Exploring Stable and Emergent Network Topologies

P.D. van Klaveren [a] H. Monsuur [b] R.H.P. Janssen [b] M.C. Schut [a] A.E. Eiben [a]

[a] *Vrije Universiteit, Amsterdam, {PD.van.Klaveren, MC.Schut, AE.Eiben}@few.vu.nl*
[b] *Nederlandse Defensie Academie, Den Helder, {H.Monsuur, RHP.Janssen}@nlda.nl*

Abstract

Many different types of networks, e.g., economic, social, computer, military and intelligent agent networks, have a fundamental property in common: they change over time. For various reasons, nodes form and terminate links, thereby rearranging the network. In this paper, we experimentally analyse a structural network mechanism by means of computer simulations. With this mechanism, nodes deliberately form and terminate links as they attempt to gain network advantage and/or an identifiable position in the network. Reiterative application of this mechanism, which only uses local network characteristics, results in emergent, stable network topologies, i.e., ring, uni/bi-polar and complete networks. The process demonstrates that local decisions can shape global network structures. Our results may be used to derive rules of thumb for designing networks.

1 Introduction

The concept of a *network* has become a basic and central notion in several scientific disciplines. In sociology and economics, many issues, like the interaction between individual entities, are formulated in terms of networks [1, 3, 7, 8]. The network approach has been fruitful due to the analytical tractability: there are several ways of measuring relevant features or network statistics such as power, domination, centrality, clustering etc. These metrics can be used to make explicit how actions of actors can be viewed as consequences of the system of relations by which they are constrained and/or empowered ('*structural analysis*').

An interesting issue is the *network formation*, where the focus is on how and why networks form and how these networks affect outcomes of social, economic and also military interaction. There are several random network models to investigate this formation process. Well-known examples are the small-world models and the preferential attachment models, which are useful for the investigation of large and complex networks that have certain properties. For example, in preferential attachment models, new nodes form links to existing nodes with probabilities depending on the number of links these nodes already have, while small-world phenomena emerge due to arbitrary re-wiring of a few links. Useful as they are, these models lack some important aspects, e.g., there are many settings in which not (only) probability, but other rational choice mechanisms determine which links will be formed. Real-world examples of such settings are political and military alliances, professional collaborations, friendships etc.

The model that we investigate and simulate in this paper falls within the category of game-theoretic models, or to be more precise within the class of *pairwise stability models* [4]. The key point of these models is that all agents have a tendency to form relationships that are (mutually) beneficial and to (unilaterally) sever relationships that are not.

The (non-probabilistic) model this paper is based upon (introduced in [6]), defines benefits in terms of properties of the network topology only. This means that the rational choice mechanism that is used to decide which links have to be created or severed, only uses (known) network statistics: the network structure itself serves as a repository of information. The fact that this model is based on reasonable network statistics only (and does not need artificial and unrealistic utility functions involving economic costs and profits), makes it a good candidate to study the development of a variety of networks, also for networks where notions of economic benefit are quite intangible, like in social, information and military networks.

As is proved in [6], starting from arbitrary networks, only a few generic stable network topologies emerge as a result of the iterated choice decisions of the individual nodes: ring, uni/bi-polar and complete topologies. In this paper, we implement an algorithm, called SENT, that can be used to simulate the mathematical model from [6]. While [6] offers a good framework and theoretical analysis, the analysis of system behavior is restricted because the system becomes too complex for networks with more than just a few nodes. Here, we report on a study based on an experimental setup, relying on computer simulations, in with randomly chosen nodes have the opportunity to change the network, in accordance with the choice mechanism described in [6].

The goal of this simulation is not the verification of the formal mathematical theorems of [6]. Instead, one of the research objectives of this paper is *to explore which topologies emerge when our experimental setup is used*. Compared to the existential mathematical proof, where we push the network towards the set of final stable configurations, the probabilistic approach as implemented in the SENT algorithm, may provide extra information concerning the probability distribution over the possible stable topologies. In our approach, we generate random networks (of different sizes and densities), run the SENT algorithm on each of these networks (for a limited amount of time), and observe to which topologies the networks converge. This will enable use to answer the following research questions:

1. to which of the four generic types of pairwise stable topologies do networks converge to;
2. how long does it take to come to these pairwise stable topologies; and
3. what is the influence of the network size and the density on the stabilization?

We are interested in answering these questions also because of their relevance in the military sciences. In military network sciences, several major research challenges have been formulated. One of these challenges, see [2, 5], is the study on *dynamics, spatial location, and information propagation in networks*. A major need in network science is a better understanding of the relationship between the architecture of the network and its function. This is particularly important in networks where dynamics plays an important role, either through the flow of information around the network or through changes in the network structure (by evolution or adaptation).

2 The SENT algorithm

In this Section, we present the *Stable-and-Emergent-Network-Topologies* (SENT) algorithm that iteratively changes the topology of a network, based on the mathematical model presented in [6]. Each node has a preference for a high status in the network and a minimal *degree*, i.e., the number of links that it has to other nodes, as long as the latter does not conflict with the former. The essential concept of the algorithm is the so-called (dichotomous) *cover status* of a node. We consider a node x *covered* when there is another node in the network that is linked to all the nodes that are linked to x, but also to at least one node that is not linked to x. If this is not the case we consider node x *uncovered*. For example, from an informational point of view, if a node x is covered it is intuitively clear that it is in a subdued position and has lower status. Randomly chosen nodes of the network can perform two actions that possibly change their cover-status: unilaterally remove a link with a node to which they are linked, or bilaterally add a link to a node to which they are not yet linked. A node can unilaterally decide to remove a link. A node will be inclined to do so, if its cover status does not decrease. The node may also link with another node. This occurs if through the creation of the new link both nodes increase their status: before the new link both are covered, afterwards both are uncovered.

We require the whole network to remain connected at all times; also, the network is homogeneous in the sense that all nodes behave the same (i.e., execute the same local code). Under these circumstances, it was proven that iterative execution of the add and remove operators leads to just a few particular types of pair-wise stable network topologies [6]. With pair-wise stability we mean that there is no pair of nodes in the network that will perform any action.

It is well known that the number of possible network topologies grows exponentially with the number of nodes. As mentioned, we are interested in the final, emerging topology if we use the SENT algorithm. The four generic types of stable topologies from [6] are: 1. a *ring*, meaning that there are more than two nodes in the network and each node has a degree of two; 2. a *uni-polar* network: there are more than three nodes in the network and the number of leaves is equal to the number of nodes in the network minus one;

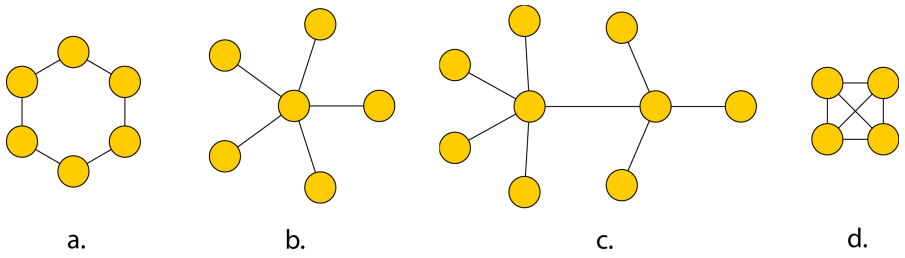

Figure 1: Types of pair-wise stable network topologies: (a.) ring, (b.) uni-polar, (c.) bi-polar, and (d.) complete.

random seed for network generation (R_{ng})	1 to 100,000
random seed for selecting nodes (R_{sn})	1
number of runs	1
maximum number of successive passive iterations ($MAX_{passive}$)	5,000
maximum number of iterations per run ($MAX_{iterations}$)	10,000

Table 1: Experimental setup

3. a *bi-polar* network: there are more than four nodes in the network and the number of leaves is equal to the number of nodes in the network minus two; 4. the *complete* network: all the nodes in the network have a degree that is the number of nodes in the network minus one. These types are shown in Figure 1.

Figure 2 shows the SENT algorithm in pseudocode. At each iteration of the algorithm, a pair of nodes is randomly selected. If the pair is already linked, then the algorithm determines whether the node that was selected first would benefit from removing its link with the second node. Moreover, the algorithm checks if the network would remain connected. If both conditions are met, then the link between the two nodes is removed. If there is not a link between the two randomly selected nodes and both are covered, then the algorithm checks if both nodes will become uncovered when they would link up. If so, a link between the two nodes is created. At the end of each iteration, if an action was performed, the algorithm checks if the network now has any of the mentioned stable topologies.

3 Experiments

We perform a number of computer experiments in order to find the answers to the questions posed in the Introduction of this paper: does the SENT algorithm lead to stable network topologies, to which ones, and, if so, how long does it take to reach stability? This Section describes the experiments, in which we basically apply the SENT algorithm on a number of different networks and monitor the topology of each network.

In the experiment, we systematically vary the *network size* N (in term of number of nodes) in the range $[5 \ldots 12, 15, 20]$ and the *network density* δ (in terms of the probability that any two nodes in the initial network are connected) in the range $[0.5, 0.55, \ldots 0.95]$. For each combination of these two parameters, we run the SENT algorithm on a randomly created network for a fixed number of iterations. In the end, we measure the *end-shape* of the topology (being either a ring, bi-polar, uni-polar, complete or other[1] and the *shaping-time* (being the number of iterations it took to reach stability).

Besides the parameters that we systematically change (network size and density) between experiments, there are an additional number of relevant variables that we summarised in Table 1. For a single run of the experiment, R_{ng} is used to create 100,000 initial networks. For each experimental condition (network size $\times$ density), each network is parsed by the SENT algorithm. Within the algorithm, pairs of nodes are randomly selected based on R_{sn}. A run finishes when either $MAX_{passive}$ is reached (i.e., the network did not change for that many iterations) or when $MAX_{iterations}$ is reached. All experiments were done on an HP Elitebook 8530w laptop with an Intel(R) Core(TM)2 Duo T9400 2.53GHz CPU and 2 GBs of memory. The laptop ran on Ubuntu with the Linux 2.6.28-11-generic kernel and the Java-6-openjdk runtime environment. Execution times of runs varied between some seconds (for small, sparse networks) up to some minutes (for large, dense networks).

[1]The 'other' type includes all topologies that do not belong to the generic types.

```
1: while no stable network detected do
2:     Select random node x
3:     Determine if x is covered or uncovered
4:     if there remain nodes other than x that have not been selected during this main iteration then
5:         Select random node y
6:         if x and y are linked then
7:             Determine if the network will remain connected if the link between x and y is removed.
8:             if Network will not remain connected then
9:                 Goto line 4.
10:            else if x is covered then
11:                Remove link with y.
12:            else
13:                Determine if x will remain uncovered if the link with y is removed.
14:                if x will remain uncovered then
15:                    Remove link with y.
16:                else
17:                    Goto line 4.
18:        else
19:            if x is uncovered then
20:                Goto line 4.
21:            else
22:                Determine if x will become uncovered when linked with y.
23:                if x will become uncovered then
24:                    x proposes to link with y.
25:                    if y accepts then
26:                        Link x and y.
27:                    else
28:                        Goto line 4.
29:                else
30:                    Goto line 4.
```

Figure 2: The main loop of the SENT algorithm in pseudocode.

```
1: Take the set S of all nodes linked to x.
2: for each node s (part of S) do
3:     determine if s is linked to all nodes in S (except himself) AND s is linked to at least one node that is
       not part of S (and that is not x).
4:     if this is the case then
5:         s covers x. Exit checking method.
6: Take the set Q of all nodes that are linked to at least one node of S AND each node q (part of Q) is not
   part of S(and is not x).
7: for each node q do
8:     determine if q is linked to all nodes in S AND q is linked to at least one node that is not part of S and
       that is not x.
9:     if this is the case then
10:        q covers x. Exit checking method.
11: if the search did not yield an s or q that covers x then
12:    x is uncovered.
```

Figure 3: The main loop of the method that determines the cover status in pseudocode.

The experiments involved a number of methods for network creation, cover status checking, checking for connectedness and checking for the four mentioned stable shapes. We briefly describe these methods here. *(i) Creating a random network* – we create a random network by filling a $N \times N$ matrix with booleans, where a *true* stands for a link between two nodes. Firstly, we fill de diagonal of the matrix with *false*, since a node is not allowed to link with itself. Then, we randomly fill every cell on the lhs and rhs of the diagonal symmetrically with *true* or *false*: we use R_{sn} to draw a number[2] and if it is smaller than δ, then we assign *true* to that cell (and its mirror-cell on the other side of the diagonal). *(ii) Checking for cover status* – to check whether a node x is covered, we implemented the algorithm shown in Figure 3. *(iii) Checking for connectedness* – each iteration starts with the network being connected, therefore we only need to check whether there exists a path between two nodes if we were to remove the direct link between the two to ensure the network will remain connected. To implement finding a path between two nodes we have used a simple depth-first search algorithm that first looks if the node we are trying to find is amongst the children of the nodes linked to x, then their children and their children and so on. *(iv) Checking for known stable shapes* – in section 2 we described the topology of the various stable networks in terms of the degree of their nodes. Therefore, checking for any of these known stable shapes is trivial: one simply creates a list of the degrees and compares this to the before-mentioned description.

3.1 Results and Analysis

Our obtained results are in Figures 4 (showing the end-shape distributions) and 5 (showing the shaping-time distributions). [3] We analyse these results according to the research questions (RQ) posed in Section 1.

RQ.1 We see in the end-shape histograms that not all experiments result in the generic topologies. The networks with 5 nodes shows a fairly even distribution of end-shapes, while the bi-polar is the most common end-shape. However, for networks where $N > 5$, our results show that the ring topology is the predominant end-shape. For networks with $N \geq 6$, all networks converge to a ring, with the exception of a few networks with high δ that converge to a complete end-shape. The dominance of the ring topology can most likely be explained by the presumed tendency of nodes to cut links, as long as it does not conflict with their preference for a high status. Validation of this observation is topic of further research.

RQ.2 When considering the shaping times we are not really interested in the exact times, but rather in the scale-up behavior of these times. In general, shaping times could depend on the shape that the network converges to, necessitating a four-fold analysis. However, in our case the ring shape is practically the only one emerging for larger networks. Looking at the related plots in Figure 5, we observe a linear scale-up, when comparing the results for 5, 10, 15, and 20 nodes. This behavior is quite the same for different density values.

RQ.3 The results show that the network size has more influence on the results than the network density. However, for smaller networks, where there is a marginal difference when the density increases, all the experiments show near-identical results independent of the connection degree. In general, we see an increase in shaping time when the size of the network increases.

4 Conclusion

In this paper, we explored which network topologies emerge when our so-called SENT algorithm is used for the purpose of network formation. The SENT algorithm is an implementation of a mathematical model for network formation presented in earlier work [6]. By means of computer simulation, we successfully attempted to scale up the theoretical findings on the mathematical model (convergence to stable network topologies) from small networks to larger networks. The two defining properties of nodes on which the algorithm works are: cover status (if node x has more links than node y, then x covers y) and degree (number of links with other nodes, in earlier work also called the *cost* of a node).

[2] We used version 13 of the *MersenneTwisterFast* implementation by Sean Luke (October 2004), which is based on version MT199937(99/10/29) of the original Mersenne Twister algorithm found at http://www.math.keio.ac.jp/matumoto/emt.html.

[3] The authors realize that due to space limitations it is hard to make out the subtle differences in the results on the given figures. For a more detailed overview, please contact the authors.

We generated 100,000 different networks (of different sizes and densities), ran the SENT algorithm on each network and recorded the topology to which the network converged (if applicable) and how long this convergence took. We observed that the ring structure proved extremely dominant in our set of experiments. A possible explanation for this observation is that the algorithm is prone to minimizing the degree of each node: groups of low-degree nodes can only become uncovered if they form a ring. Looking at convergence times, we found that for the ring the required time increases faster with each increment of the network size than for the other shapes: in other words, rings take the longest time to shape. We also observed that the outcome of an experiment is more determined by the network size than the network density – except for the small-sized networks.

Acknowledgments The research for this project was funded by the NLDA within the research programme 'Situational Awareness and Optimal Deployment' conducted by the Faculty for Military Sciences.

References

[1] A.L. Barabási. *Linked: How Everything Is Connected to Everything Else and What It Means*. PLUME, 2000.

[2] National Research Council Committee on Network Science for Future Army Applications. *Network Science*. The National Academies Press, 2005.

[3] B. Dutta and M.O. Jackson, editors. *Networks and Groups, Models of Strategic Formation*. Studies in Economic Design. Springer, 2003.

[4] M.O. Jackson. *Social and Economic Networks*. Princeton University Press, 2009.

[5] H. Monsuur. Assessing situation awareness in networks of cooperating entities: A mathematical approach. *Military Operations Research*, 12(3):5–15, 2007.

[6] H. Monsuur. Stable and emergent network topologies: A structural approach. *European Journal of Operational Research*, 183(1):432–441, 2007.

[7] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[8] D.J. Watts. *Small Worlds, The Dynamics of Networks Between Order and Randomness*. Princeton Studies in Complexity. Princeton University Press, 2003.

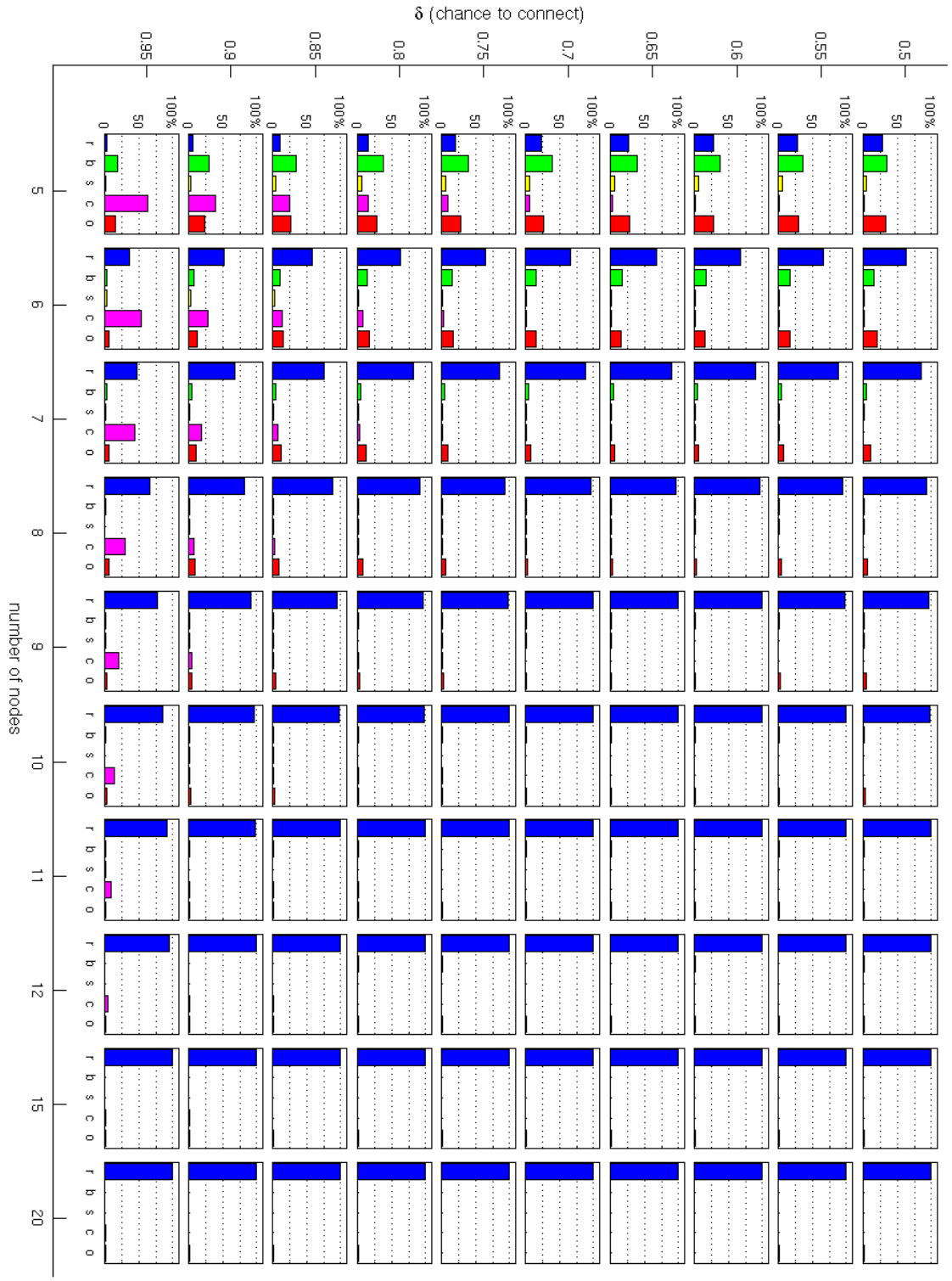

Figure 4: End shape distributions shown in percentages for the following end shapes: (r)ing, (u)ni-polar, (b)i-polar, (c)omplete and (o)ther — a combination of $MAX_{passive}$ and $MAX_{iterations}$.

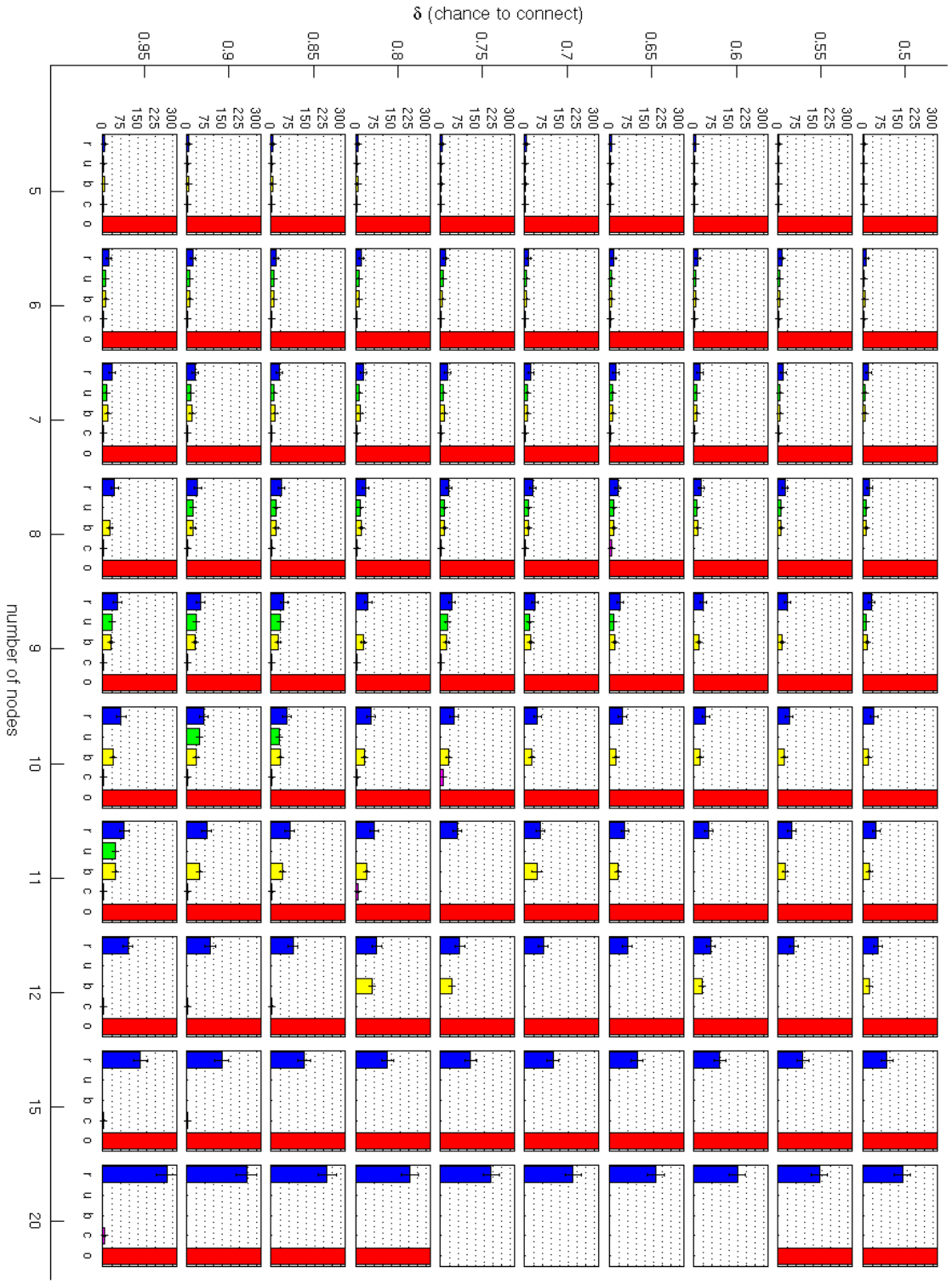

Figure 5: Average number of iterations required to reach the following end shapes: (r)ing, (u)ni-polar, (b)i-polar, (c)omplete and (o)ther — a combination of $MAX_{passive}$ and $MAX_{iterations}$. Notice that we have cut off the last bar — since it terminates after at least 5000 iterations — to allow for better readability.