A pseudo-random network mobile automaton with linear growth

Tommaso Bolognesi

CNR/ISTI, Pisa - t.bolognesi@isti.cnr.it

Abstract. Based on the *mobile automaton* computational paradigm, a (fully deterministic) algorithm is introduced that grows planar graphs while exhibiting surprisingly uniform pseudo-random dynamics. The graph growth rate, as a function of the automaton steps, is O(n), and nicely combines with the O($\sqrt{n}$) of a previously introduced algorithm of ours: these two rates are also achieved by two most elementary visit-and-grow procedures with regular dynamics, of which our automata can be viewed as randomized versions. Applications of mobile automata to fundamental physics and quantum gravity have been recently suggested, but other application areas are envisaged as well.

Keywords. Graph algorithms, pseudorandom numbers, cellular automata, network mobile automata.

Cyclic mobile automata with regular dynamics

In a classical Turing machine, a control head moves on a tape of binary cells and performs 1-step moves and read/write operations, based on a state transition table; the extension to cell arrays of two or more dimensions is straightforward. Somewhat similar to two-dimensional Turing machines, a *planar network mobile automaton* is a model of computation in which a control head moves by unit steps on the nodes of a planar graph and performs local, read/write operations on the structure of the latter, based on some control logic. Typically, the control logic includes a set of graph rewrite rules and, possibly, criteria for resolving potential nondeterminism in rule application. However, the control head is *stateless*: it does not store information for deciding which rule to apply and where to move next.

We are interested in visit-and-*grow* patterns, thus we admit rewrite rules that add nodes to the graph. Let us start by considering graphs where each node may only have degree 2, that is, exactly two neighbors. Clearly these graphs can only be cycles. Two most elementary mobile automata for cyclic graphs are depicted in Figure 1.

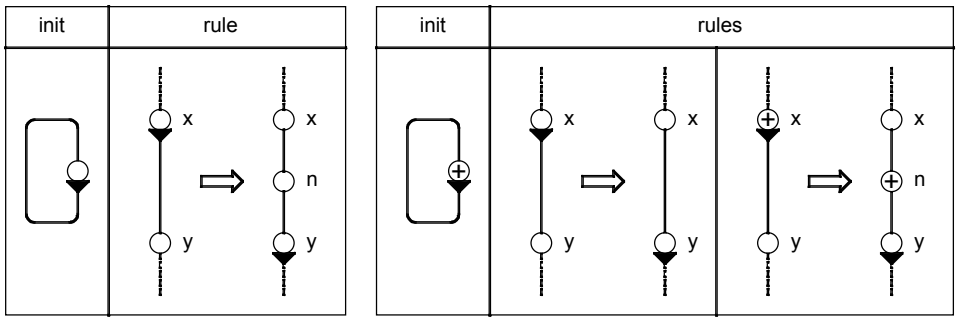

Figure 1 - Two cyclic mobile automata yielding linear (left) and square root (right) node growth

In Figure 1, x and y identify nodes, n identifies a newly created node, and a small triangle indicates the position of the control point. Both automata move always in the same direction around the cycle. The automaton on the left introduces a new node at every step; the one on the right does it every time a circular visit is completed, and, for detecting the end of a cycle, it flags a node by the '+' symbol.

We are particularly interested in visit-and-grow algorithms that never permanently abandon regions of the growing graph. To this purpose, we assign progressive natural numbers to the nodes, as they are created, and use them for plotting the trajectory of the control point. The *revisit indicators* for the two automata above are shown in Figure 2. Note that the automata do not make any use of these numbers: the only node labelling used as state information is the flag, in one case.

The revisit indicators of both automata exhibit a regular pattern, for which one can anticipate future values without need to run the computation step by step.

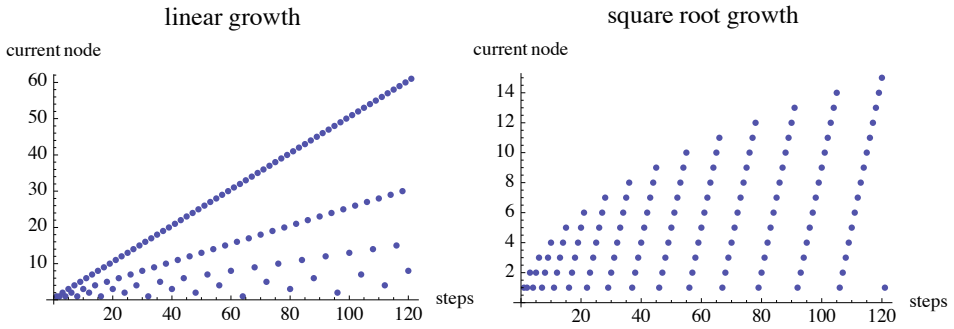

Figure 2 - Revisit indicators for the linear (left) and square root (right) mobile automata of Figure 1

The first revisit indicator exhibits an overall linear growth, and is the well known fractal sequence (1, 1, 2, 1, 3, 2, 4, 1, 5, 3, 6, 2, 7, 4, 8, 1, ...), described in [5] and [3]. This sequence is self-similar: it reproduces itself once the elements in odd positions are deleted. Clearly, every node of the growing graph is visited infinitely often. In particular, node k (k = 1, 2, ...) is visited at steps $2^m (2k - 1)$, with m = 1, 2, ... , so that the intervisit intervals for any given node grow exponentially. The second revisit indicator exhibits a square root growth; again, every node is visited infinitely often, but now intervisit intervals grow linearly.

In the next two sections we introduce two mobile automata that exhibit chaotic behaviour and appear as pseudo-random versions of the two regular automata above. Interestingly, this is achieved without exploiting any state information other than local topology: no node/edge labels or flags are involved. Quite obviously, these restrictions make it impossible to obtain complex dynamics while manipulating just cycles; we shall therefore consider slightly more complex graphs, and handle *planar trinets*.

Linear chaotic trinet mobile automaton

A planar *trinet* is a planar, undirected, connected graph in which all nodes have degree 3. By definition, any planar trinet T can be embedded on the sphere in such a way that edges do not cross one another; such an embedding partitions the spherical surface into regions, or *faces*. The dual graph D of T is a planar graph in which nodes correspond to the faces of T's embedding and the edges reflect the face adjacency relation.

The faces of a trinet dual are always (possibly degenerate) triangles, while node degrees have no upper bound. Six examples of planar trinets are shown in Figure 3 (black nodes), each with its corresponding dual graph (white nodes).

A connected graph is *n-connected* when n is the smallest number of edges one has to remove in order to disconnect it. By definition, a trinet is at most 3-connected. Figure 3 shows examples of 1-, 2-, and 3-connected trinets.

Our planar trinet mobile automata modify the local structure of the graph by using the two graph rewrite rules illustrated in Figure 4 (left), that we call R and S. In terms of trinets, rule R replaces a node with a triangle of interconnected nodes, while rule S preserves the number of nodes and modifies the local edge pattern. In terms of trinet duals, rule R places a new node inside a triangular face and connects it to its three vertices, while rule S flips the diagonal of the rhombus formed by two adjacent triangles. (These two rules are known in knot theory, and play an important role also in Loop Quantum Gravity [6].)

Proposition 1. Let *T* be a (connected) planar graph, and *D* be its dual. Then (i) *T* is 1-connected if and only if *D* has a loop edge; (ii) *T* is 2-connected if and only if *D* has double edges (ones that connect the same pair of nodes). **Proof**. Trivial. []

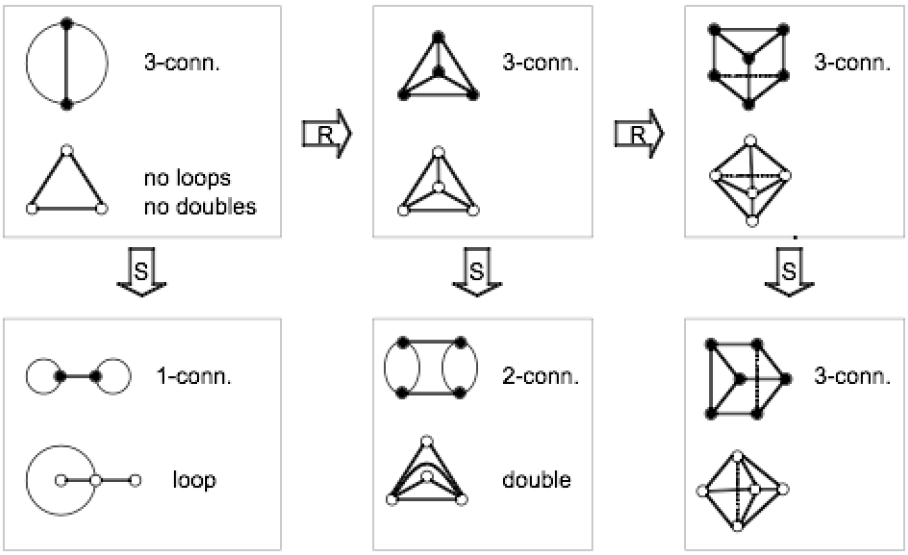

Figure 3 - Six trinets with connectivity 1, 2, or 3, their duals, and their transformations by rules R and S

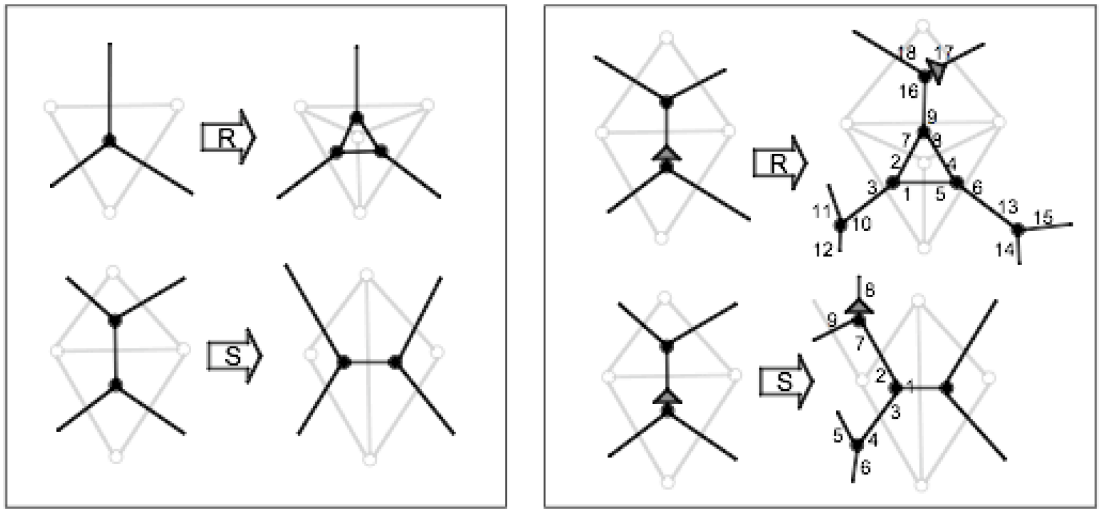

Figure 4 - Rules R and S applied to trinets and to their duals (left) - choice of next head position (right)

Figure 3 illustrates the relation between the connection degree (1-3) of some trinets and the presence/absence of loops and doubles in their duals; it also shows examples of application of rules *R* and *S* to simple trinets. For example, the application of *R* to the upper left trinet preserves 3-connectivity, while the application of rule *S* to the same trinet yields a 1-connected trinet.

The definition of a planar trinet mobile automaton (TMA) involves describing: (i) an inital condition -- a trinet and an oriented location of the control head; (ii) a criterion for chosing between rules *R* and *S*, and (iii) the step of the control head, i.e. where it moves after a rewriting; this may depend on the applied rule. Let us define our first planar automaton, called TMA1.

Definition 1 [TMA1]

- **Initial condition**: the 3-connected trinet in the upper-left corner of Figure 3, with control placed at either node.

- **Rule choice criterion**: choose *S* whenever it does not violate the 3-connectivity of the trinet, otherwise choose *R*.

- **Step**: depending on the applied rule, move the control head to the new location marked by the grey triangle in Figure 4 (right). []

In our *Mathematica* implementations of network mobile automata, we conveniently manipulate trinet duals; then, in light of Proposition 1, the rule choice criterion is implemented by checking whether the new edge that Rule *S* would introduce is a loop or a duplicate. The initial trinet is 3-connected and, by induction, so are all the subsequently generated graphs.

Fact 1 - A 3-connected trinet may not have loop edges. **Proof** - If it had one, incident to, say, node *p*, then the removal of the only other edge incident to *p* would disconnect the graph, thus contradicting its being 3-connected. []

Fact 2 - A 3-connected trinet may not have double edges either, unless it is the *two*-node trinet in the upper-left corner of Figure 3. **Proof** - The simple proof, again by contradiction, is obtained by considering the sub-trinet including two double edges *e1* and *e2*, connecting nodes *p* and *q*, and by observing that the presence of two further, *distinct* edges *e3* and *e4* connected, respectively, to *p* and to *q*, would imply that the trinet is 1- or 2-connected, regardless of where *e3* and *e4* are placed with respect to the finite region between *e1* and *e2* (inside or ouside). []

Thus, the faces of the running trinet, except for the initial one, are always at least triangular, and their dual graphs have nodes with degree at least 3. Figure 5 illustrates the revisit indicator for TMA1 corresponding to a 20,000-step computation, and the resulting trinet.

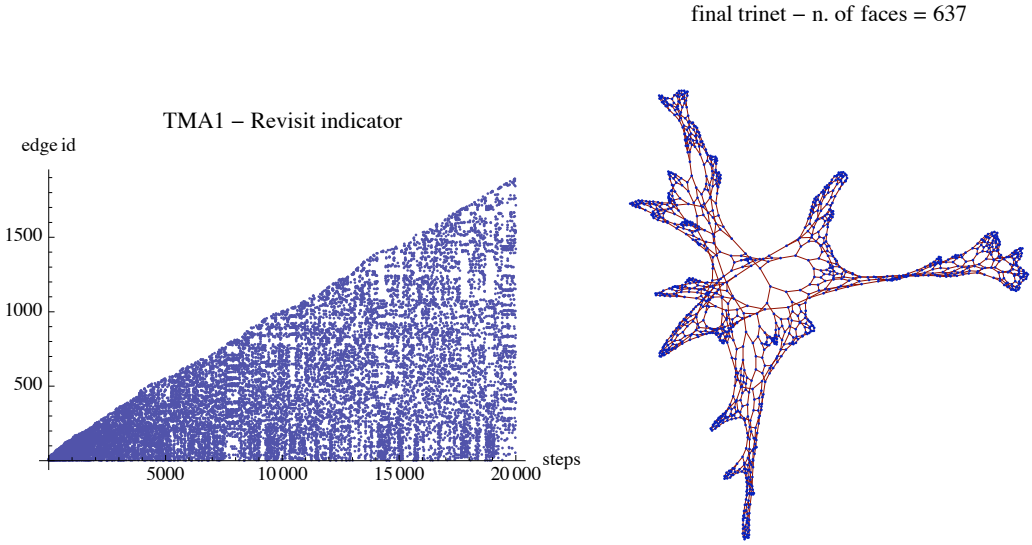

Figure 5 - Revisit indicator and final trinet for a 20,000-step computation of TMA1

Although in the revisit indicators introduced earlier the current position of the control head is represented by *node* identifiers, an edge or face identifier could as well be chosen, either of the trinet or of its dual, without affecting the general pattern of the revisit indicator -- whether regular, fractal, linear, random-like, etc. -- except for a scale factor (in a trinet with n nodes and e edges, it must be $3n = 2e$, so that, for example, edges are created 1.5 times faster than nodes). In the revisit indicators just introduced we record current trinet *edges*, that is, we trace the edges successively traversed by the automaton, as identified by the grey triangles in Figure 4 (right). In doing so, we adopt the rather natural convention of preserving the identifier of the edge that 'flips' when rule *S* is applied.

Figure 5 indicates that the growth is roughly linear, with one new edge created about every 10 steps, or, a new face created by rule *R* every 30 steps, since in a planar trinet with e edges the number of faces is $f = (1/3)e + 2$, as implied by Euler's formula $f - e + n = 2$, and by equation $3n = 2e$ above.

In Figure 6 we visualize the cumulative number of edge occurrences for the 20,000 step computation of TMA1 in Figure 5; point (e, f) in this plot indicates that edge e occurs a total of f times in that specific computation. The origin of the vertical axis is conveniently set to -1, for better visualisation of zeroes. Since the creation of an edge does not count as a visit, one does find zeroes in the plot, but these occur only for the most recently created edges, i.e. in the r.h.s. portion of the function: eventually, *all* edges are visited, and, apparently, they are visited infinitely often. In Figure 6 we also plot, as a solid line, the theoretical edge occurrence density. We have assumed that: (i) the revisit indicator contour is perfectly linear, with rate $k = 1893/20{,}000$, where 1893 is the highest numbered edge found; (ii) at step s the current edge is a random variable with uniform distribution in the interval [0, k*s]; (iii) these variables are independent. Then,

letting $h = 1/k$, the expected number $F(e)$ of occurrences of e in the computation can be expressed, in a continuous approximation, as

$$F(e) = \int_{h\,e}^{20\,000} \frac{h}{s}\, ds = h\,(\text{Log}[20\,000] - \text{Log}[h * e])$$

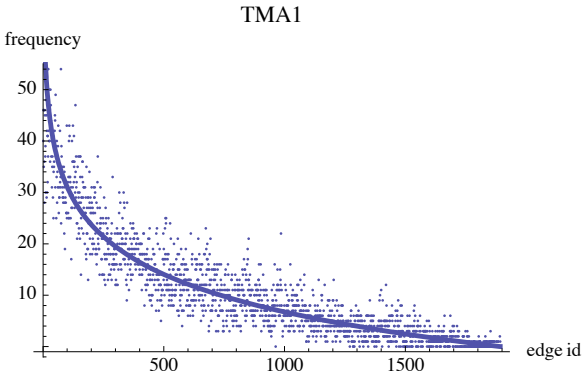

Figure 6 - Density of edge occurrences for a 20,000 step computation of TMA1 (compare with Figure 5)

In this paper we are not particularly interested in detailed measures of the intrinsic 'degree of randomness' achieved by TMA1. However, an effective way to appreciate its singular behaviour consists in comparing its revisit indicator with those of similar TMAs. Figure 7 shows the revisit indicators for a family of 3-connectivity-preserving trinet mobile automata of which TMA1 itself is a member. These automata share with TMA1 the initial trinet and the rule choice criterion, but differ from it and from one another in the step of the control head, which is controlled by two parameters *R-code* and *S-code*, ranging, respectively, in intervals [1, 18] and [1, 9], and yielding 18 * 9 = 162 family members. The way these parameters determine the step is illustrated in Figure 4 (right). The numbers in figure correspond to values of the *R-code* (resp. *S-code*) parameter, and identify the new position of the control head after rule *R* (resp. *S*) is applied. Note that, based on symmetry considerations, we can safely restrict to 9 values for the *S-code* parameter. For space reasons, some rows are not shown in the table: but they all replicate the pattern of row 1. Interestingly, this family of 3-connectivity-preserving automata also yields some regular revisit indicators equivalent to those introduced in the previous section. The revisit indicator for TMA1 has code pair {17, 8}, as shown by the grey triangular markers in Figure 4 (right), and appears as unique in its class. Three other cases of apparently non regular indicators appear in the table, where 500-steps computations are considered. But longer computations reveal that case {17, 4} is a perturbed version of the fractal sequence, and cases {13, 5} and {17, 9} eventually stabilize to a periodic behaviour with linear growth. In conclusion, TMA1 is the only case, in this family, which appears to behave pseudo-randomly forever.

Of course we have no absolute guarantee -- no formal proof -- that the automaton will not eventually stabilize. Our conjecture is based only on the remarkable difference between the revisit indicator of TMA1, which we run for up to 340,000 steps, and those of the dozens of mobile automata with random-like initial behaviour whose eventual stabilization we have been able to observe [1, 2]: while TMA1 yields a dense, uniform distribution, in the latter we always observe transient phases with large empty space-time regions (similar to those in Figure 11, r.h.s.).

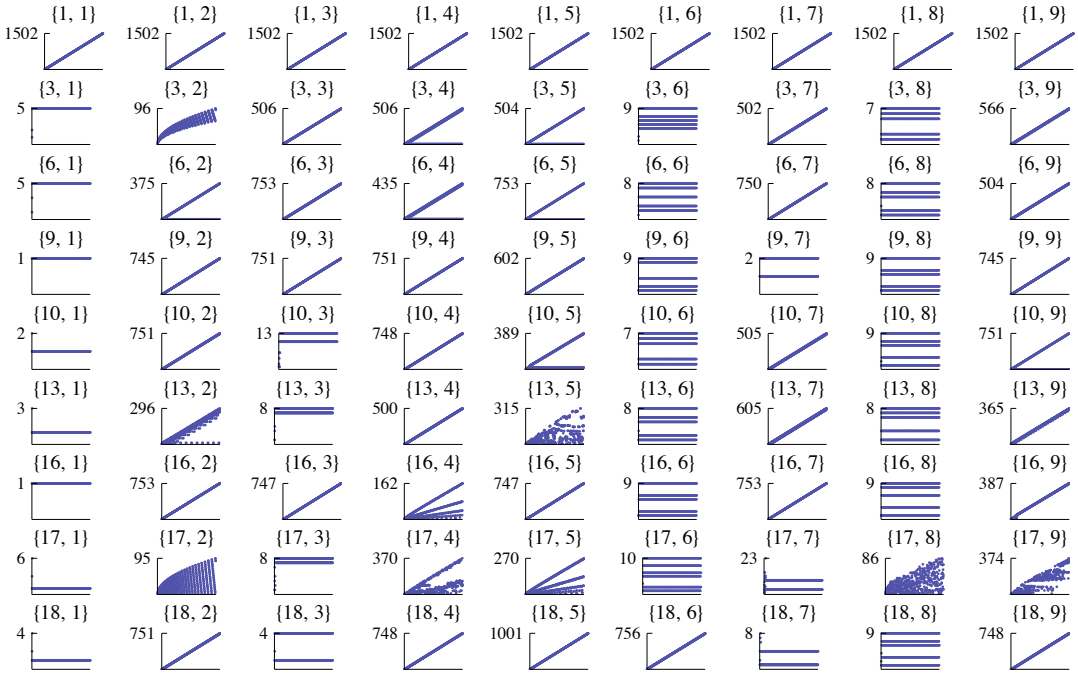

Figure 7 - 500-step revisit indicators for a family of automata including TMA1 (case {17, 8})

A peculiarity of TMA1, and a departure from any reasonable notion of random planar graph, is that square faces are almost completely missing in the trinets it computes. For the 20,000-step computation, yielding a total of 637 trinet faces, the following distribution was obtained (read: 35 triangles, 1 square, 321 pentagons, etc.):

{{3, 35}, {4, 1}, {5, 321}, {6, 121}, {7, 68}, {8, 38}, {9, 19}, {10, 11}, {11, 6}, {12, 6}, {13, 5}, {14, 1}, {16, 2}, {18, 1}, {19, 1}, {23, 1}}

Incidentally, these data definitely depart also from the limit distribution mentioned in [7], p. 1038, obtained by interated random application of rule *S* to the hexagonal grid.

As a final comparative element, in Figure 8 we show the revisit indicators for *2000*-step computations of a *RandomWalk* and a *RandomJump* automaton. These automata use the same rule choice criterion as TMA1, but different policies for the step: for *RandomWalk* the next head position is defined, at every step, by new random values of *R-code* and *S-code*; for *RandomJump* these parameters are ignored and a new position is chosen randomly in the graph -- not just in the neighborhood.

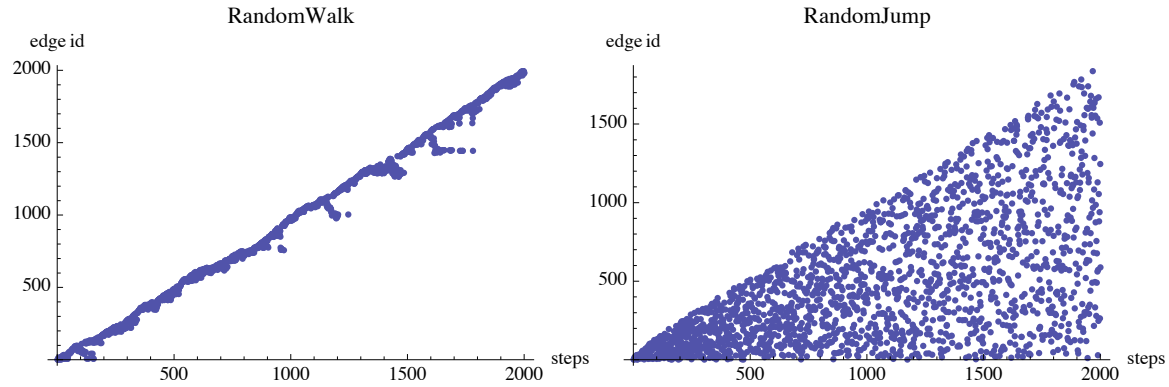

Figure 8 - Comparing revisit indicators for TMA1 and two randomized automata,

By inspecting these 2000-step indicators, and comparing them with the one for TMA1 in Figure 5 (which was based on a *20,000*-step computation) two remarkable facts emerge:

- In both random cases the number of edges is roughly the same as the number of steps, implying that the frequency of rule *R* application is 1/3, since each such application introduces 3 new edges in the dual graph. This is quite in contrast with the TMA1 case, with its ratio 1/30.
- The *RandomWalk* automaton, which represents a sort of randomized version of TMA1 (in light of its 'short' step policy), completely fails to revisit past regions of the growing graph: to obtain such a fair behaviour we have to randomize more drastically, and introduce random *jumps*.

Thus, the brownian-like motion of the pseudo-random TMA1, which is based on a fully deterministic procedure, is much more effective than a 'truly' random, brownian-like automaton, in its revisiting performance. In part this phenomenon can be explained by considering that the *RandomWalk* automaton grows the graph about 10 times faster than TMA1. However, in the next section we show that variants of the *RandomWalk* with considerably lower node growth rates still fail to achieve the revisit ability of TMA1.

Sublinear chaotic trinet mobile automaton

In [2] another, pseudo-random trinet mobile automaton was discovered, which we call now TMA2, and which differs from TMA1 in the fact of exhibiting a square-root node growth rate. We shortly recall the definition and features of TMA2 here, for comparison with TMA1.

Definition 2 [TMA2]

- **Initial condition**: the trinet in the upper-left corner of Figure 3, with control placed at either node (same as TMA1).

- **Rule choice criterion**: choose *S* whenever it does not create trinet faces smaller than a triangle, otherwise choose *R*.

- **Step**: depending on the applied rule, move the control head to the location marked by the grey triangle in the r.h.s. part of Figure 4 (same as TMA1). []

In terms of trinet duals, the rule choice criterion is that *S* can be applied only when it decreases (by 1 unit) two node degrees that have at least value 4, which we call the *threshold parameter*. We have observed above that the condition for choosing *S* in TMA1 (*S* preserves trinet 3-connectivity, that is, it does not introduce loops or doubles in the *dual*) implies the condition for choosing *S* in TMA2 (*S* does not create trinet faces smaller than a triangle, that is, it does not introduce loops or doubles in the *trinet*). The converse is not true: Figure 9 illustrates two applications of rule *S* that are admitted by TMA2, but prevented by TMA1: they do not introduce loops or doubles in the trinet, but reduce its connectivity below 3. Hence, TMA2 may produce trinets which are 1- or 2-connected.

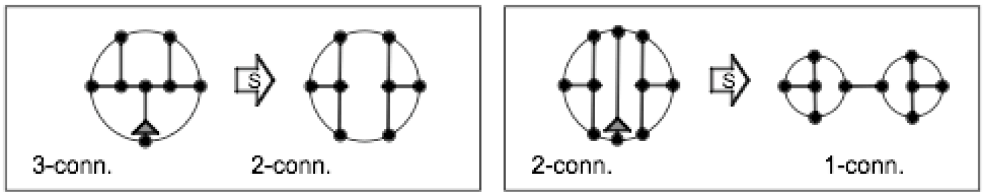

Figure 9 - Rule S may reduce trinet connectivity without introducing planar trinet faces smaller than triangles

Figure 10 illustrates the remarkably uniform revisit indicator for TMA2, which plots the current trinet edge as a function of the step, for a 20,000-step computation; the figure also shows the final trinet, and the growth of the number of trinet faces, which closely matches function $\sqrt{2}\ \sqrt{\text{steps}}\ + 3$.

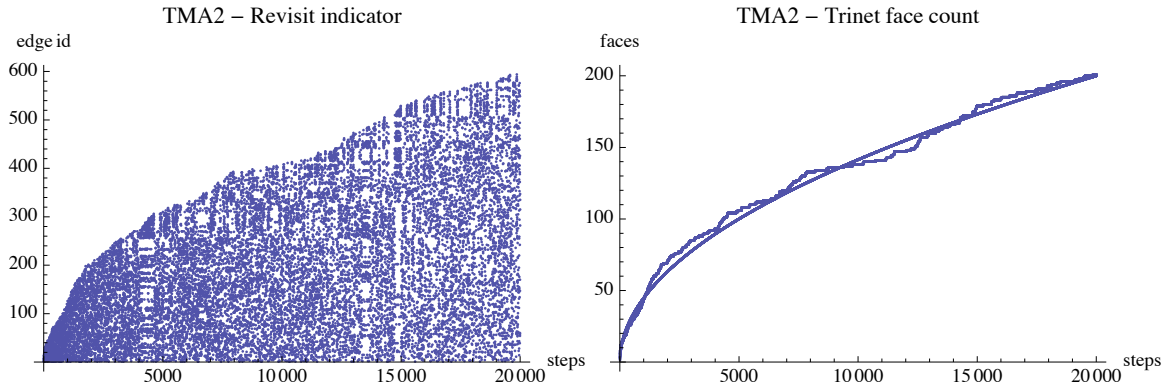

Figure 10 - Revisit indicator, final trinet and trinet face count (dual node count) for a 100,000-step computation of TMA2

A table of revisit indicators for a family of 162 trinet mobile automata that includes TMA2, analogous to the table in Figure 7, is introduced in [2]. The two tables appear quite similar, but one can spot {*R-code*, *S-code*} pairs yielding completely different dynamics in the two cases (e.g. {6, 4}, {11, 4}, {16, 2}, {17, 5}; see [2]). Surprisingly, the revisit indicator for TMA2 has again codes {17, 8}, and again this appears as the only element in its class to behave pseudo-randomly forever; all other apparently random indicators eventually stabilize to periodic dynamics.

It is quite remarkable that, in the two considered families, the pseudo-random cases correspond to only one, and exactly the same {*R-code*, *S-code*} pair: this peculiar choice of how the control head moves after applying rule *R* or *S* seems to be responsible for the random like dynamics, independently of the rule choice criteria.

The distribution of face sizes, for the 469-face trinet produced by a 100,000-step computation of TMA2 is (read: 2 triangles, 1 square, 256 pentagons, etc.):

{{3, 2}, {4, 1}, {5, 256}, {6, 105}, {7, 43}, {8, 32}, {9, 11}, {10, 5}, {11, 7}, {12, 2}, {14, 3}, {15, 1}, {18, 1}}

which, again, represents a departure from the limit distribution mentioned in the analogous remark about TMA1.

Figure 11 shows the revisit indicators for two *RandomWalk* automata whose rule choice criterion is based on the above introduced *threshold* parameter. They use, respectively, *threshold* = 4 (like TMA2) and *threshold* = 2. Every move of the control head is determined by a new random value for *R-code* or *S-code*.

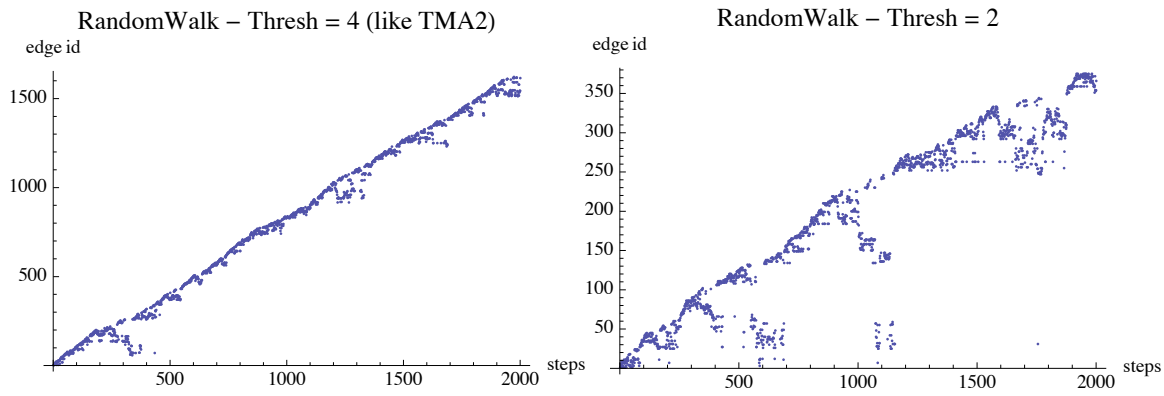

Figure 11 - Comparing revisit indicators for TMA1 and two randomized automata,

As expected, a lower value of the threshold implies less frequent application of the 'generative' rule *R*, thus a lower growth rate, as revealed by the upper values on the vertical axes; this in turn increases the chances to revisit past regions of the graph. However, the lowest threshold value that can be chosen -- threshold 2, yielding a frequency of rule *R* application of about 1/16 -- still fails to emulate the fair revisit indicator of TMA2.

Related work, potential applications, conclusions

The idea to study the 'spontaneous' dynamics of simple computational systems and to experimentally detect the emergence of complex behavior has been first proposed and thoroughly investigated by S. Wolfram in [7], where special attention is devoted to graph rewriting for its possible applications to fundamental physics. The *network mobile automaton* idea itself is due to Wolfram, but the experiments he has carried out with this model, as shortly reported in a note at p. 1040 of [7], have not succeeded in exposing especially complicated behavior. The remarkable pseudo-random automata illustrated in this paper represent a definite step forward in this investigation, and could perhaps be compared with the 'class 3' elementary cellular automata (e.g., Rule 30 - see [7]), which also produce pseudo-randomness, undefinitely, starting from elementary initial conditions. As it appears, both in elementary cellular automata and in our trinet mobile automata, pseudo-randomness is achieved in rare circumstances.

Our first experiments with planar trinet mobile automata are described in [1], where we adopted only rule *R*, and a rather different policy for the step of the control head. Those experiments did yield regular, possibly fractal dynamics, and cases with random-like initial transients, but all of them eventually settle to periodic behavior. In our subsequent experiments [2], we used rules *R* and *S*, and the three parameters {*threshold, R-code, S-code}*, and obtained a wider variety of emergent features, but, out of about a thousand cases, we only found *two* cases of apparently endless pseudo-randomness, namely {4, 17, 8} and {5, 9, 8}. Both exhibit revisit indicators with square root growth, with different multiplicative factors. No linear or quasi-linear pseudo-random revisit indicator of this type was ever found by using the threshold-based rule choice criterion. In this paper we have shown that linearly growing pseudo-randomness can be achieved by changing that criterion.

Trinet mobile automata represent an extremely simple model of computation. They share with Turing machines and cellular automata a locality feature -- updatings depend only on local information -- but are simpler than the former, in that the control head does not carry around state information, and simpler than the latter, in that no global synchronization of site (cell) updatings is required. This circumstance suggests that network mobile automata might prove useful in bio-computing applications, for example in building nano-scale structures by using an agent of very limited operational capabilites. Once (physical/chemical/biological) substrata are identified that offer adequate levels of flexibility and stability, that can play the role of the growing trinets, and once associated processes are identified that may implement the TMA head operation, a range of possible applications would be at reach, exploiting the variety of regular and pseudo-random patterns that can be obtained in this way, as shown here and in [2]. Another very attractive application area for network mobile automata is Quantum Gravity, where graph rewrite rules such as *R* and *S* and their higher dimensional versions ('Pachner moves') are used for animating discrete models of spacetime (see, e.g., [4]). While the idea to apply Pachner moves to discrete models of space is mentioned in several theoretical physics papers, we are not aware of any substantial experimental effort to investigate their emergent properties in the long run, and in a fully deterministic setting.

Finally, we believe we have provided some evidence that the apparently bizarre NKS idea of exhaustively exploring portions of the computational universe in search for useful algorithms, without a specific target in mind, but with adequate visualization techniques, can be fruitful.

References

1 - T. Bolognesi, 'Planar Trivalent Network Computation', Proceedings of Machines, Computations, and Universality - MCU 2007 - Lecture Notes in Computer Science, 4664, Springer, 2007, pp. 146-157.

2 - T. Bolognesi, 'Planar Trinet Dynamics with Two Rewrite Rules', to appear in *Complex Systems*, 2008.

3 - C. Kimberling, 'Numeration systems and fractal sequences', Acta Arithmetica 73 (1995) 103-117.

4 - F. Markopoulou, 'Dual formulation of spin network evolution', preprint available as arXiv:gr-qc/9704013v1, url: citeseer.ist.psu.edu/markopoulou97dual.html.

5 - N. J. A. Sloane, The On-Line Encyclopedia of Integer Sequences, http://www.research.att.com/~njas/sequences/-A003602

6 - L. Smolin, Three Roads to Quantum Gravity, Basic Books (Perseus Books Group), 2001.

7 - S. Wolfram, A New Kind of Science, Wolfram Media Inc., 2002.