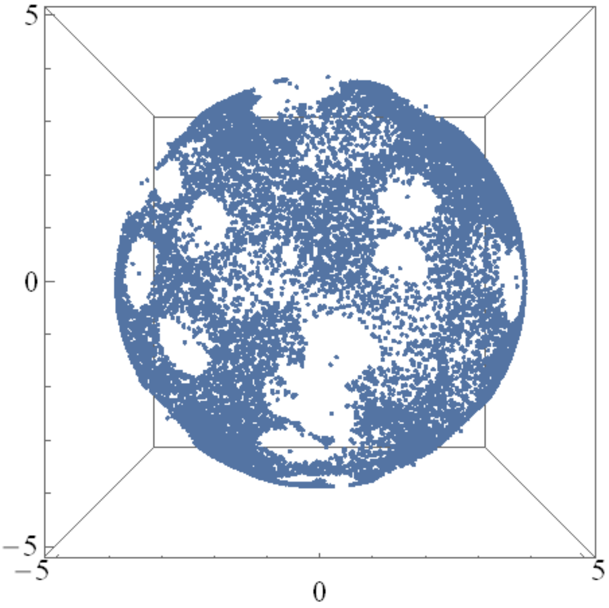

Figure S2: Chaotic deletion of vertices results in a sub-optimal final distribution. Here, the process is shown after 20,000 points have been deleted out of an initial set of 35,000.

If, however, we proceed to sort vertices by order of increasing (θ, ϕ) coordinates, and *then* perform the deletion, we get an optimal distribution of vertices, since the surviving vertices are effectively listed as closely as possible from top to bottom and left to right (i.e., spiraling down from the north of the sphere; see Figure S3). Convergence to the minimal area is then only a matter of saturating the sphere with enough points for a given radius, which the NAG operation achieves easily.

2.4 Repeat as necessary

Once a list of vertices is purged, it is guaranteed to have at least a distance l between each of its elements. At this point it can be pushed back to the beginning of the algorithm to serve as the next generator of vertices. The whole process will be repeated a predefined number of n iterations, with each iteration corresponding to an increase of one unit of Planck time. We call each post-purge set a "generation" of vertices, and for n iterations of the algorithm we will end up with $n + 1$ total generations. We call one full set of n iterations a "run" of the algorithm, and several runs can be performed to allow for statistical analysis of the process.

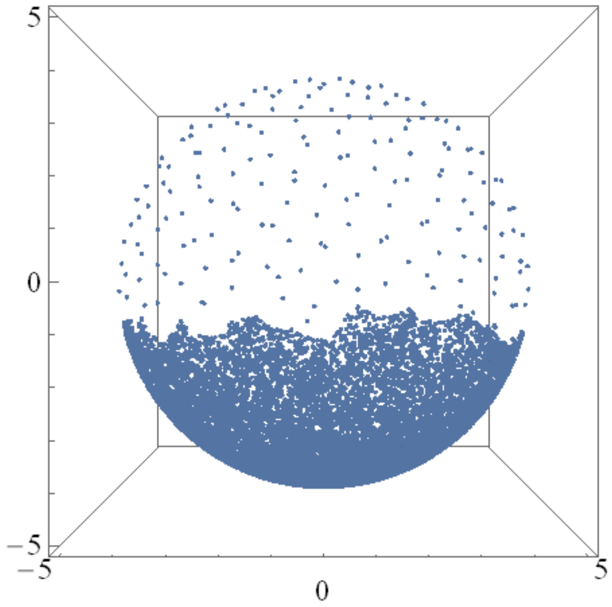

Figure S3: Sample of an ordered deletion of vertices from an initial 35,000 points after 20,000 have been deleted.

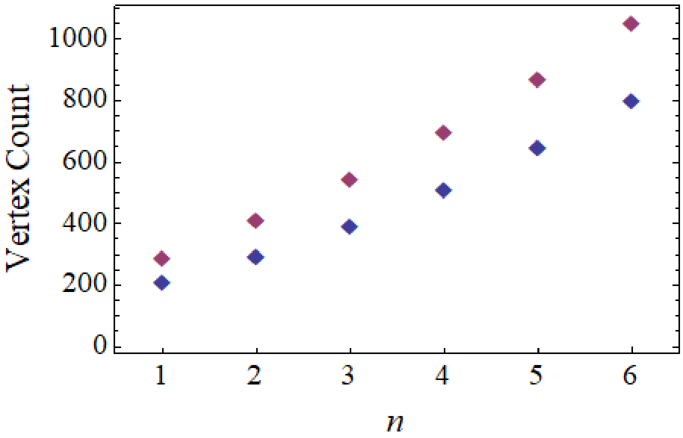

Figure S4: A test comparing vertex counts for sorted deletion (above) and unsorted deletion (below) for 35,000 randomly generated vertices after five consecutive increments of $\Delta r = 1$ starting from $r = 5$.

2.5 Triangulation

Once all programed runs are performed, a triangulation is calculated for each generation of vertices (an example is shown in figure S7). The triangulation algorithm is a Delaunay triangulation (see for instance De Berg et al., 2008 for more details), which in this case results in a unique triangulation for a given set of fixed vertices in 3D space. This is in contrast to the triangulation performed in CDT, where different triangulations for a given surface are allowed with the condition that simplices remain equilateral and the vertices are not fixed.

The resulting triangulated geometries are highly irregular in the first few iterations (see Figure S7), but become more symmetrical as the iteration number increases, as in Figures S8 and S9. As the radius

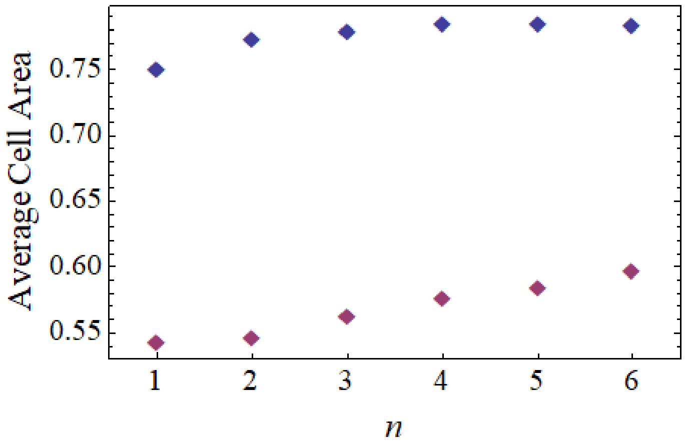

Figure S5: A test comparing average cell areas for unsorted deletion (above) and sorted deletion (below) for 35,000 randomly generated vertices after five consecutive increments of $\Delta r = 1$ starting from $r = 5$.

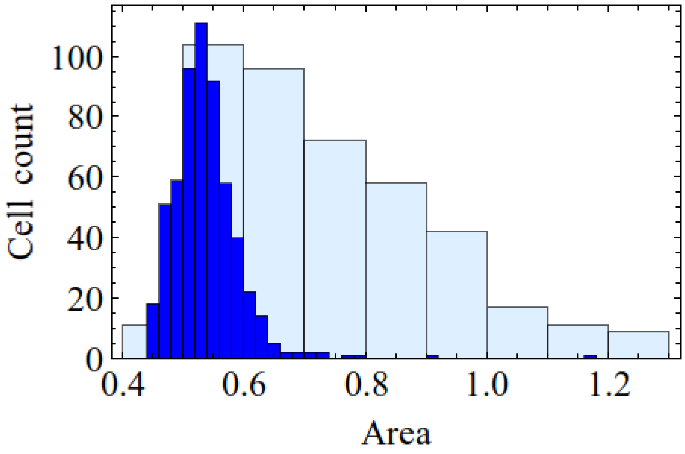

Figure S6: Comparison of cell-area distributions for the first set of 35,000 vertices at $r = 5$ after sorted (blue, foreground) and unsorted (light blue, background) deletion.

increases, the average cell area slowly converges to the minimum of ~ 0.433 as long as the deletion process is ordered as described above. When this is not the case, the average cell area levels off at ~ 0.8.

2.5.1 Close Neighbors

We tested the behavior of the simulations under the scenario where vertices could only combine with their immediate neighbors instead of combining with all other vertices. First, an adjacency graph was created for the whole set of points at the beginning of each iteration with the built-in function `AdjacencyGraph`, which takes as an argument a list of vertices and their adjacency matrix, the latter of which we recovered as a property of the `DelaunayMesh` performed on the vertices. Iterating over each vertex of the main adjacency graph, we recovered the neighbors for each point with another built-in function, `AdjacencyList`. With these results stored in lists, we could reliably combine vertices with their immediately adjacent neighbors only. Given that each vertex had an average of 6 or 7 neighbors, this resulted in many fewer vertices generated in each iteration and therefore the algorithm ran much faster than the “all versus all” approach, allowing us to perform many more runs and iterations.

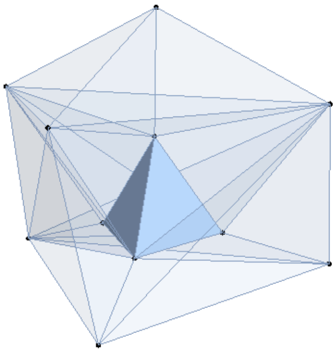

Figure S7: The first and second sets shown together with timelike links between them.

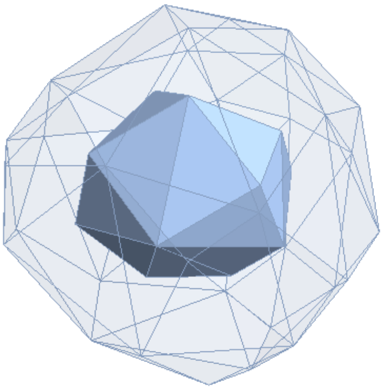

Figure S8: The fourth and fifth sets and their triangulated surfaces, shown together. Timelike links have been omitted for clarity.

2.6 Analysis

By design, the radius follows a random-walk behavior for all scenarios, increasing according to $\langle R\rangle = R_0 + l\sqrt{n}$ (figure S10). The number of vertices and total area track the behavior of R but, interestingly, the average cell area doesn't always converge to the minimum of a triangle with sides $l = 1$, $\sqrt{3}/4 \sim 0.433$, but instead depends on both the vertex creation and deletion algorithms chosen. The average cell area is greatest for the close neighbors algorithm with unsorted deletion, converging to a value of about 1. Combining close neighbors with sorted deletion gives an average area of ∼0.8, which is also the value for all points combining with each other and an unsorted deletion process. Finally, all points combining with all others in addition to sorted deletion results in a slow but steady convergence to the minimal area, though calculation times allowed us to get only as close as ∼0.5 (average cell areas for each of these scenarios are shown in figures S11 to S14).

2.7 Spacetime Foam

In order to simulate a spacetime foam made of cells that each go through the process described above, we used the results of different realizations for individual cells and then placed them randomly on a square 3D

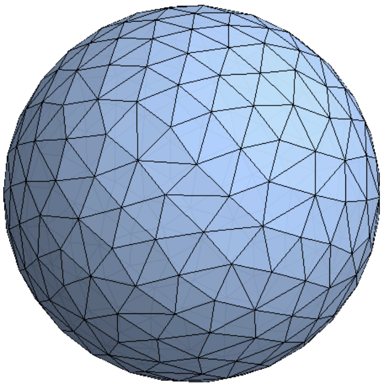

Figure S9: As the number of iterations increases and the radius grows, more points fit into the surface area and the triangulation resembles the sphere more and more.

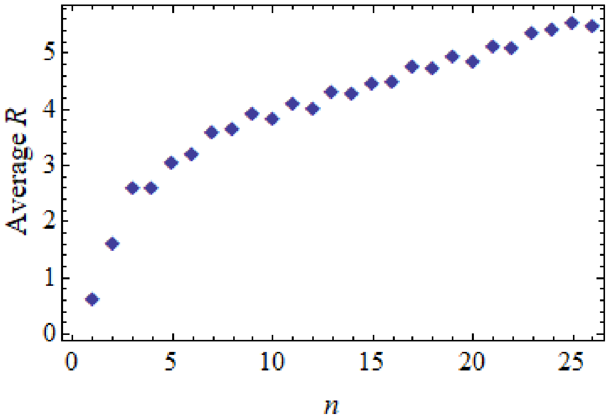

Figure S10: The average radius given in function of iteration number n, for a 200×25 trial (runs×iterations; error bars are omitted for clarity).

lattice. All cells begin with the initial tetrahedron and are allowed to evolve independently. This eventually leads to their volumes overlapping, but no restrictions are applied to this process for now.

REFERENCES

De Berg M, Cheong O, Van Kreveld M, Overmars M. *Computational Geometry* (Berlin, Heidelberg: Springer Berlin Heidelberg) (2008), 723–741. doi:10.1007/978-3-540-77974-2.

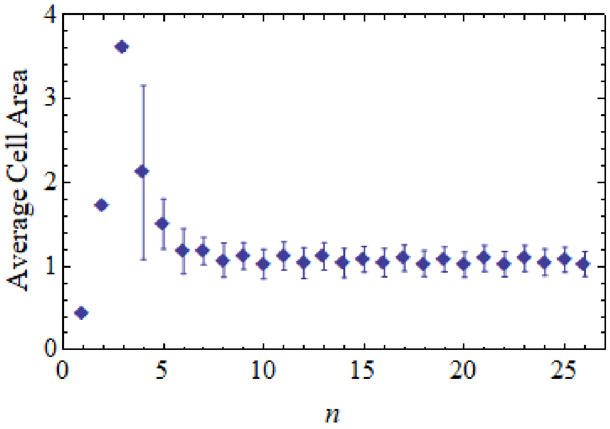

Figure S11: The average cell area given in function of generation number n, for a 200×25 trial combining only close neighbors and unsorted deletion. The average cell area stabilizes at ~ 1.

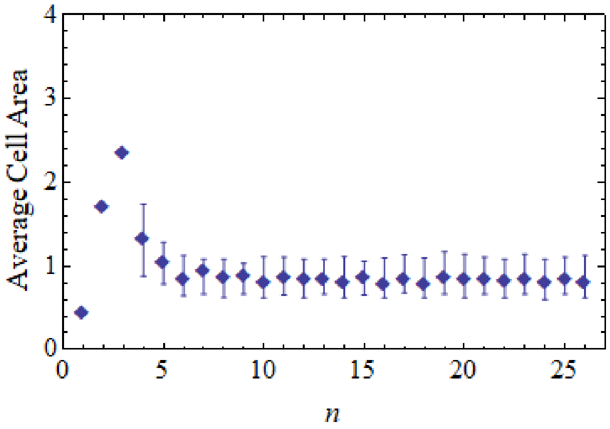

Figure S12: The average cell area given in function of generation number n, for a 50×25 trial combining only close neighbors but with sorted deletion. The average cell area stabilizes at ~ 0.8.

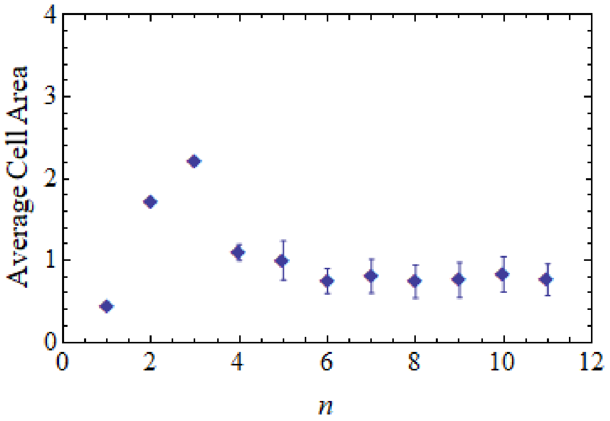

Figure S13: The average cell area given in function of generation number n, for a 20×10 trial combining all points with all others and using unsorted deletion. The average cell area stabilizes at ~ 0.8.

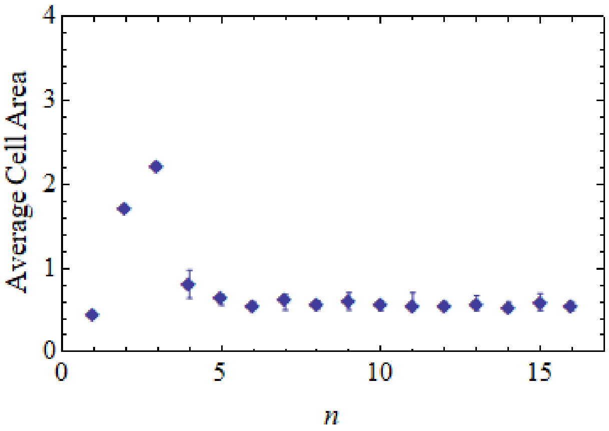

Figure S14: The average cell area given in function of generation number n, for a 20×15 trial combining all points with all others and using sorted deletion. Here, the average cell area slowly approaches the minimum, with a final area of ~ 0.5 after the performed iterations.

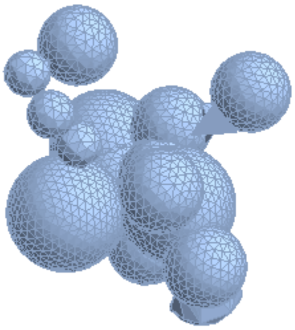

Figure S15: A spacetime foam of 20 cells after an evolution of 10 iterations. The cells were evolved under the full combination algorithm with sorted deletion.

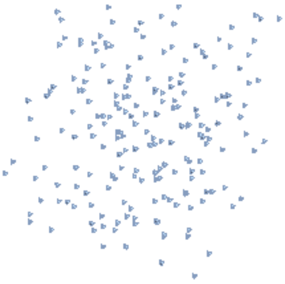

Figure S16: A spacetime foam of 200 cells in the initial stage. For practical computational purposes, the cells in this example were evolved using the close-neighbor algorithm, which leaves radius behaviors untouched.

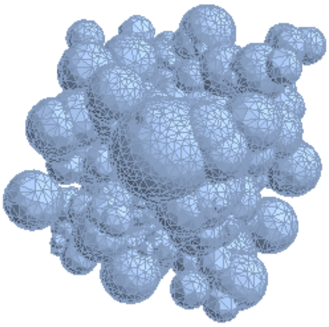

Figure S17: The same set of cells from figure S16 after 10 iterations.

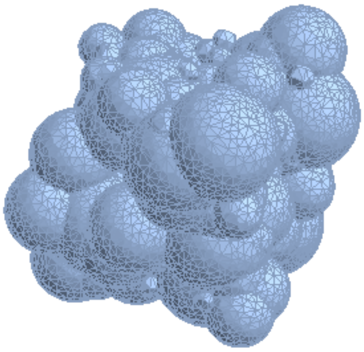

Figure S18: The evolution of spacetime cells after 20 iterations.