

APLICAÇÃO DE MODELOS DE AGENTES E REDES COMPLEXAS
NA CARACTERIZAÇÃO E ESTUDO DA INTERAÇÃO ENTRE
INSETOS SOCIAIS, COM ENFOQUE EM FORMIGAS

Murilo Dantas de Miranda

Dissertação apresentado à Universidade Estadual Paulista “Júlio de Mesquita Filho” para a obtenção do título de Mestre em Biometria.

BOTUCATU
São Paulo - Brasil
Março – 2012

**APLICAÇÃO DE MODELOS DE AGENTES E REDES COMPLEXAS
NA CARACTERIZAÇÃO E ESTUDO DA INTERAÇÃO ENTRE
INSETOS SOCIAIS, COM ENFOQUE EM FORMIGAS**

Murilo Dantas de Miranda

Orientadora: Prof^a. Dra. **Claudia Pio Ferreira**

Dissertação apresentado à Universidade Estadual Paulista “Júlio de Mesquita Filho” para a obtenção do título de Mestre em Biometria.

BOTUCATU
São Paulo - Brasil
Março – 2012

FICHA CATALOGRÁFICA ELABORADA PELA SEÇÃO DE AQUIS. E TRAT. DA INFORMAÇÃO
DIVISÃO TÉCNICA DE BIBLIOTECA E DOCUMENTAÇÃO - CAMPUS DE BOTUCATU - UNESP
BIBLIOTECÁRIA RESPONSÁVEL: **ROSEMEIRE APARECIDA VICENTE**

Miranda, Murilo Dantas de.

Aplicação de modelos de agentes e redes complexas na caracterização e estudo da interação entre insetos sociais, com enfoque em formigas / Murilo Dantas de Miranda. – Botucatu : [s.n.], 2012

Dissertação (mestrado) – Universidade Estadual Paulista, Instituto de Biociências de Botucatu

Orientador: Claudia Pio Ferreira

Capes: 90194000

1. Inseto. 2. Comportamento - Avaliação. 2. Formiga – Identificação.

Palavras-chave: Autômatos celulares; *Camponotus senex*; Rede aleatória; Rede mundo pequeno.



UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Botucatu



MEMBROS DA COMISSÃO JULGADORA DA DISSERTAÇÃO DE MESTRADO DE **MURILO DANTAS DE MIRANDA**, INTITULADA **APLICAÇÃO DE MODELOS DE AGENTES E REDES COMPLEXAS NA CARACTERIZAÇÃO E ESTUDO DA INTERAÇÃO ENTRE INSETOS SOCIAIS, COM ENFOQUE EM FORMIGAS**, APRESENTADA AO INSTITUTO DE BIOCIÊNCIAS, UNESP, CAMPUS DE BOTUCATU, SÃO PAULO, EM 01 DE MARÇO DE 2012.

APROVADA PELA COMISSÃO JULGADORA:


Profa. Dra. CLAUDIA PIO FERREIRA


Prof. Dr. PAULO FERNANDO DE A MANCERA


Prof. Dr. MARCIO ROBERTO PIE

Dedico a painho e as mainhas.

Agradecimentos

Agradeço a Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) (Processo 2010/04671-9).

À Profa. Dra. Claudia P. Ferreira, minha orientadora, pelo voto de confiança, pela fundamental contribuição no meu crescimento enquanto pesquisador.

Ao Programa de Pós-Graduação em Biometria do Instituto de Biociências da Universidade Estadual Paulista “Júlio de Mesquita Filho” de Botucatu e a seus professores, na pessoa da coordenadora Profa. Dra. Luzia A. Trinca, pela oportunidade de concretizar meu crescimento científico e profissional.

Ao laboratório de Dinâmica Evolutiva e Sistemas Complexos da Universidade Federal do Paraná, principalmente ao Prof. Dr. Marcio R. Pie, aos mestrandos Felipe M. Neves e Marcelo Borges.

Aos membros das bancas de qualificação e defesa.

Aos meus amigos baianos, em especial MSc. Fabricio Gama, MSc. Saymon H. S. Santana, MSc. Jailson dos S. Novais, MSc. João R. Silva que desse a graduação me dão suporte na realização dos trabalhos e pela nossa amizade.

Aos meus amigos paulistas e mineiros, em especial as irmãs Gomes (Vivian e Juliana) pelo acolhimento na cidade, Danilo S. Ré e Ana L. Pavan.

A toda minha família, em especial a minha irmã Marcia G. D. de Miranda e a minha sobrinha Danielly D. M. dos Santos, pelo apoio em muitos momentos de minha vida, e ao meu primo insuportável Sérgio M. M. Bittencourt. Também agradeço a minha avó, meus tios e tias, primos e todos que foram fundamentais nesta minha trajetória.

Aos meus amigos do Laboratório de Entomologia da Universidade Estadual de Feira de Santana, principalmente Patrícia L. Oliveira-Rebouças e Miriam Gimenes.

A todas as pessoas que, direta ou indiretamente, contribuíram para a concretização deste trabalho.

*“All the boys, all the girls,
All the mess in the world
All the boys, all the girls,
All the mess that occurs.
All the high’s, all the low’s”
Coldplay - Charlie Brown*

Sumário

	Página
LISTA DE FIGURAS	vii
LISTA DE TABELAS	ix
RESUMO	x
SUMMARY	xii
1 INSETOS SOCIAIS: MOTIVAÇÃO BIOLÓGICA	1
1.1 Introdução	1
1.2 Metodologia	3
1.2.1 Autômatos Celulares	3
1.2.2 Redes Complexas	6
2 AUTÔMATOS CELULARES	8
2.1 Introdução	8
2.2 Composição	9
2.2.1 Tipos de redes	9
2.2.2 Tipos de células	10
2.2.3 Tipos de vizinhança	11
2.2.4 Regras de atualização	11
2.2.5 Condições de contorno	12
2.2.6 Condições iniciais	13
2.3 Classificação	13
3 REDES COMPLEXAS	16
3.1 Introdução	16

3.2	Conceitos Básicos	18
3.3	Topologias de Rede	19
3.3.1	Distribuição de Grau	19
3.3.2	Coeficiente de Agrupamento	19
3.3.3	Caminho Mínimo Médio	20
3.4	Tipos de Rede	20
3.4.1	Rede Regular	20
3.4.2	Rede Aleatória	20
3.4.3	Rede Mundo Pequeno	21
3.4.4	Rede Livre de Escala	22
4	RESULTADOS E DISCUSSÃO	25
4.1	Redes Complexas	25
4.2	Autômatos Celulares	34
4.2.1	Caracterização da rede de interação do ato comportamental de ativação- inativação em formigas	34
4.2.2	Evolução temporal da rede de interação do ato comportamental de ativação-inativação para duas castas de formigas	39
5	CONCLUSÕES	48
	REFERÊNCIAS BIBLIOGRÁFICAS	49
	APÊNDICES	55

Lista de Figuras

	Página
1 Tipos de redes utilizadas em modelos de autômatos celulares (AC)	10
2 Tipos de células utilizadas em modelos de AC	10
3 Tipos de vizinhança utilizadas em modelos de AC	11
4 Tipos de condições de contorno utilizadas em modelos de AC	12
5 Tipos de condições iniciais utilizadas em modelos de AC	13
6 Padrões espaço-temporais de Wolfram (1986)	14
7 Diferentes rede geradas pelo algoritmo de Watts & Strogatz (1998) . . .	23
8 Exemplo de rede regular	25
9 Rede aleatória geradas pelo algoritmo de Albert & Barabási (2002) . . .	26
10 Distribuição dos graus das redes aleatórias geradas pelo algoritmo de Barabási & Albert (1999)	27
11 Distribuição dos graus das redes aleatórias geradas pelo algoritmo de Albert & Barabási (2002)	27
12 Rede de mundo pequeno gerada pelo algoritmo de Watts & Strogatz (1998)	28
13 Distribuição dos graus da rede de mundo pequeno gerada pelo algoritmo de Watts & Strogatz (1998)	29
14 Caminho mínimo médio e coeficiente de agrupamento como funções da probabilidade de realocação de conexões p (algoritmo de Watts & Strogatz (1998))	30
15 Exemplo de rede livre de escala	31
16 Distribuição dos graus dos nós de uma rede livre de escala	31

17	Desenho esquemático da distribuição da informação em uma rede com 10 nós	32
18	Distribuição da informação para rede aleatória de Erdős & Rényi (1960)	33
19	Distribuição da informação para a rede mundo pequeno com $p = 0,1$. . .	33
20	Distribuição da informação para a rede livre de escala	34
21	Grau médio em função da densidade de agentes na rede	36
22	Diferentes valores do caminho mínimo médio e do coeficiente de agrupamento em função da densidade de agentes na rede para os modelos com condição de contorno de valores fixos e periódica	37
23	Relação do período de sincronização e a razão sinal-ruído em função da densidade de agentes na rede para os modelos com condição de contorno de valores fixos e periódica	38
24	Evolução temporal dos valores do grau médio e do caminho mínimo médio para a colônia e rainhas	40
25	Diferentes topologias na casta das rainhas	41
26	Distribuições dos graus de interação da colônia	42
27	Distribuições dos graus de interação para a casta das rainhas	42
28	Evolução temporal da distribuição dos graus da rede de interações da formiga <i>T. rugatulus</i>	45
29	Período de sincronização e razão sinal-ruído em função do tempo	47

Lista de Tabelas

	Página
1	Frequência absoluta do ato comportamental “allogrooming” para <i>C. senex</i> 6
2	Caminho mínimo médio e coeficiente de agrupamentos médio para as redes de mundo pequeno geradas com diferentes valores de p 29
3	Valores dos parâmetros ajustados para colônia utilizando diferentes dis- tribuições de probabilidade 43
4	Valores dos parâmetros ajustados para a casta das rainhas utilizando diferentes distribuições de probabilidade 44

APLICAÇÃO DE MODELOS DE AGENTES E REDES COMPLEXAS NA CARACTERIZAÇÃO E ESTUDO DA INTERAÇÃO ENTRE INSETOS SOCIAIS, COM ENFOQUE EM FORMIGAS

Autor: MURILO DANTAS DE MIRANDA

Orientadora: Prof^ª. Dra. CLAUDIA PIO FERREIRA

RESUMO

Insetos sociais são conhecidos pela capacidade que possuem de gerar comportamentos coletivos robustos a partir de informações limitadas ao nível de indivíduo, sem que haja um controle central. Esta propriedade de geração de padrões a partir de regras locais é conhecida como auto-organização. As formigas são um exemplo de sucesso ecológico e o único grupo exclusivamente eusocial dentro de Hymenoptera. Devido a sua abundância e diversidade de hábitos alimentares exercem efeitos importantes na maioria dos ecossistemas terrestres. Dentre os vários tipos de interação que ocorrem entre formigas, a limpeza mútua (*allogrooming*) está entre os principais atos comportamentais, onde membros de determinada espécie executam a limpeza em outros indivíduos pertencente aos seu grupo social. Nesta tese buscamos investigar as redes de interações geradas pelo modelo proposto por Miramontes et al. (1993), o qual simula o comportamento individual de formigas, mais especificamente processos de ativação e inativação gerados por interações locais, as quais levam a sincronização temporal de atividades da colônia. Uma extensão do modelo Miramontes et al. (1993) para duas castas de formigas tecelãs foi construída. Nesse novo modelo foi utilizado dados etológicos de limpeza mútua (*allogrooming*) obtidos do trabalho de Blonder & Dornhaus (2011), e o ato de ativação e inativação das castas e da colônia, como descrito no modelo de Miramontes et al. (1993). Os resultados obtidos sugerem que a densidade da colônia está entre 0,20 a 0,50, pois nesse intervalo a colônia tem altos valores de coeficiente de agrupamento e razão sinal-ruído, e baixos valores de caminho médio mínimo e período de sincronização.

Temos ainda que a topologia das interações para o ato *allogrooming* da colônia muda com o tempo, inicialmente a distribuição dos graus segue uma lei de potência (a qual caracteriza uma rede livre de escala) e depois uma distribuição de Poisson.

Palavras-chave: rede aleatória, rede mundo pequeno, autômatos celulares, *Camponotus senex*

APPLYING AGENT BASED MODELS AND COMPLEX NETWORKS TO STUDY AND CHARACTERIZE INSECT SOCIAL NETWORKS, EMPHASIZE ON ANTS

Author: MURILO DANTAS DE MIRANDA

Adviser: Prof^a. Dra. CLAUDIA PIO FERREIRA

SUMMARY

Social insects are known to have the ability to generate robust collective behavior from the limited information at the individual level, without central control. This property of generating patterns from local rules is known as self-organization. The aims of this study were to characterize the networks of interactions generated by the model proposed by Miramontes et al. (1993), which simulates individual behavior of ants, more specifically the activation and inactivation processes generated by local interactions, which cause the temporal synchronization of activities of the colony. A second model was proposed from the model Miramontes et al. (1993) for two castes of weaver ants, in this model was used ethological data of allogrooming and the act of activation and inactivation of caste and colony. The results show that the density of the colony is between 0.20 to 0.50, the colony as this range has high clustering coefficient values and signal to noise ratio, and low values of average path length and minimum synchronization period. We still have the topology of interactions for the act allogrooming of the colony changes with time, initially the distribution of degrees follows a power law, then a Poisson distribution. The model indicates that the initial network behavior generated by the act allogrooming is the type of scale free.

Keywords: random network, small-world network, cellular automata, *Camponotus senex*

1 INSETOS SOCIAIS: MOTIVAÇÃO BIOLÓGICA

1.1 Introdução

Uma interação envolve dois ou mais indivíduos e um ou mais atos comportamentais, essas interações entre indivíduos são elementos básicos na organização social (Hinde, 1976). A organização social é resultante das interações locais entre os indivíduos e seu ambiente, sendo definida como uma classe importante de relações ecológicas entre os coespecíficos, podendo incluir as relações de competição, cooperação e dominância na aquisição de recursos ou companheiros (Wilson, 1975). É também importante no contexto de biologia populacional, influenciando o fluxo de gene e padrão espacial de cada espécie (Wilson, 1975; Whitehead, 1997). Na análise da organização social das espécies é muito importante utilizar uma ferramenta sofisticada como, por exemplo, as redes complexas para responder a perguntas, como: quais as implicações na organização social para a evolução na escolha do companheiro, nas estratégias de aprendizagem e na transmissão de doenças? Como a organização social permite que os animais se adaptem as mudanças ambientais? (Krause et al., 2007; Wey et al., 2008).

A organização social ou sociabilidade nos insetos é dividido pelos entomólogos em insetos eusociais, que cooperam na reprodução e divisão do trabalho reprodutivo, e insetos subsociais, que possuem hábitos sociais não tão fortemente desenvolvidos, tendo uma extensão menor de cooperação e divisão do trabalho reprodutivo (Gullan & Cranston, 2008). A eusocialidade é restrita as ordens Hymenoptera (formigas, algumas abelhas e vespas) e Isoptera (cupins).

Os insetos sociais são conhecidos pela capacidade de gerar comportamentos coletivos robustos a partir de informações limitadas a nível de indivíduo, sem que haja um controle central. Esta propriedade de geração de padrões globais a partir de regras locais é conhecida com auto-organização (Bonabeau et al., 1997). A investigação de fenômenos coletivos em sistemas biológicos é uma tarefa muito difícil e a modelagem matemática, aliada a dados observacionais, é uma ferramenta poderosa no entendimento e caracterização destes comportamentos.

Dentre os diversos trabalhos na área de modelagem matemática e computacional utilizando os insetos sociais como motivação biológica (Solé et al., 1993; Naug & Camazine, 2002; Fewell, 2003) podemos citar aqueles que estudaram o comportamento das formigas, desde o fluxo de informação na colônia (Blonder & Dornhaus, 2011) e formação de trilhas (Edelstein-Keshet et al., 1995; Watmough & Edelstein-Keshet, 1995) até sincronização de atos comportamentais (Franks et al., 1990; Cole, 1991, 1992).

Segundo Hölldobler & Wilson (1990), as formigas representam uma ramificação evolutiva que obteve muito sucesso ecológico. Este sucesso pode estar associado com o estilo de vida predatório social (Wilson, 1987; Hölldobler & Wilson, 1990). Além disso, a sua grande diversidade se deve a enorme variedade de habitats de nidificação, preferências alimentares e comportamento social com divisão de trabalho (Wilson, 1987; Hölldobler & Wilson, 1990).

Dos vários tipos de interações que ocorrem com as formigas, a limpeza mútua (“allogrooming”) está entre os principais atos comportamentais. Entendemos como limpeza mútua o comportamento no qual o indivíduo remove os parasitas dos outros indivíduos da colônia, tornando-a mais resistente a patógenos. O padrão temporal dos atos comportamentais é uma das características fundamentais de qualquer ato comportamental principalmente para sua compreensão como um mecanismo adaptativo (Cole, 1991).

Os objetivos desse trabalho são caracterizar as redes de interações geradas pelo modelo baseado em agentes proposto por Miramontes et al. (1993), que

simula o comportamento individual de formigas (supondo uma única casta), mais especificamente processos de ativação e inativação causados por interações locais; modificar este modelo incluindo duas castas (operárias e rainhas), e dados etológicos de limpeza mútua da formiga tecelã *Camponotus senex* Smith (1858).

1.2 Metodologia

As técnicas computacionais e matemáticas utilizadas no estudo do comportamento de limpeza mútua (“allogrooming”) entre formigas foram os autômatos celulares e redes complexas. A seguir comentamos as implementações realizadas neste trabalho. Posteriormente, faremos uma introdução ao formalismo de autômatos celulares e redes complexas.

1.2.1 Autômatos Celulares

O modelo desenvolvido por Miramontes et al. (1993) tem por objetivo reproduzir o ato comportamental de ativação e inativação dos indivíduos de uma colônia de formigas. Para isso, eles propuseram um autômato celular bidimensional probabilístico de tamanho linear $N = 10$ com dois estados (agente ativo ou inativo) e condição de contorno fixa. Cada autômato representa um indivíduo da colônia o qual interage localmente com seus vizinhos mais próximos (vizinhança de Moore).

Neste modelo, cada agente a_i na rede é caracterizado por três quantidades:

$$a_i = \{x_i, y_i, m_i\}, \quad (1)$$

em que x_i e y_i são as coordenadas do agente na rede e m_i é a variável de estado (agente ativo ou inativo), a qual é dada pela função de Heaviside (função degrau)

$$m_i = \begin{cases} 1, & \text{se } S_i > 0 \\ 0, & \text{caso contrário} \end{cases}, \quad (2)$$

em que S_i representa o valor de atividade do agente e é dada pela seguinte expressão:

$$S_i^{t+1} = \tanh \left[g \left(\sum_{j=1}^k J_{ij} S_j^t + J_{ii} S_i^t \right) \right], \quad (3)$$

sendo g um parâmetro que permite controlar o ângulo da função tangente hiperbólica, k o número de vizinhos e t o tempo. A função tangente hiperbólica limita os valores de S_i entre -1 e 1. O parâmetro J_{ii} representa a interação do indivíduo com ele mesmo e os demais J_{ij} são os acoplamentos obtidos a partir da matriz

$$J = \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix},$$

em que J_{11} , J_{12} , J_{21} e J_{22} representam, respectivamente, a interação entre agentes ativo-ativo, ativo-inativo, inativo-ativo e inativo-inativo. Essa matriz é muito importante, pois contém informações sobre a biologia dos insetos sociais, tais como a frequência dos atos comportamentais das diferentes castas (operárias, rainhas e soldados).

Indivíduos ativos podem se movimentar caso haja em sua vizinhança sítios vazios, e os indivíduos inativos têm probabilidade $\epsilon=0,01$ de ficarem ativos (ativação espontânea). A condição inicial é gerada aleatoriamente, com a_i indivíduos distribuídos na rede, cada qual com um valor de S_i^0 . Os resultados apresentados no artigo, foram obtidos utilizando $J_{11} = J_{12} = J_{21} = J_{22} = 1$ e $g=0,05$. Este modelo reproduziu bem o padrão do ato comportamental de ativação e inativação observado no artigo experimental de Cole (1991).

Para caracterizar a rede de interação do modelo proposto por Miramontes et al. (1993) construímos a matriz de incidência (após 1000 iterações), I , a qual determina se há ou não interação entre os agentes. Assim, se os agentes i e j (vizinhos) estão ativos temos que $I_{ij} = 1$ senão $I_{ij} = 0$ sendo $i \neq j$ e $i, j = 1, \dots, Q$, em que Q é o número total de agentes na rede. Utilizamos condição de contorno fixa (apresentaremos resultados utilizando também condição de contorno periódica), atualização paralela e rede de tamanho linear $N = 25$. Para esse modelo foram realizadas 25 simulações.

Foram feitas medidas relativas à topologia da rede, tais como coeficiente de agrupamento, caminho médio, grau médio e distribuição dos graus. Os resultados obtidos foram comparados com os valores esperados para redes aleatórias com mesmo grau médio. As distribuições dos graus foram ajustadas para as distribuições: Normal, Exponencial discreta, Poisson e de Lei de potência, e para selecionar o melhor ajuste foi utilizado o critério de Informação de Akaike corrigido (AICc) (Burnham & Anderson, 2002). Os ajustes das distribuições dos graus e as figuras das redes geradas pelas simulações foram feitas no programa R 2.13.0 (R Development Core Team, 2011), somente para os melhores ajustes foram construídos os gráficos das distribuições dos graus. Também foram calculados o período de sincronização do comportamento de ativação-inativação do formigueiro, p , utilizando a transformada rápida de Fourier nos dados de séries temporais. A razão sinal-ruído foi definida como $SNR = \bar{p}/(\bar{p} + \sigma)$, em que $\bar{p}$ é o período médio e σ é o desvio padrão obtido das simulações. Quando o sinal é puro ($\sigma = 0$), $SNR = 1$.

Também modificamos o modelo anterior, com o objetivo de considerar dois tipos de castas na colônia, ou seja, cada agente a_i foi caracterizado por $a_i = \{x_i, y_i, m_i, C_i\}$, em que C_i representa o tipo de casta. Foram utilizados 200 agentes, sendo 20 rainhas e 180 operárias, distribuídos, inicialmente, de maneira aleatória em uma rede bidimensional de tamanho linear $N = 40$. Esses valores foram escolhidos a partir do trabalho de Santos et al. (2005), na tentativa de manter os mesmos padrões utilizados no experimento. Foram realizadas $M = 10$ simulações e, após o transiente ($t \geq 1000$), a cada 300 iterações até $t = 5800$, foram feitas diversas medidas para estudar a evolução temporal da topologia da rede de interações.

Neste caso, construímos a matriz de incidência da seguinte forma: inicialmente é verificado se os agentes i e j vizinhos estão ativos, se ativos é verificado se os agentes são da mesma casta ou não, então é sorteado um número da distribuição uniforme entre 0 e 1, e, se esse número for menor do que o valor de probabilidade de interação entre os agentes, então temos que $I_{ij} = 1$ senão $I_{ij} = 0$ sendo $i \neq j$ e $i, j = 1, \dots, Q$, em que Q é o número total de agentes na rede. Os valores de

probabilidade de interação entre os agentes foram obtidos a partir dos dados do comportamento de limpeza mútua das formigas tecelãs *Camponotus senex* apresentados no trabalho de Santos et al. (2005). Usando a razão entre a frequência absoluta do comportamento de limpeza mútua (Tabela 1) e da quantidade total das interações (19900 interações) de uma rede completa com 200 agentes, o segundo modelo recebeu as seguintes informações nos valores de probabilidade de interação: operária interage com operária com probabilidade de 0,0123, operária interage com rainha com probabilidade de 0,0086 e rainha interage somente com operária com probabilidade de 0,0036. Assim sendo, os elementos da matriz de incidência dependem do estado do agente e da probabilidade de interação entre os agentes vizinhos ativos.

Tabela 1: Frequência absoluta do ato comportamental “allogrooming” para *C. senex* (Santos et al., 2005).

Ato comportamental	Operária	Rainha
“Allogrooming” operária	244	172
“Allogrooming” rainha	71	0

1.2.2 Redes Complexas

Os códigos implementados na geração das redes complexas estudadas nesse trabalho foram desenvolvidos em linguagem de programação C. Foram realizadas 20 simulações para cada tipo de rede complexa, as quais foram caracterizadas através de medidas relativas à sua topologia, tais como coeficiente de agrupamento, caminho médio, grau médio e distribuição dos graus. Este estudo permitiu a construção de algoritmos para caracterizar as redes de interações dos modelos anteriormente descritos.

Também medimos a velocidade de distribuição de informação em cada tipo de rede da seguinte maneira: um nó da rede foi escolhido aleatoriamente para receber uma informação, e a partir desse momento foi observado o tempo necessário para que toda rede esteja com essa informação. O objetivo foi relacionar e discutir

o tipo de rede de interação com o controle de doenças, a distribuição de feromônios de alerta e a estabilidade da rede frente a pequenas perturbações.

2 AUTÔMATOS CELULARES

2.1 Introdução

Os autômatos celulares (AC) são modelos matemáticos usados para estudar sistemas complexos que possuem um grande número de entidades (autômatos) simples e idênticas. A noção de autômatos celulares teve origem nos trabalhos de John von Neumann e Stanislaw Ulam. Um dos primeiros modelos de autômatos celulares foi criado por volta de 1952, por John von Neumann, no contexto de auto-replicação. Entretanto, o desenvolvimento prático se deu com Conway nos anos 60 com o desenvolvimento do Jogo da Vida, que representava um sistema de vida e morte entre indivíduos (Deutsch & Dormann, 2005).

Nos últimos anos, os autômatos celulares foram propostos para um grande número de aplicações biológicas (Ermentrout & Edelstein-Keshet, 1993), incluindo as ecológicas (Bagnoli & Bezzi, 1998; Cannas et al., 1999), epidemiológicas (Schönfisch, 1995), etológicas (Solé et al., 1993; Herz, 1994), evolutivas (Bagnoli & Bezzi, 1998), e imunológicas (Ahmed, 1996; Bezzi et al., 1997; Meyer-Hermann, 2002; Mallet & De Pillis, 2006).

De maneira simplificada, podemos descrever o autômato celular como uma rede discreta na qual cada sítio (ou célula) aloca um autômato e a cada autômato está associado a valores discretos dentre um conjunto finito que definem seu estado, estes autômatos interagem entre si através de regras bem definidas. As células podem ser vistas como os locais onde os indivíduos vivem ou, algumas vezes, como os próprios indivíduos. Elas podem ser interpretadas como células biológicas, uma população de peixes em um lago, como territórios de aves e assim por diante (de Vries

et al., 2006).

As regras de transição entre estados (regras de atualização) que regem a evolução do sistema ocorrem mediante passos de tempo discretos e são definidas por um conjunto de elementos denominado vizinhança. A cada passo de tempo, os sítios são atualizados de acordo com as regras de evolução do sistema. O valor assumido por cada sítio pode, por exemplo, depender do seu valor e dos valores dos sítios conectados a ele no passo de tempo anterior com alguma probabilidade associada, neste caso a atualização é dita síncrona e o autômato probabilístico.

Por fim, os autômatos celulares, assim como outros modelos baseados em agentes, são interessantes para os biólogos, por diversas razões. Em primeiro lugar, muitas estruturas na biologia são discretas, e um modelo natural para descrever tais estruturas seria um sistema discreto (por exemplo, indivíduos de uma população). Em segundo lugar, as regras para o nascimento, morte ou migração pode ser especificada de forma direta. Em terceiro lugar, o tempo e os padrões dos autômatos celulares podem ser interpretados diretamente em termos biológicos. Em suma, modelos de autômatos são sistemas dinâmicos discretos capazes de gerar padrões de alto nível de complexidade.

2.2 Composição

2.2.1 Tipos de redes

A rede do AC pode ser representado por um vetor ou matriz de comprimento finito e os elementos constituintes são os sítios, que representam cada posição do vetor ou matrizes os quais os autômatos estão dispostos. As redes podem possuir qualquer número de dimensão, contudo é mais comum o uso de redes unidimensionais (vetor), bidimensionais (matriz) ou tridimensionais (matriz) (Figura 1). As redes também podem ser regulares ou irregulares e sua escolha depende, unicamente, do problema a ser tratado.

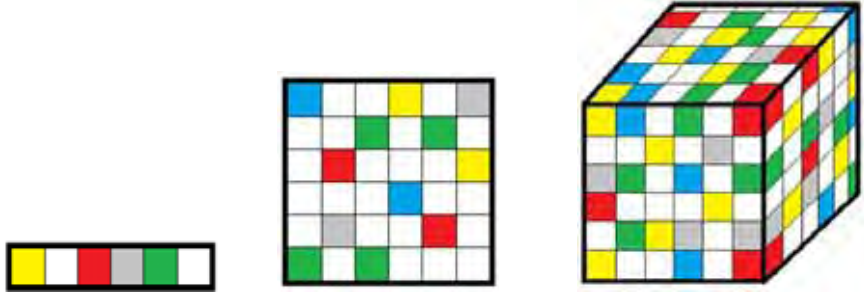


Figura 1: Exemplos de rede utilizadas nos modelos de AC, da esquerda para direita temos representado uma rede unidimensional, bidimensional e tridimensional.

2.2.2 Tipos de células

As células ou sítios σ_i , sendo $i = 1, \dots, N$ e N o número de sítios da rede, são elementos básicos da rede L sobre a qual a dinâmica acontece. Possuem a função de memória, pois alocam as entidades e guardam valores de propriedades das mesmas. Dependendo da modelagem proposta de um sistema, a forma geométrica de cada célula pode assumir diversas configurações. No entanto, o mais comum é o uso de células regulares como célula quadrada, triangular e hexagonal e, de mesmo tamanho (Figura 2). Sendo a quadrada a forma mais comum devido à sua simplicidade na implementação.



Figura 2: Exemplos de tipos de célula utilizadas no autômato celular, na esquerda célula quadrada e na direita célula triangular.

2.2.3 Tipos de vizinhança

A vizinhança é uma coleção N_i de elementos usada para determinar o estado do sítio alvo σ_i após o processo de atualização. Ela pode ser local (elementos adjacentes) ou não, pode ser previamente definida ou arbitrária (randômica), pode conter o sítio alvo ou não. Uma vizinhança pode assumir qualquer configuração espacial e sua escolha depende do processo a ser modelado. Os tipos de vizinhança mais utilizados são a vizinhança de von Neumann e a de Moore. A vizinhança de von Neumann considera apenas as quatro células mais próximas como vizinhas da célula que será atualizada, e a vizinhança de Moore considera as oito células vizinhas mais próximas (Figura 3).

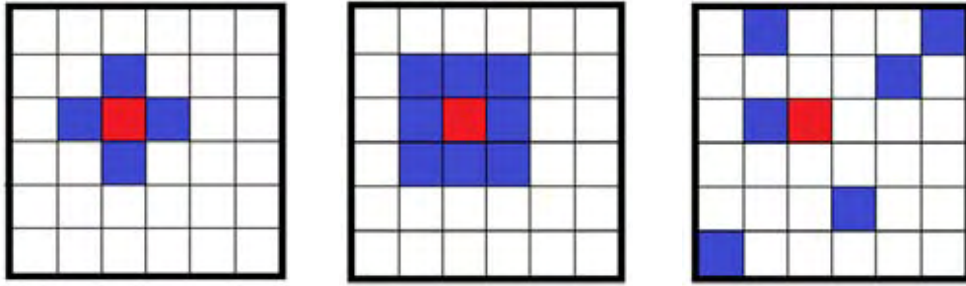


Figura 3: Exemplos de tipos de vizinhança de uma célula, da esquerda para direita temos a vizinhança de Neumann, vizinhança de Moore e a vizinhança aleatória.

2.2.4 Regras de atualização

São as regras ϕ que definem o estado σ_i do sítio alvo de atualização no passo de tempo seguinte através dos estados dos sítios σ_j contidos em sua vizinhança N_i . Quando a todos os sítios da rede se aplica a mesma regra de atualização, o modelo é dito homogêneo. Assim,

$$\sigma_i(t+1) = \phi(\sigma_j(t) \in N_i).$$

Dado o conjunto de estados possíveis para um sítio $\Sigma \equiv \{0, 1, 2, \dots, k-1\}$ e o tamanho da vizinhança (i elementos), o número de configurações possíveis para a

vizinhança será k^i e o número de regras possíveis será k^{k^i} . Como a rede L possui N elementos, ela apresenta k^N configurações possíveis.

A atualização pode ocorrer de forma síncrona (paralela - as regras são aplicadas simultaneamente a todos os sítios) ou sequencial (as regras são aplicadas em cada sítio um após o outro).

2.2.5 Condições de contorno

Durante a implementação computacional, um limite para o tamanho da rede é imposto e torna-se necessário determinar condições de contorno de acordo com o problema proposto. As condições de contorno mais utilizadas são (Figura 4):

- Contorno periódico: a borda de um dos lados da rede recebe o valor do último sítio do lado oposto. Dessa forma, uma rede unidimensional com contorno periódico torna-se uma fita fechada sobre si mesma e uma rede bidimensional torna-se um toróide;
- Contorno reflexivo: os valores dos sítios que ocupam o contorno refletem os valores dos sítios que antecedem a borda;
- Contorno de valores fixos: ocorre quando são determinados valores para o contorno que são constantes durante toda a simulação.

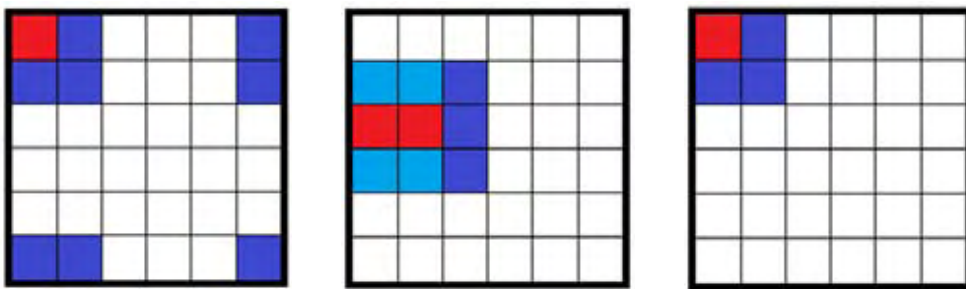


Figura 4: Exemplos de tipos de condições de contorno da rede, da esquerda para direita temos o contorno periódico, o reflexivo e o de valores fixos.

2.2.6 Condições iniciais

São os valores atribuídos aos estados dos sítios no começo da simulação ($t = 0$) e o conjunto paramétrico adotado. Em modelos de autômatos celulares, pequenas variações nas condições iniciais são capazes de produzir efeitos drásticos após poucos passos de tempo (Figura 5).



Figura 5: Exemplos de tipos de condições iniciais da rede, da esquerda para direita temos uma condição inicial aleatória e uma condição fixa definida pelo usuário.

2.3 Classificação

Com base nos padrões espaço-temporais de evolução dos sistemas, (Wolfram, 1986) propôs uma classificação (Figura 6):

- Classe I (comportamento fixo): A evolução temporal leva todos os autômatos ao mesmo estado, ou seja, o sistema converge para uma única solução. Em analogia aos modelos de equações diferenciais ordinárias (EDO) essa primeira classe corresponde a um modelo contínuo cuja solução converge para um ponto fixo estável;
- Classe II (comportamento cíclico ou periódico): A evolução temporal leva a estabilidade e a periodicidade do sistema. Há a emergência de estruturas espacialmente separadas e que não transitam pela rede. O análogo de sistemas dinâmicos contínuos corresponderia a modelos cujas trajetórias no espaço de configurações converge para um ciclo limite;

- Classe III (comportamento caótico): A evolução leva a uma sucessão aleatória de padrões caóticos não-periódicos. Apresenta um dinâmica rica, porém não há comportamento emergente. O comportamento dessa classe de autômatos celulares é análogo ao comportamento caótico observado em sistemas de equações diferenciais não-lineares;
- Classe IV (comportamento complexo): A evolução leva a formação de estruturas espacialmente separadas que se movimentam de maneira imprevisível e formam padrões de alta complexidade no espaço e no tempo. Não há análogo em sistemas de EDO.

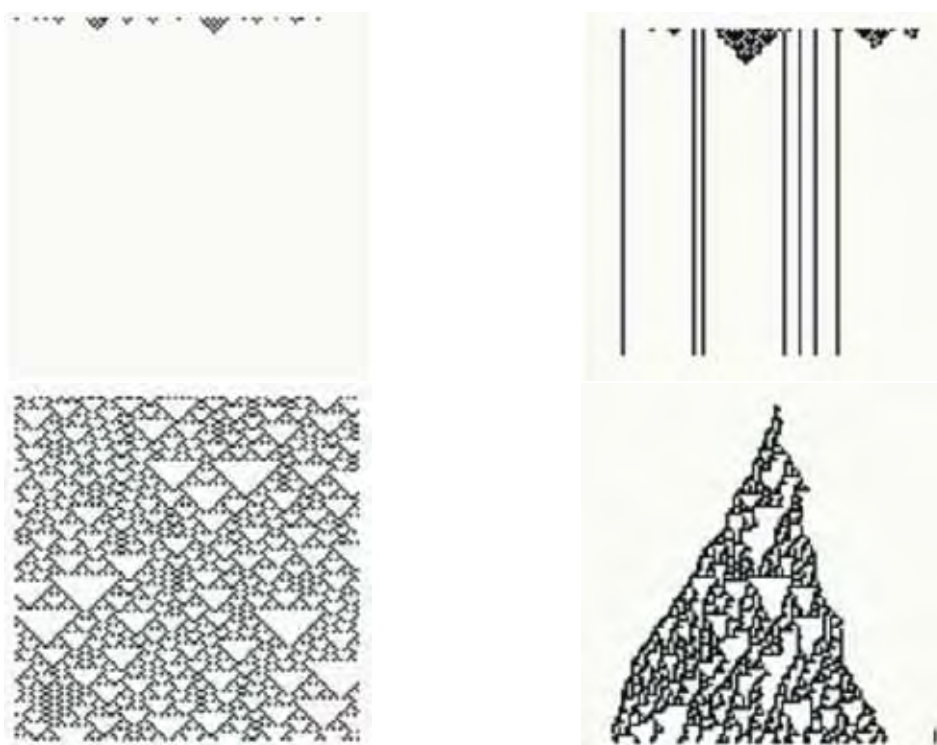


Figura 6: Exemplos de padrões espaço-temporais gerados por modelos de AC (Wolfram, 1986), de cima para baixo e da esquerda para a direita temos, um autômato da classe I, II, III e IV.

Quando se pensa em sistemas biológicos, tem-se que os autômatos celulares dos tipos I e II são muito simples para serem usados na modelagem desses

sistemas, enquanto que os da classe III, apesar de apresentarem uma dinâmica rica, não apresentam nenhum tipo de comportamento emergente, de modo que esse tipo de classe também não é considerado apropriado. Já os classe IV podem gerar estruturas que persistem, ou seja, levam a emergência de ordem a partir de interações locais, uma característica bastante comum em sistemas biológicos.

3 REDES COMPLEXAS

3.1 Introdução

O estudo das redes tem uma longa história com um dos ramos da Matemática Discreta conhecido como Teoria dos Grafos. Desde o seu surgimento em 1736, quando o matemático suíço Leonhard Euler publicou a solução para o problema das Pontes de Königsberg (consiste em encontrar um caminho que passe somente uma única vez em cada pontes da cidade prussiana de Königsberg, hoje Kaliningrad, Rússia). Nos últimos anos, uma série de artigos de revisão (Dorogovtsev & Mendes, 2002; Albert & Barabási, 2002; Newman, 2003; Boccaletti et al., 2006) e livros (Newman et al., 2006; Barrat et al., 2008) sobre redes complexas são encontrados na literatura e seu desenvolvimento se dá conjuntamente com outras áreas como, a Sociologia, a Comunicação, a Física e a Biologia. Embora cada área, em muitos casos, tenha introduzido sua própria nomenclatura, a linguagem formal para a descrição das redes é encontrado na Teoria Matemática de Grafos. Por outro lado, o estudo das grandes redes, como as biológicas, tem estimulado a definição de novas métricas e estatísticas nas redes.

O aumento da capacidade de processamento dos computadores possibilitou o reconhecimento e a manipulação de conjuntos de dados em várias redes de grande escala, permitindo a análise detalhada de suas propriedades estruturais e funcionais. Esses estudos tiveram como motivação o desejo de entender os diversos sistemas reais, que vão desde redes de comunicação até sistemas de cadeias ecológicas. As principais publicações de redes complexas abordam sobre o ambiente *World Wide Web* (Albert et al., 1999), a Internet (Yook et al., 2001), a rede de colaboração de

atores de cinema e entre os cientistas (Watts & Strogatz, 1998; Newman, 2001), a rede de contato sexual humano (Liljeros et al., 2001), as redes celulares (Jeong et al., 2000), as redes ecológicas (Montoya & Solé, 2002), dentre outros.

As redes complexas são classificadas de acordo com o seu contexto, elas são divididas em cinco grupos: as redes biológicas, as redes econômicas, as redes de informação, as redes sociais e as redes tecnológicas. Nesse trabalho daremos ênfase as redes biológicas.

As redes biológicas referem-se aos conjuntos complexos de interações entre genes, proteínas e processos moleculares que regulam a vida biológica, além das redes de reações metabólicas, as redes neuronais e diversos trabalhos na área de Ecologia e Evolução (Jeong et al., 2000; Proulx et al., 2005; Lewinsohn et al., 2006; Bascompte, 2007; Bascompte & Jordano, 2007; Bastolla et al., 2009), ou seja, as redes permeiam o mundo biológico em vários níveis que vão desde a microescala da química biológica, genética e proteômica para a grande escala da cadeia alimentar.

No nível de microescala, muitos aspectos relevantes da complexidade biológica são envolvidas na estrutura e dinâmica das redes emergentes em diferentes níveis organizacionais, desde as vias bioquímicas intracelulares até redes de regulação genética ou redes de interação protéica (Barabási & Oltvai, 2004). Numa escala maior, redes biológicas podem descrever as interações dos indivíduos em várias populações. Nesta área, a biologia pode sobrepor-se com a ciência social. Um exemplo típico é dado pela rede que descreve o contato de relações sexuais que é importante do ponto de vista social e de grande preocupação na epidemiologia das doenças sexualmente transmissíveis (Liljeros et al., 2001). Por fim, em larga escala encontramos as redes descrevendo a cadeia alimentar de ecossistemas inteiros. Essa rede descreve quais as espécies que se alimentam de outras espécies. Nesta perspectiva, os nós representam as espécies e as ligações são interações tróficas antagônicas do tipo presa-predador. Entretanto, a classe das redes biológicas possui um grande problema na obtenção de dados que possam ser utilizados para esses fins (Barrat et al., 2008).

3.2 Conceitos Básicos

Em termos gerais, uma rede é qualquer sistema que admite uma representação matemática abstrata como um grafo, cujos nós (vértices) identificam os elementos do sistema e em que o conjunto de ligações (arestas) representam a presença de uma relação ou interação entre os elementos. Um alto nível de abstração, geralmente, se aplica a uma ampla variedade de sistemas. Nesse sentido, as redes fornecem um quadro teórico que permite uma representação conceitual de interações em sistemas complexos onde a caracterização do nível de sistema implica o mapeamento das interações entre um grande número de indivíduos (Boccaletti et al., 2006).

Um grafo $G = (V, C)$ consiste em dois conjuntos finitos, um conjunto V de pontos, chamados de nós (ou vértices), que identificam os elementos desse sistema e um conjunto C de linhas de conexões, chamadas de arestas (ou conexões ou ligações) que identificam as relações entre os elementos desse sistema, de tal modo que cada aresta conecta dois vértices, chamados de extremidades da aresta.

Um grafo pode ser representado através da matriz de adjacência $\mathbf{A}$. Os coeficientes binários a_{ij} dessa matriz $N \times N$ indicam se há ou não aresta partindo do nó i e chegando ao nó j , de modo que $a_{ij} = 1$ se e somente se os dois vértices i e j são adjacentes (ou vizinhos) em G , caso contrário $a_{ij} = 0$. Aqui, por definição, nenhum vértice é considerado adjacente a si mesmo, $a_{ii} = 0$.

Os nós i e j são adjacentes se a aresta $(i, j) \in C$. Num grafo não-direcionado, a presença da aresta que parte de i e chega j implica a existência da aresta que parte de j e chega a i . Portanto, o par (i, j) é não-ordenado. Num grafo direcionado, essa implicação não existe, ou seja, o par (i, j) é ordenado. Num grafo não-direcionado com N nós, o número máximo de pares não-ordenados, i.e., o número máximo de arestas que podem existir, é

$$\binom{N}{2} = \frac{N!}{(N-2)!2!} = \frac{N(N-1)}{2}.$$

3.3 Topologias de Rede

3.3.1 Distribuição de Grau

Num mundo real, nem todos os nós da rede tem o mesmo número de conexões. O grau (ou conectividade) k_i de um nó i é o número de arestas incidentes com o nó, e é definido em termos da matriz de adjacência $\mathbf{A}$ como:

$$k_i = \sum_{j \in C} a_{ij}. \quad (4)$$

Portanto, é útil estudar a distribuição de grau $P(k)$ da rede. Esta função dá a probabilidade de um dado nó, escolhido aleatoriamente na rede, ter um grau k fixo de conexões. O grau médio $\langle k \rangle$ pode ser assim obtido de $P(k)$:

$$\langle k \rangle = \sum_{k=k_{min}}^{k_{max}} kP(k), \quad (5)$$

sendo k_{min} o mínimo valor de k encontrado na rede e k_{max} , o máximo valor.

3.3.2 Coeficiente de Agrupamento

Definimos o coeficiente de agrupamento do i -ésimo nó da rede, C_i , como sendo a razão entre o número de conexões E_i existentes entre os k_i adjacentes a este nó e o número máximo de arestas possíveis, $k_i(k_i - 1)/2$:

$$C_i = \frac{2E_i}{k_i(k_i - 1)}. \quad (6)$$

De acordo com esta definição, o i -ésimo elemento terá coeficiente de agrupamento $C_i = 0$ se nenhum de seus adjacentes estiver conectado entre si e $C_i = 1$ se todos os seus adjacentes estiverem interconectados.

Finalmente, para medir o agrupamento de uma rede por inteiro definimos o coeficiente de agrupamento da rede, $\langle C \rangle$, como sendo a média dos C_i para todos os nós da rede:

$$\langle C \rangle = \frac{1}{N} \sum_{i=1}^N C_i. \quad (7)$$

3.3.3 Caminho Mínimo Médio

O caminho mínimo entre dois nós i e j (não necessariamente único) é definido como o número de arestas percorridas do nó i para o nó j pelo caminho mais curto. É útil representar todos os menores caminhos de um grafo G como uma matriz D , em que seus elementos d_{ij} representam o comprimento do nó i para o nó j , quando dois vértices não estão conectados, ou seja, pertencem a componentes distintos, eles são ditos desconectados e são representados por $d_{ij} = \infty$. O maior valor de d_{ij} é chamado de diâmetro do grafo.

Outra definição do tamanho linear da rede é o caminho mínimo médio, e é definido como o valor médio de d_{ij} para todos os possíveis pares de vértices na rede:

$$\langle l \rangle = \frac{1}{N(N-1)} \sum_{i,j \in C, i \neq j} d_{ij}. \quad (8)$$

3.4 Tipos de Rede

3.4.1 Rede Regular

A rede regular é o tipo mais simples e é considerada uma rede reticulada em uma dimensão com condições periódicas de contorno, isso é uma rede em anel com N nós. Estes nós estão conectados com seus d vizinhos mais próximos ($d/2$ de cada lado). Portanto, todos os nós possuem o mesmo grau de conectividade, além de coeficientes de agrupamento e caminho mínimo médio bastante grandes comparados com os outros tipos de rede.

3.4.2 Rede Aleatória

No primeiro modelo de Erdős & Rényi (1959), define-se um grafo aleatório com N nós conectados por M arestas escolhidas aleatoriamente dentre as $N(N-1)/2$ possíveis. Nesse modelo, N e M são parâmetros fixos e é representado por $G_{N,M}$.

No modelo de Erdős & Rényi (1960) há N nós e cada par é conectado com probabilidade p , ou seja, cada uma das $N(N-1)/2$ arestas existe, independentemente das demais, com probabilidade p . Existem $C_{N(N-1)/2}^M$ grafos possíveis que formam um espaço de probabilidade em que cada grafo é igualmente provável, consequentemente, o número total de conexões é uma variável aleatória proporcional a probabilidade p e com valor esperado

$$E(M) = p \frac{N(N-1)}{2}, \quad (9)$$

e é representado com $G_{N,p}$ (Albert & Barabási, 2002).

Se G é um grafo com N nós e M arestas, a probabilidade de obtê-lo por esta construção é dado por

$$P(G) = p^M (1-p)^{N(N-1)/2-M}. \quad (10)$$

O grafo aleatório $G_{N,p}$ tem uma distribuição de graus binomial. A probabilidade p_k de que um nó escolhido aleatoriamente está ligado a outros k nó é exatamente:

$$P(k) = \binom{N-1}{k} p^k (1-p)^{N-1-k}. \quad (11)$$

Para N grande e $\langle k \rangle$ fixado, a distribuição dos graus é aproximada por uma distribuição de Poisson:

$$P(k) = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!}. \quad (12)$$

Os modelos de Erdős & Rényi (1959, 1960), apesar de serem robustos para muitas redes, não descrevem aquelas redes que apresentam crescimento contínuo, as que seguem distribuição dos graus em lei de potência e as redes reais.

3.4.3 Rede Mundo Pequeno

As redes de mundo pequeno possuem características tanto de redes aleatórias quanto de regulares. A rede de mundo pequeno é caracterizada por alto

coeficiente de agrupamento e caminho mínimo curto. A rede foi proposta por Watts & Strogatz (1998) com o objetivo de reproduzir uma característica bastante comum nas redes sociais, a de que nossos amigos também são amigos entre si. O algoritmo criado por eles possui somente um parâmetro que consiste em interpolar uma rede regular em anel e uma rede aleatório. O algoritmo utilizado na construção desse modelo é o seguinte:

1. Começa-se com uma rede regular em anel com N nós, em que cada nó está conectado com os seus d vizinhos mais próximos ($d/2$ de cada lado). Para ter-se uma rede regular de agrupamento considerável, mas ainda extensa, toma-se $N \gg d \gg \ln(N) \gg 1$.
2. Realoca-se cada uma das arestas já existentes com probabilidade p . Se realocada, uma dada conexão passa a ligar um elemento da rede com qualquer outro, independente da distância geográfica.

Seguindo estas regras, uma rede gerada com $p = 0$ seria uma rede regular, e uma com $p = 1$ seria uma rede aleatória. O processo gerador está indicado na Figura 7.

Tanto o coeficiente de agrupamento quanto o caminho mínimo médio das redes regulares são grandes em relação as outras redes. Enquanto, as redes aleatórias apresentam coeficiente de agrupamento baixo e caminho mínimo médio baixo. Entre estas duas redes encontramos a topologia de mundo pequeno que consiste em valores baixos para o caminho mínimo médio e alto coeficiente de agregação. Nesta faixa, um número pequeno de conexões de longo alcance entre os nós, produzidos por $pN(d/2)$ arestas, garantem um encurtamento das distâncias na rede, enquanto as várias conexões de curto alcance garantem sua alta agregação.

3.4.4 Rede Livre de Escala

Diferentemente dos outros modelos que possuíam distribuição de grau que segue a distribuição de Poisson, as redes livres de escala possuem poucos nós mais

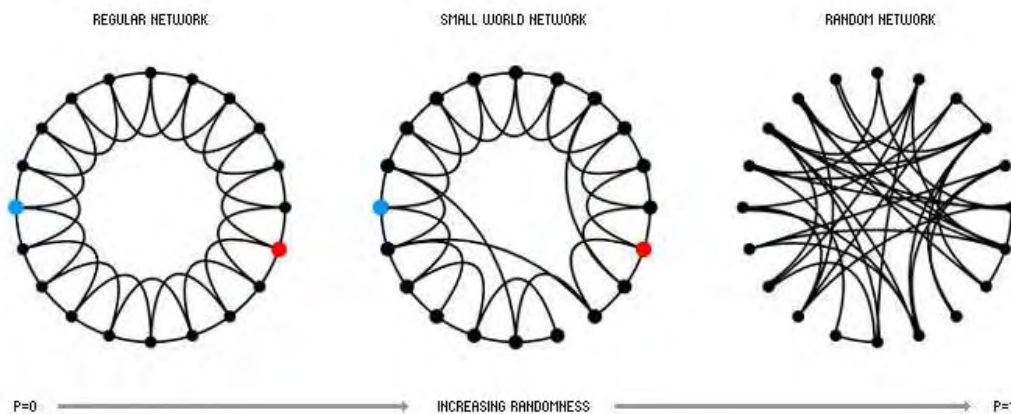


Figura 7: Diferentes rede geradas pelo algoritmo de Watts & Strogatz (1998). Da esquerda para a direita, temos uma rede regular ($p = 0$), uma rede de mundo pequeno ($p = 0,1$) e uma rede aleatória ($p = 1$). Fonte: <http://escoladeredes.ning.com/profiles/blogs/redes-complexas-da-internet-as>

conectados, enquanto a maioria dos demais possuem baixo índice de conectividade. Este tipo de rede é independente do número N de nós. Sua principal característica, que a diferencia das redes anteriores, é a probabilidade de conexão que é dada por (Barabási & Albert, 1999)

$$P(k) \sim k^{-\gamma}, \quad (13)$$

em que k é o coeficiente de conectividade ou número de conexões e o expoente γ varia, aproximadamente, entre 1 e 3 para a maioria das redes reais (?).

Barabási & Albert (1999) propuseram um algoritmo para criar uma rede livre de escala reproduzindo a distribuição de grau tão característica. Para isso, dois ingredientes são fundamentais para a construção dessas redes: *crescimento*, pois as redes reais apresentam mecanismo de adicionar nós à rede, e *conexão preferencial*, os nós adicionados na rede tendem a associar-se àqueles nós que apresentam grande número de conexões. O algoritmo é bastante simples:

- **Crescimento:** É criada uma rede com m_0 de nós, conectados entre si. Então,

adicionamos novos nós individualmente à rede e fará um número m de ligações com menor ou igual m_0 nós.

- **Conexão Preferencial:** Ao escolher as m conexões de cada nó novo da rede, vamos assumir que a probabilidade Π de que um novo nó se conecte com o i -ésimo nó já existente da rede depende do grau k_i deste nó, de forma tal que:

$$\Pi(k_i) = \frac{k_i}{\sum_{j=1}^N k_j}, \quad (14)$$

em que $\Pi(k_i)$ e k_i são as probabilidades e o grau de conectividade do i -ésimo nó, respectivamente, e N é o número de nós a qualquer instante da evolução da rede.

Desta forma, um nó sendo adicionado tem uma probabilidade grande de se conectar com um nó que tenha k_i muito grande, esse nó é chamados de *hubs*. Observa-se que as redes livre de escala são mais estáveis do que as outras redes, mas os nós *hubs* não podem ser afetados, pois se isto ocorrer, existe a possibilidade de formar aglomerados de nós e consequente isolamento desses. Para as outras propriedades de redes geradas desta forma, como coeficiente de agrupamento e caminho mínimo médio, não há previsões analíticas.

4 RESULTADOS E DISCUSSÃO

Inicialmente, desenvolvemos os códigos em linguagem de programação C, os quais geraram redes com diferentes topologias. Posteriormente, estas redes foram caracterizadas utilizando algumas medidas como coeficiente de agrupamento $\langle C \rangle$, caminho médio $\langle l \rangle$, grau médio $\langle k \rangle$ e distribuição dos graus $P(k)$. Estudamos, então, como uma determinada informação propaga nos diferentes tipos de rede.

4.1 Redes Complexas

A rede regular em anel (Figura 8) gerada com $N = 100$ nós e vizinhança igual a 10 (5 vizinhos de cada lado) possui as seguintes características em relação a sua topologia: $\langle k \rangle = 10$, $\langle l \rangle = 5,454$ e $\langle C \rangle = 0,667$.

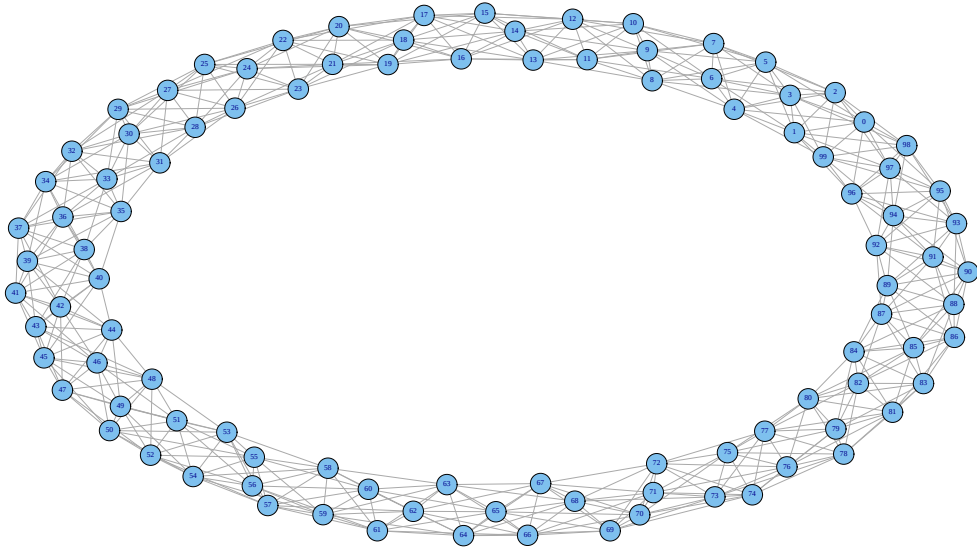


Figura 8: Rede regular com $N = 100$ nós e $d = 10$ vizinhos.

As redes aleatórias geradas pelo algoritmo de Barabási & Albert (1999) construídas com $N = 100$ nós e $M = 500$ arestas apresentaram as seguintes propriedades topológicas: $\langle k \rangle = 10$, $\langle l \rangle = 2,230 \pm 0,006$ e $\langle C \rangle = 0,101 \pm 0,007$. Enquanto, nas redes aleatórias geradas pelo algoritmo de Albert & Barabási (2002) construídas com o mesmo número de nós e $p = 0,101$, probabilidade de conexão entre dois nós quaisquer, mostraram as seguintes características: $\langle k \rangle = 9,972 \pm 0,431$, $\langle l \rangle = 2,232 \pm 0,037$ e $\langle C \rangle = 0,101 \pm 0,010$ (Figura 9). Sabemos que uma rede gerada a partir do número de arestas é equivalente a uma rede gerada com p quando $M = pN(N-1)/2$ (Albert & Barabási, 2002), o que é verdade para as redes geradas anteriormente, de maneira que as mesmas são equivalentes.

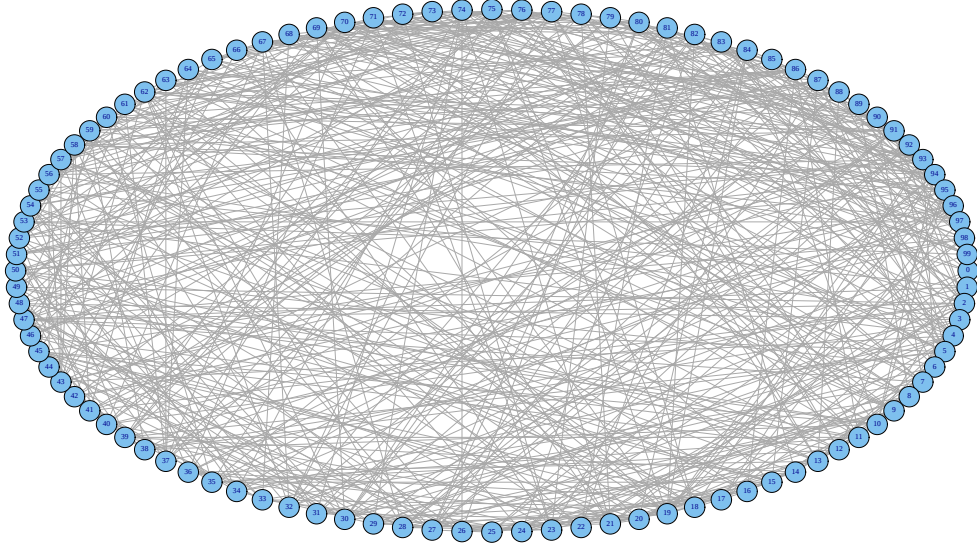


Figura 9: Rede aleatória com N nós onde cada par é conectado com probabilidade igual a 0,101.

Os valores de $\langle k \rangle$, para estas redes, foram obtidos também pelo ajuste de uma distribuição de Poisson utilizando o programa R (R Development Core Team, 2011). Para a rede de ? obtivemos $\langle k \rangle = 10,019$, enquanto para a rede de Barabási & Albert (1999) $\langle k \rangle = 10,049$ (Figuras 11 e 10).

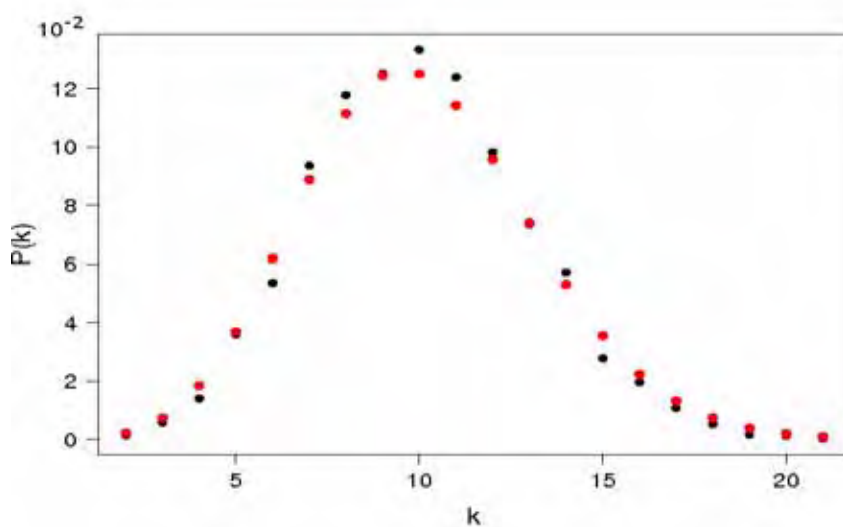


Figura 10: Distribuição dos graus das redes aleatórias geradas pelo algoritmo de Barabási & Albert (1999). Em preto os dados obtidos a partir das simulações e em vermelho os valores ajustados.

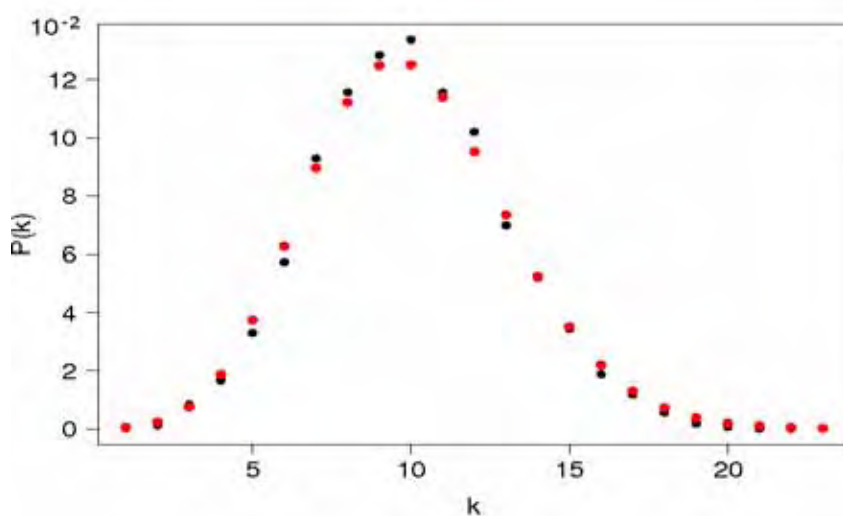


Figura 11: Distribuição dos graus das redes aleatórias geradas pelo algoritmo de Albert & Barabási (2002) com $p = 0,101$. Em preto os dados obtidos a partir das simulações e em vermelho os valores ajustados.

Na rede de mundo pequeno gerada com $p = 0,1$ obtivemos para o ajuste da distribuição dos graus uma Poisson com parâmetro igual a 10,2 (Figuras 12 e 13), para o caminho mínimo médio, $\langle l \rangle = 2,660$ e para o coeficiente de agrupamento, $\langle C \rangle = 0,485$. Os valores de caminho mínimo médio e coeficiente de agrupamento obtidos para outros valores de p são mostrados na Tabela 2.

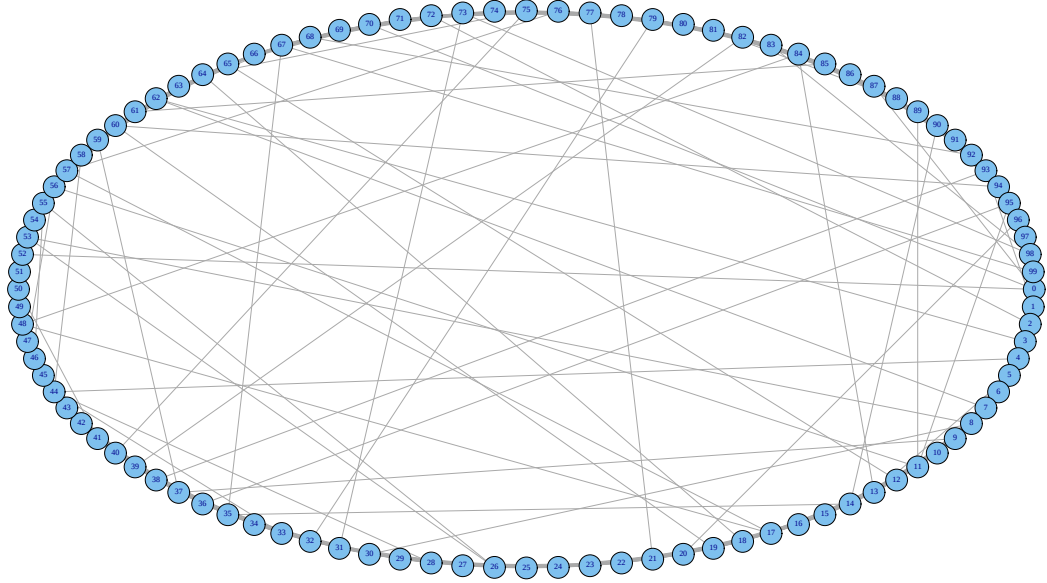


Figura 12: Rede de mundo pequeno gerada com $p = 0,1$ (algoritmo de Watts & Strogatz (1998)).

A Figura 14 mostra o comportamento da medida de caminho mínimo médio, C_a , e coeficiente de agrupamento, l_a , obtidos para redes de diferentes topologias. Neste caso, partimos de uma rede regular com $N = 100$ e $d = 10$ vizinhos, e com a probabilidade p , mudamos uma conexão na rede (variemos p de 0,0001 até $p = 1,0$ utilizando o algoritmo de Watts-Strogatz). Para $p = 0$ temos uma rede regular, $p = 1$ rede aleatória e $0,001 \leq p \leq 0,01$ uma rede de mundo pequeno. Os resultados mostrados são os valores médios obtidos em 10 simulações, normalizados com os respectivos valores obtidos para a rede regular.

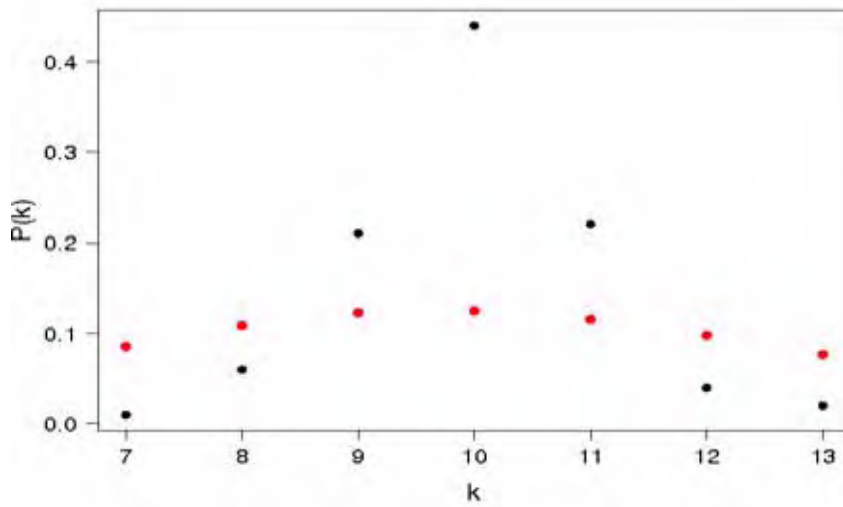


Figura 13: Distribuição dos graus dos nós da rede mundo pequeno para $p = 0,1$. Em preto os dados obtidos a partir das simulações e em vermelho os valores ajustados.

Tabela 2: Tabela com os valores do caminho mínimo médio $\langle l \rangle$ e do coeficiente agrupamentos médio $\langle C \rangle$ para as redes mundo pequeno geradas com diferentes valores de p .

p	$\langle l \rangle$	$\langle C \rangle$
0,0001	5,412	0,666
0,001	5,002	0,662
0,005	4,024	0,647
0,01	3,660	0,635
0,05	2,860	0,548
0,07	2,758	0,520
0,1	2,660	0,485
0,2	2,612	0,465
1	2,230	0,101

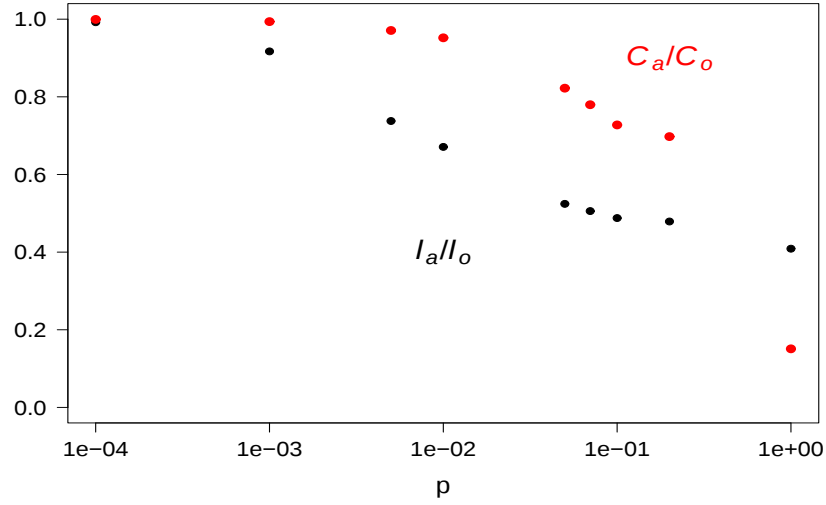


Figura 14: O comportamento do caminho mínimo médio $\langle l \rangle$ (pontos pretos) e do coeficiente de agrupamento $\langle C \rangle$ (pontos vermelhos) como funções da probabilidade de realocação de conexões p no modelo de Watts-Strogatz. As redes foram geradas com $N = 100$ e $d = 10$ (5 vizinhos de cada lado). Na figura C_o e l_o são, respectivamente, os coeficientes de agrupamento e caminho médio para a rede regular.

As redes livre de escala geradas pelo algoritmo de Barabási & Albert (1999) apresentaram $\langle k \rangle = 2,040$, $\langle l \rangle = 4,133 \pm 0,408$ e $\langle C \rangle = 0,003 \pm 0,002$ (Figura 15). O ajuste obtido pela distribuição dos graus dos nós foi de $P(k) = 0,683k^{-2,190}$ (Figura 16). Observe que o valor do expoente está entre 1 e 3 indicando uma lei de potência para esse tipo de rede.

Uma vez construída a rede, escolhemos em $t = 0$, um nó para receber uma informação. Para este nó, verificamos o caminho mínimo entre ele e seus vizinhos mais próximos e incrementamos $t = t + 1$. Este processo é repetido até que todos os nós da rede recebam essa informação como mostra a Figura 17.

Observando a distribuição da informação nos diferentes tipos de redes, verificamos que a rede livre de escala precisa de mais tempo para que todos os nós tenham recebido a mesma informação, enquanto que as redes aleatória e mundo pe-

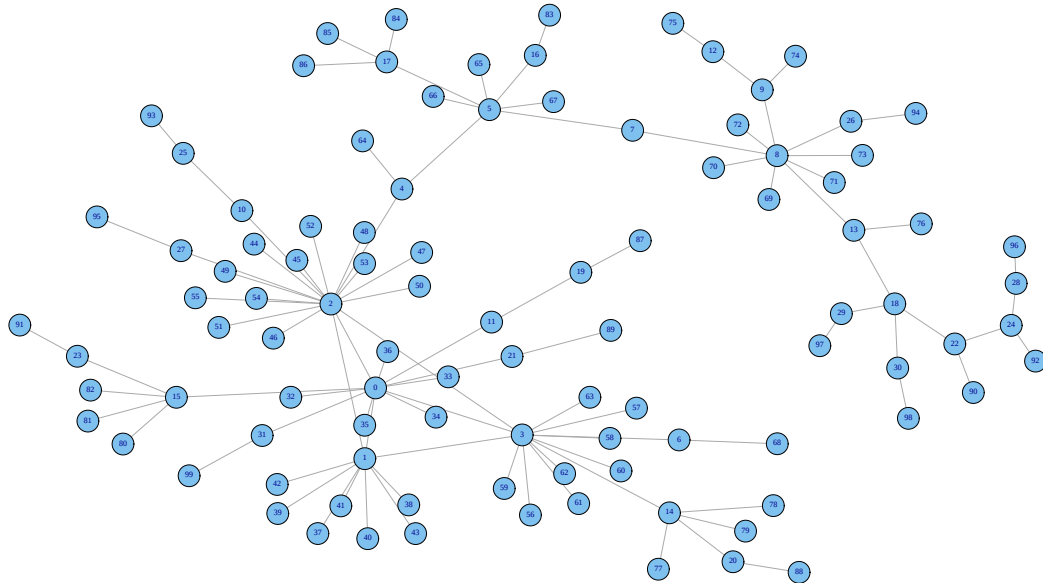


Figura 15: Rede livre de escala gerada a partir de $m_0 = 3$ nós conectados entre si. Cada novo nó adicionado na rede só realiza uma conexão com os outros nós. A rede final é composta por 100 nós.

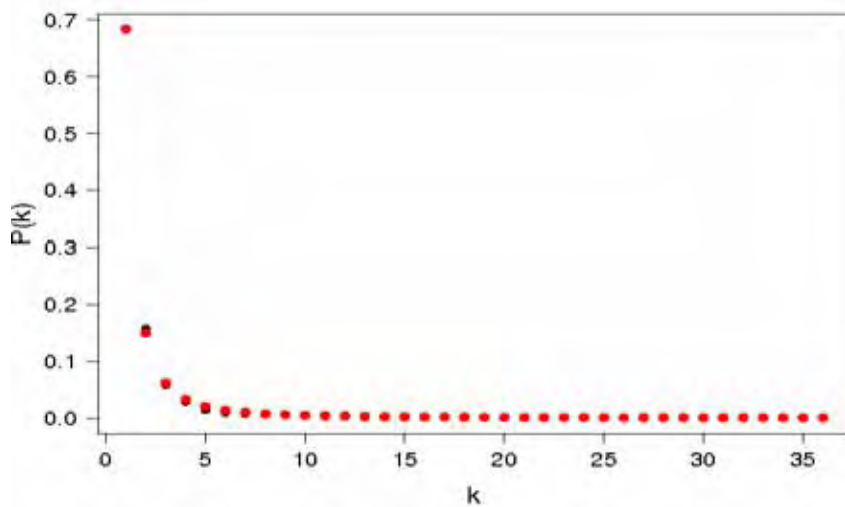


Figura 16: Distribuição dos graus dos nós da rede livre de escala gerada a partir de $m_0 = 3$ nós conectados entre si e sua rede final com 100 nós. Em preto os dados obtidos a partir das simulações e em vermelho os valores ajustados.

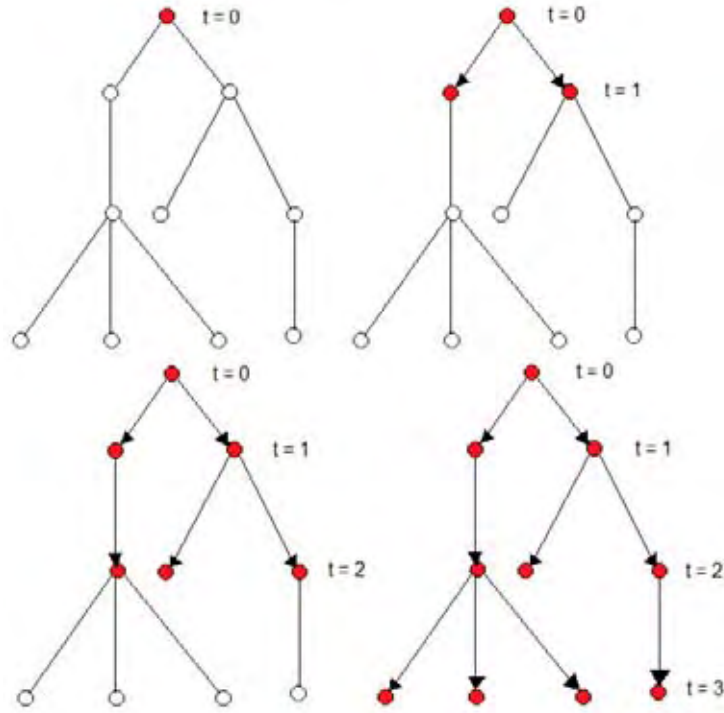


Figura 17: Desenho esquemático da distribuição da informação em uma rede com 10 nós. Em $t = 0$ somente um nó recebeu a informação (bolinha vermelha) e nos tempos seguintes essa informação foi passada para os nós vizinhos, em $t = 3$ toda rede possui a informação.

queno têm os menores tempo relativos a distribuição de informação na rede (Figuras 18, 19 e 20). A distribuição da informação na rede aleatória de Erdős & Rényi (1959) é similar a distribuição na rede aleatória de Erdős & Rényi (1960). Portanto, pode-se concluir que apesar das redes livre de escala serem mais resistentes/robustas, no sentido de invasão (Naug & Camazine, 2002; Pie et al., 2004), as redes aleatória e de mundo pequeno têm maior facilidade na comunicação entre os nós, de maneira que a informação flui mais rápida neste tipo de rede. Os resultados apresentados são a média de 20 simulações. Todas as redes tem $N = 100$ nós e da ordem de 500 conexões.

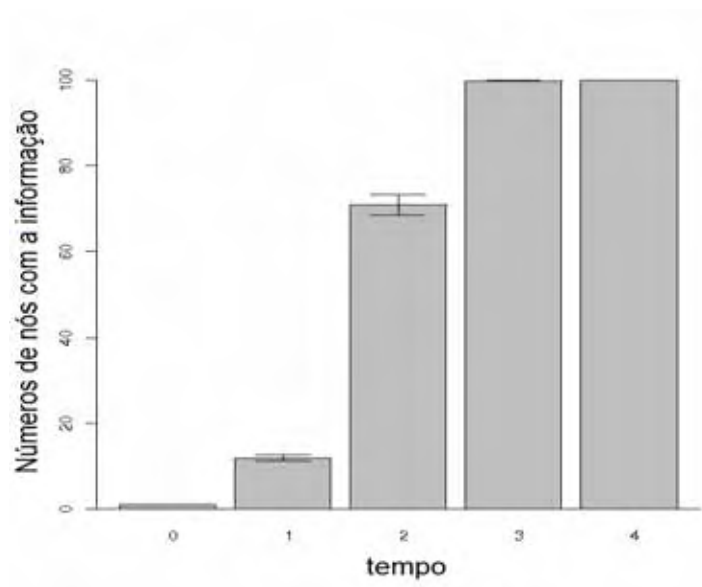


Figura 18: Distribuição da informação para rede aleatória Erdős & Rényi (1960). As barras horizontais são os erros-padrão.

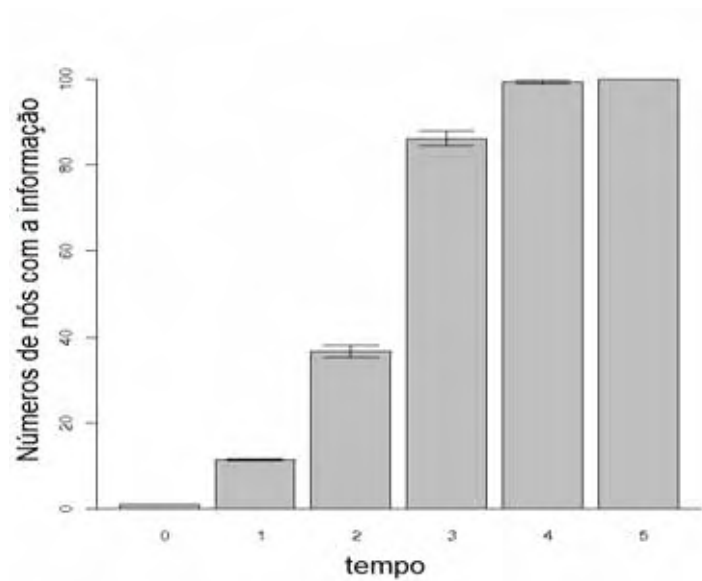


Figura 19: Distribuição da informação para a rede mundo pequeno com $p = 0, 1$. As barras horizontais são os erros-padrão.

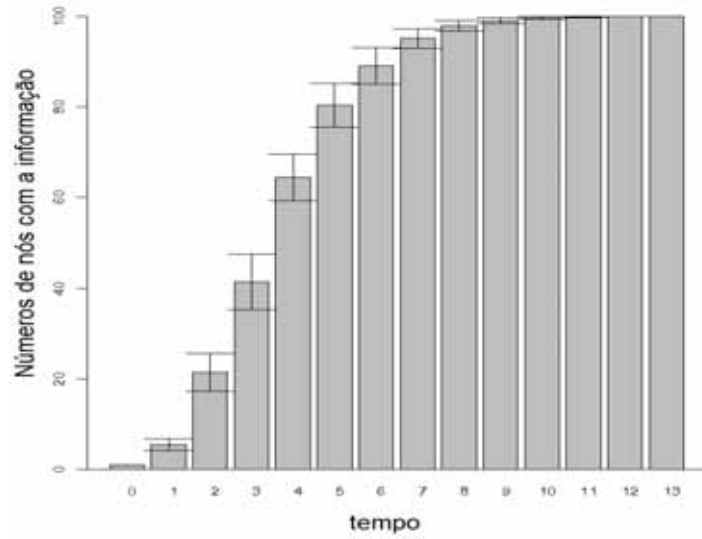


Figura 20: Distribuição da informação para a rede livre de escala com 100 nós. As barras horizontais são os erros-padrão.

4.2 Autômatos Celulares

Numa primeira abordagem o modelo proposto por Miramontes et al. (1993) foi implementado com o objetivo de caracterizar o tipo de rede de interação que surge do ato comportamental de ativação-inativação dos indivíduos da colônia. Utilizamos diferentes condições de contorno (periódica e de valores fixos) e variamos a condição inicial, dada pela densidades de agentes na rede (número de agentes dividido por N^2). Após 1.000 iterações construímos a matriz de incidência e caracterizamos as redes geradas pelo modelo de acordo com as medidas relativas a topologia da rede. Para o segundo modelo, construído com duas casta (operárias e rainhas) a evolução temporal da topologia da rede foi descrita.

4.2.1 Caracterização da rede de interação do ato comportamental de ativação-inativação em formigas

Para o modelo de autômatos descrito anteriormente (seção 1.2.1), com condição de contorno periódica, obtivemos que para densidade de agentes na rede

de 0,05 à 0,964 estas redes foram classificadas como redes aleatória de Erdős & Rényi (1960). Para densidades entre 0,976 a 0,997 classificamos estas redes como de mundo pequeno. Para densidade igual a 1,00 a rede gerada possui características mistas, sendo sua distribuição dos graus igual a de uma rede regular, e seus valores de caminho médio e de agrupamento iguais a de uma rede de mundo pequeno.

Do ponto de vista biológico, modelar um formigueiro com condições de contorno periódica, é pouco realista, de maneira que testamos a influência das condições de contorno neste problema. Utilizando então, condições de contorno fixas, obtivemos que, foram classificadas como redes aleatórias, as redes com densidades de 0,05 à 0,70, e como mundo pequeno as redes com densidades de 0,80 à 0,995. Para densidade igual a 1,00 o resultado foi muito semelhante ao obtido com condições de contorno periódica, exceto pela distribuição dos graus.

Os valores do grau médio de interação foram maiores quando utilizamos condição de contorno periódica do que com a condição de contorno de valor fixo. Isso já era esperado, pois a condições de contorno de valor fixo limita a difusão dos agentes na rede, e conseqüentemente, limita o número de contatos, diminuindo o grau médio da rede (Figura 21). Também houve um deslocamento no valor máximo obtido para o grau médio, para condições de contorno de valor fixo obtivemos $\langle k_{max} \rangle = 205,45$ na densidade de 0,5, enquanto para a condição de contorno periódica $\langle k_{max} \rangle = 438,18$ na densidade de 0,8.

Para as redes com condição periódica de contorno, nas densidades de 0,20 à 0,80, obtivemos os maiores valores para os coeficientes de agrupamento médio ($0,909 \pm 0,029$) e os menores valores para o caminho médio ($1,091 \pm 0,030$), entretanto, para as redes geradas com a condição de contorno de valor fixo, o maior valor para os coeficientes de agrupamento foi de $0,793 \pm 0,038$ e o menor valor para o caminho mínimo médio foi de $1,267 \pm 0,057$ em um intervalo de densidade, de 0,20 à 0,50 (Figura 22). Especula-se que as colônias tenham densidade entre 0,20 e 0,50 visto que neste intervalo a transmissão de informação é máxima. No trabalho realizado por Vieira et al. (2007) sobre arquitetura de ninhos da formiga *Ectatomma vizottoi*

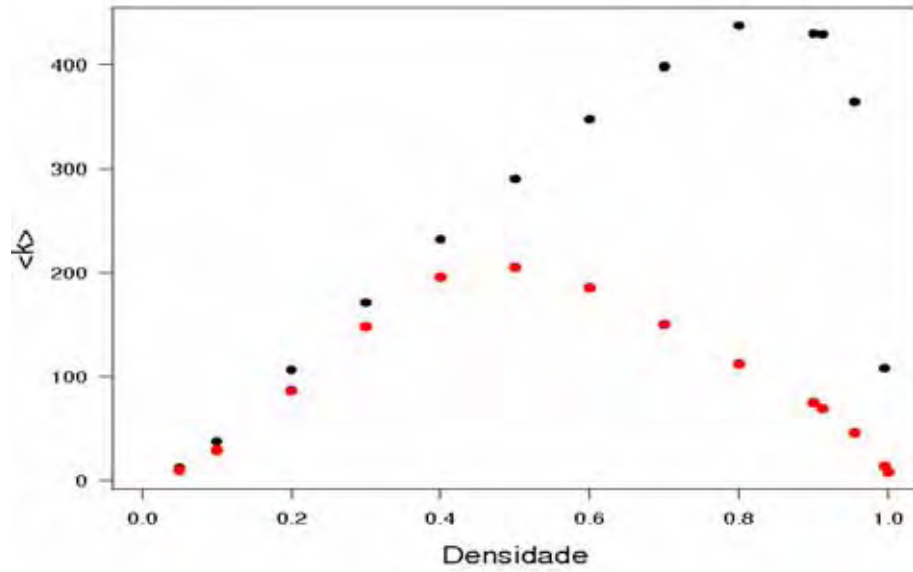


Figura 21: Grau médio em função da densidade de agentes na rede. Em vermelho para o modelo com condição de contorno valores fixos e em preto para o modelo com condição de contorno periódica.

Almeida (1987), os pesquisadores fornecem as dimensões (comprimento, largura e altura) das câmaras e o tamanho populacional dos oito ninhos. A partir desses dados calculamos o quociente entre as áreas de cada ninho com o número de operárias, o valor médio obtido foi de $0,225 \pm 0,052$ sendo que o menor valor foi de 0,148 e o maior foi de 0,278, corroborando com os resultados obtidos nesse trabalho.

Para o período de sincronização e para os valores da razão sinal-ruído (Figura 23), o modelo obteve comportamento similar para as duas condições de contorno a partir da densidade de 0,20, sendo que o período de sincronização para a condição de contorno de valor fixo foi maior do que a periódica e os valores da razão sinal-ruído foram acima de 0,70 indicando uma boa transmissão de informação na colônia. Observe que o comportamento do período de sincronização e da razão sinal-ruído em relação a densidade de agentes corroboram com os resultados obtidos na análise da topologia da rede (valor máximo de $\langle C \rangle$ e mínimo de $\langle l \rangle$) com relação

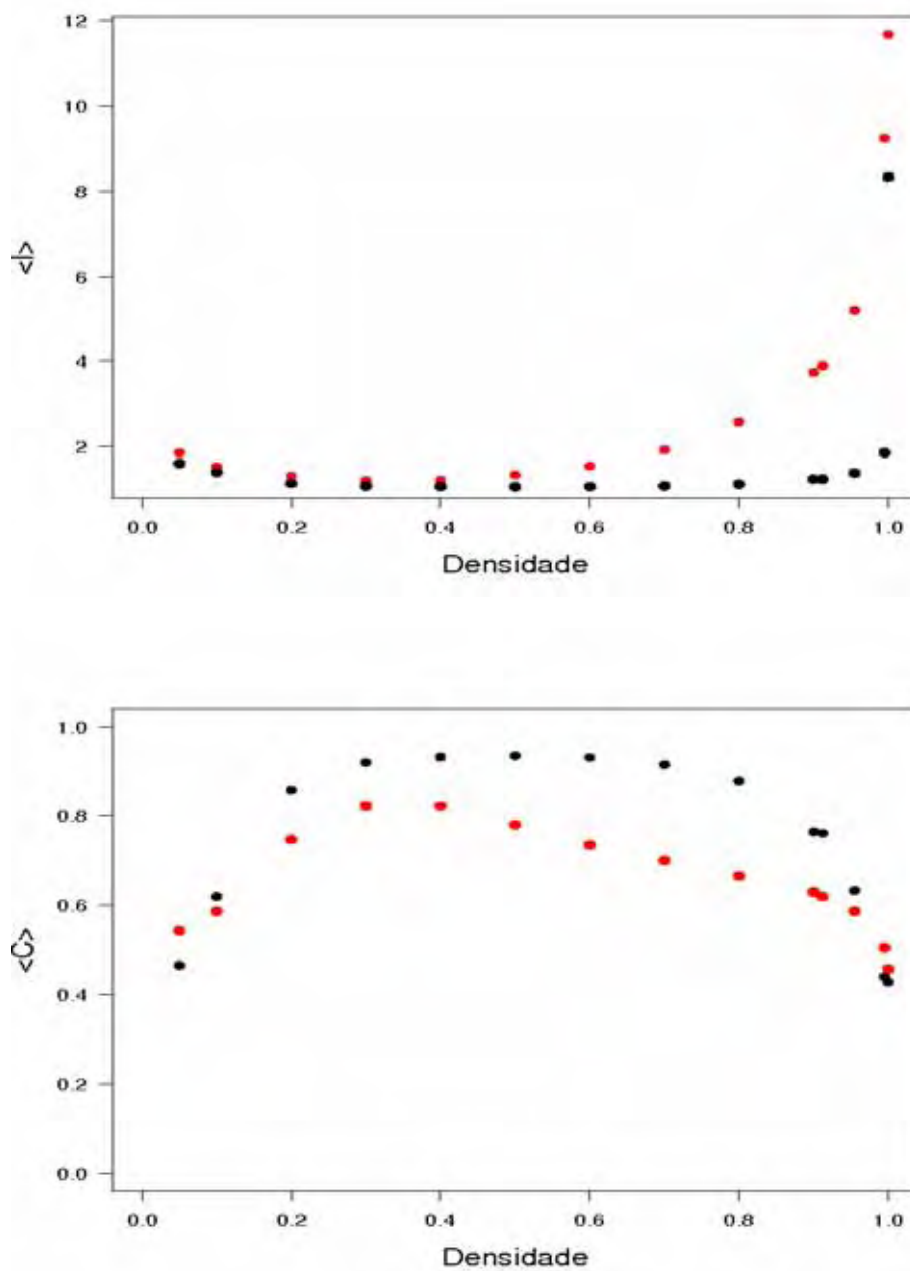


Figura 22: Diferentes valores do caminho mínimo médio $\langle l \rangle$ e do coeficiente de agrupamento $\langle C \rangle$ em função da densidade de agentes. Em vermelho para o modelo com condição de contorno de valores fixos e em preto para condição de contorno periódica.

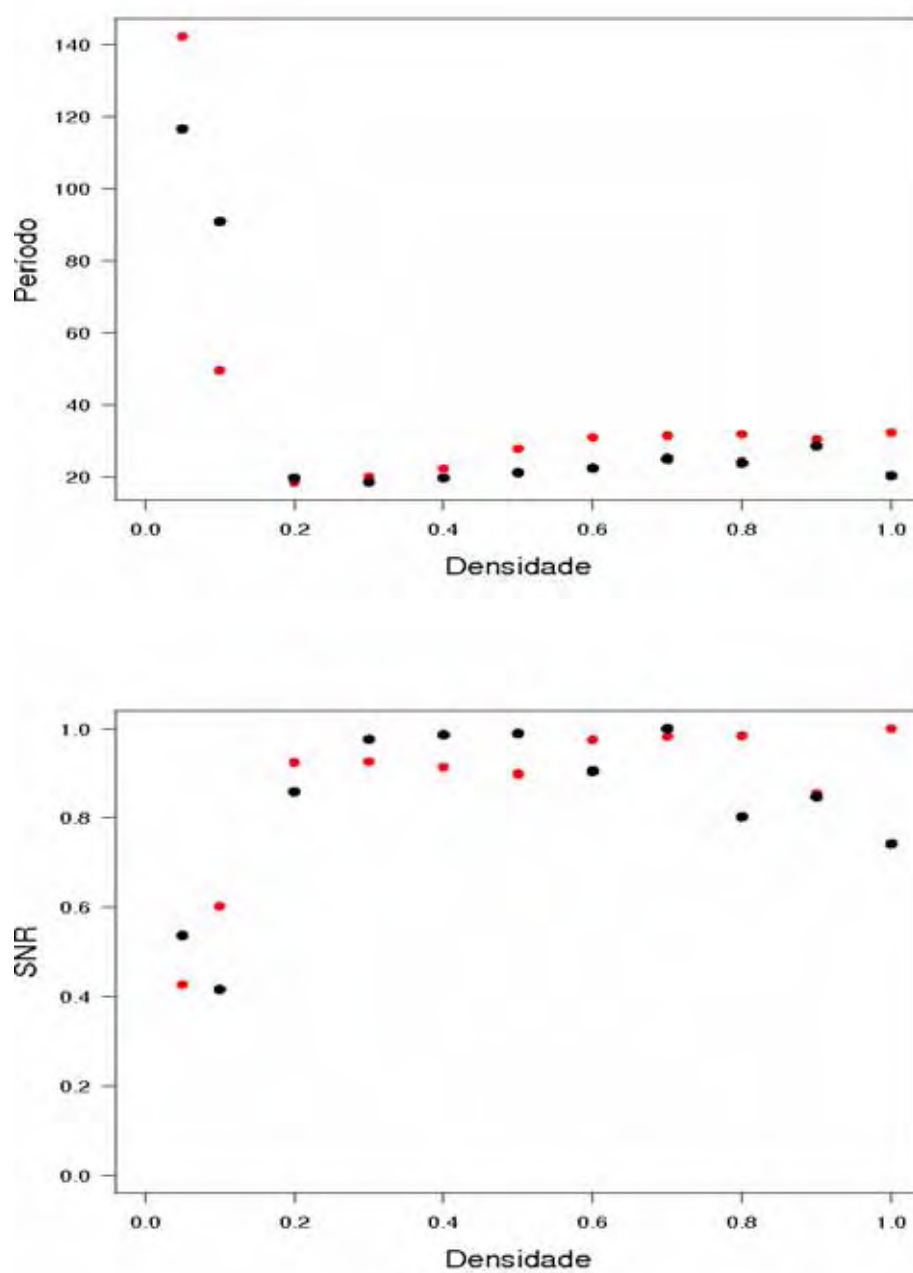


Figura 23: Relação do período de sincronização e a razão sinal-ruído (SNR) em função da densidade de agentes na rede. Em vermelho para o modelo com condição de contorno de valores fixos e em preto para condição de contorno periódica.

ao limiar para a densidade de formigas na colônia.

Em geral, as redes geradas pela condição de contorno periódica resultaram os maiores valores para o grau médio, coeficiente de agrupamento e os menores valores para o caminho mínimo médio. Isso foi devido a ausência de estruturas que limitassem o movimento dos agentes da rede, como ocorreu nas redes geradas pela condição de contorno de valores fixos.

4.2.2 Evolução temporal da rede de interação do ato comportamental de ativação-inativação para duas castas de formigas

A Figura 24 mostra o comportamento temporal do caminho mínimo médio e do grau médio obtido para a colônia e para as rainhas. Os valores de caminho mínimo médio e do grau médio obtidos para a colônia foram iguais ao da casta das operárias. Observamos que os maiores valores para o grau médio e os menores valores para o caminho mínimo médio, como esperados, foram obtidos para tempos grandes, sendo que estes valores devem estabilizar visto que assintoticamente a rede se tornará quasi-completa, pois somente não teríamos interações entre as rainhas. Os valores do parâmetro grau médio da casta das rainhas foram menores do que da colônia, enquanto os maiores valores do caminho mínimo médio foram da casta das rainhas. Os coeficientes de agrupamento médio para as operárias e para a colônia foram aproximadamente $0,356 \pm 0,062$ nos diferentes tempos. Lembramos que não existe coeficiente de agrupamento para a casta das rainhas, pois não existe interação entre elas para o ato comportamental limpeza mútua.

As redes geradas pelas operárias e da colônia nos diferentes tempos foram classificadas como rede aleatória, enquanto que as redes geradas pelas interações das rainhas foram inicialmente classificadas como redes livre de escala, depois sua topologia foi modificada (Figura 25). Esses resultados são facilmente verificados a partir das Figuras 26 e 27 e das Tabelas 3 e 4.

Para a colônia, temos que a distribuição dos graus segue uma distribuição de Poisson com parâmetro igual a 6,041 no tempo 1000 e no tempo 5800

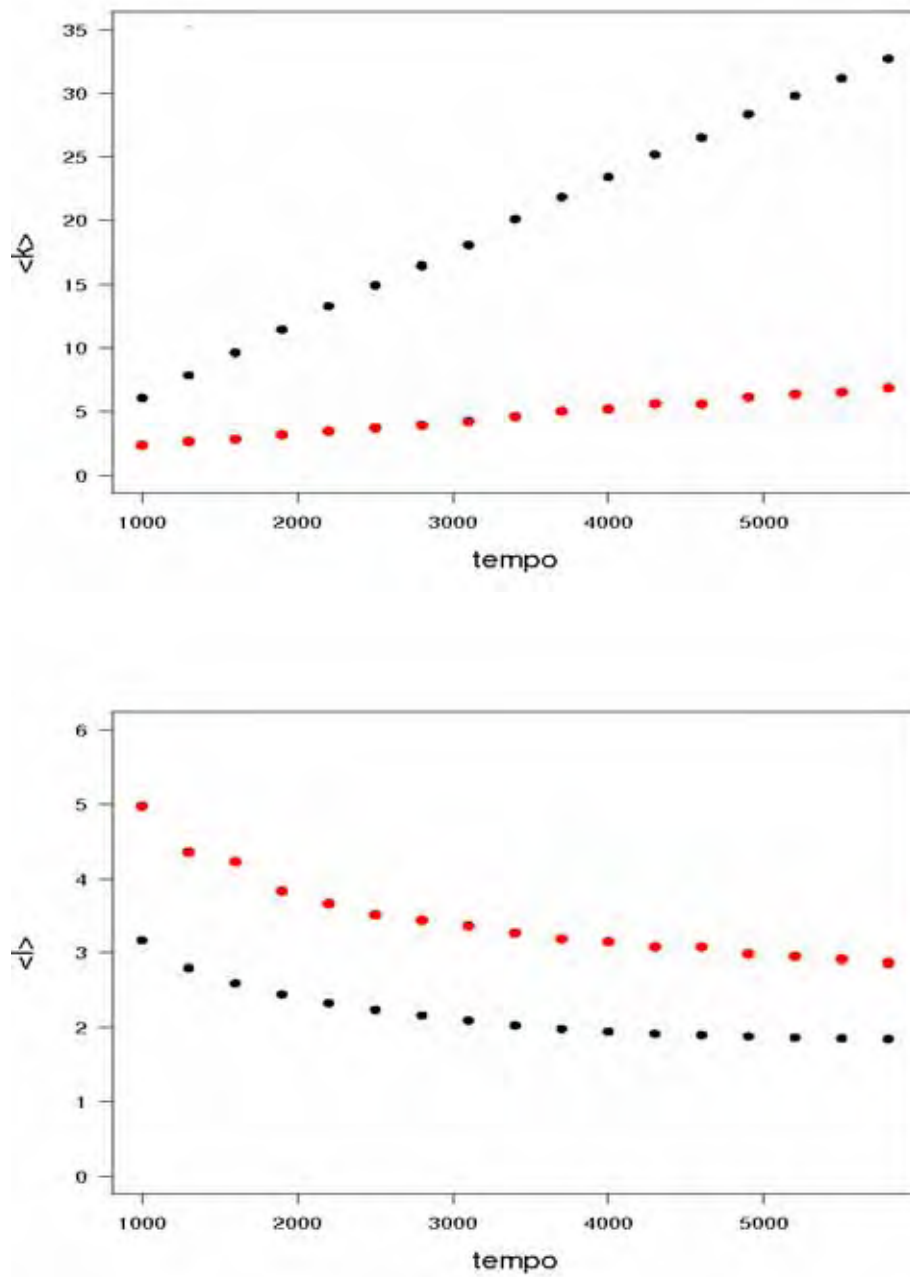


Figura 24: Evolução temporal dos valores do grau médio $\langle k \rangle$ e do caminho mínimo médio $\langle l \rangle$ para a colônia e rainhas. Em preto para colônia e em vermelho somente as rainhas.

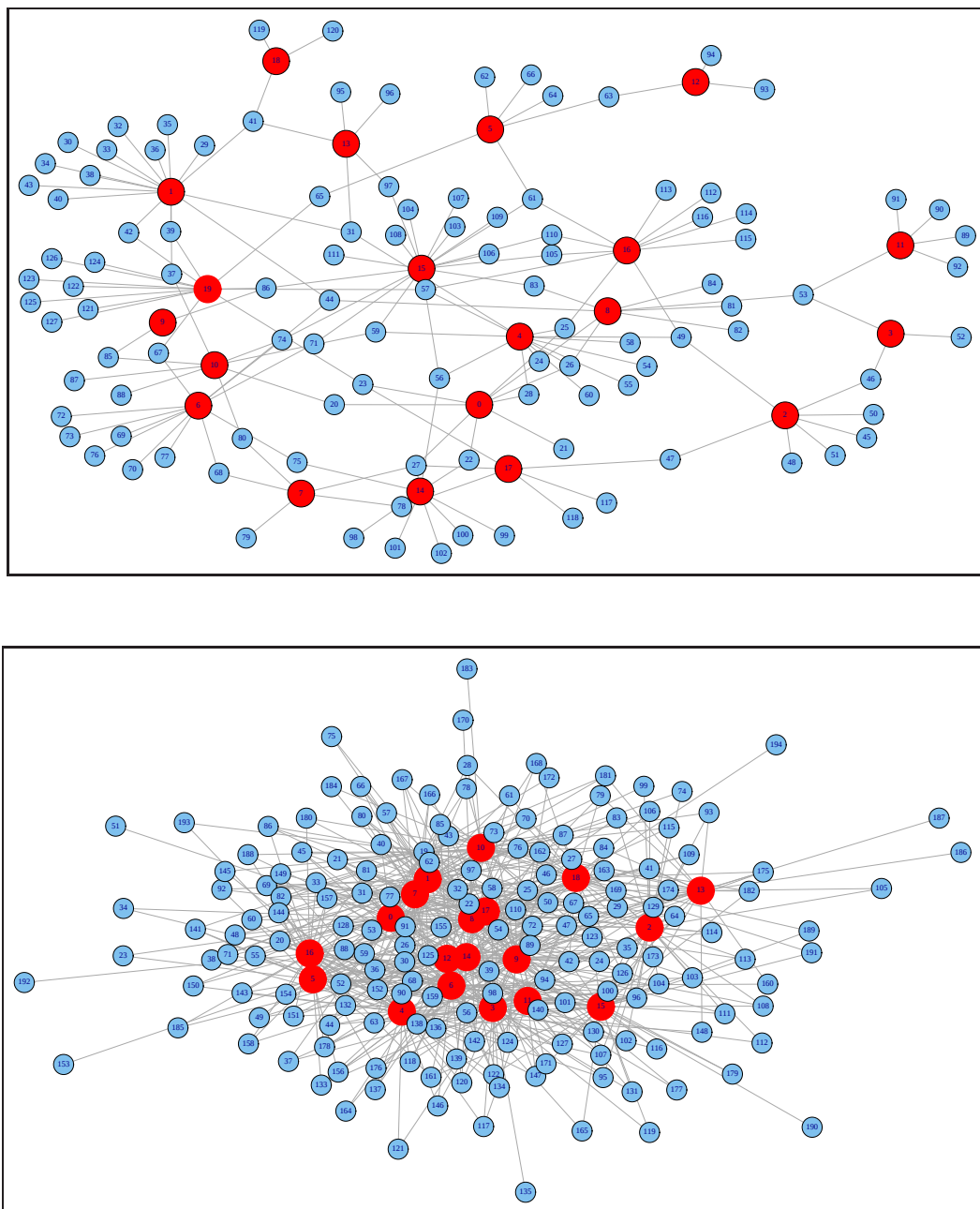


Figura 25: Diferentes topologias na casta das rainhas. Acima temos uma rede livre de escala para $t = 1000$ e abaixo topologia modificada para $t = 5800$. Em azul temos as operárias e em vermelho as rainhas.

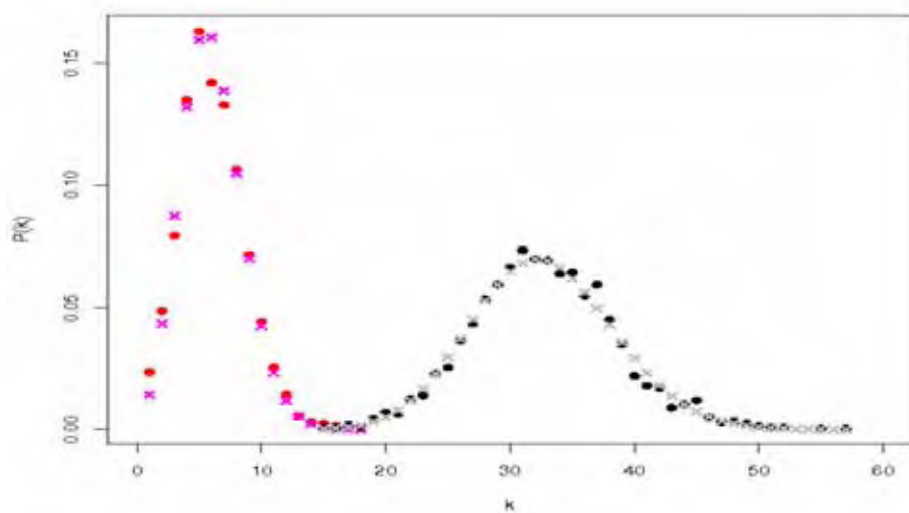


Figura 26: Distribuições dos graus de interação da colônia nos tempos 1000 e 5800. Os pontos vermelhos são os valores obtidos a partir das simulações no tempo 1000 e em preto no tempo 5800, e os x's correspondem aos valores ajustados.

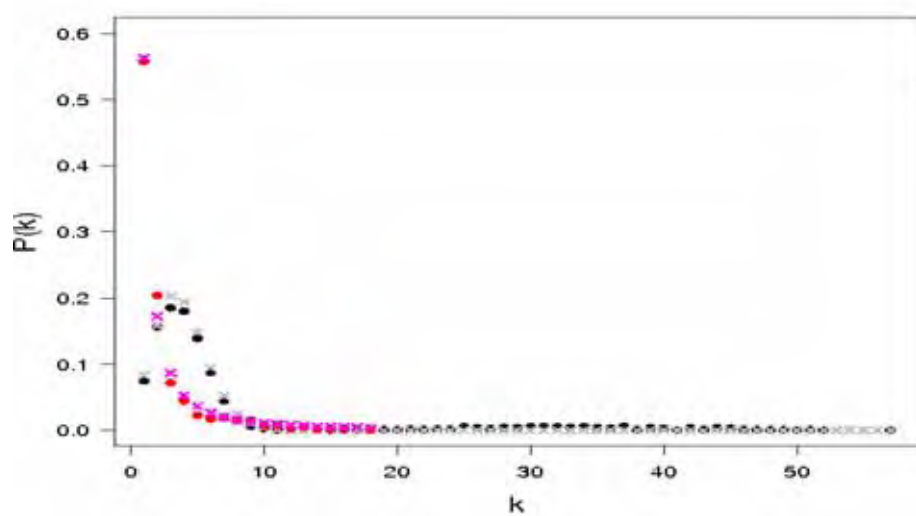


Figura 27: Distribuições dos graus de interação para a casta das rainhas nos tempos 1000 e 5800. Os pontos vermelhos são os valores obtidos a partir das simulações no tempo 1000 e em preto no tempo 5800, e os x's correspondem aos valores ajustados.

segue uma distribuição Normal com média igual a 32,436 e variância igual a 30,736 (Tabela 3). Entretanto, para as rainhas a distribuição dos graus no tempo 1000 tem uma distribuição de lei de potência com expoente igual a 1,701, mas no tempo 5800 a distribuição dos graus é melhor ajustada por uma distribuição Normal com média igual a 3,472 e variância igual a 4,328 (Tabela 4). Albert & Barabási (2002) mostraram que no modelo de redes livre de escala sem crescimento a distribuição dos graus muda com a evolução do tempo, nesse caso da distribuição de lei de potência para uma normal.

Tabela 3: Valores dos parâmetros ajustados para os dados de distribuição dos graus de interação da colônia para o tempo 1000 e 5800 utilizando as distribuições de probabilidade: lei de potência, Normal, Poisson e Exponencial discreta. Os parâmetros com * indica que $p < 0,05$ e os termos entre parênteses são os erros-padrão.

Modelos	Parâmetros	Tempo 1000		Tempo 5800	
		Estimativa	AICc	Estimativa	AICc
Lei de Potência	α	-0,268 ($\pm 0,243$)	-47,114	-0,040 ($\pm 0,440$)	-174,321
	β	0,095 ($\pm 0,043$)*		0,029 ($\pm 0,044$)	
Normal	μ	5,759 ($\pm 0,077$)*	-122,427	32,436 ($\pm 0,096$)*	-353,088
	σ^2	6,629 ($\pm 0,327$)*		30,736 ($\pm 0,871$)*	
Poisson	λ	6,041 ($\pm 0,060$)*	-129,508	32,686 ($\pm 0,103$)*	-352,376
Expon. discreta	λ	0,109 ($\pm 0,034$)*	-53,603	0,030 ($\pm 0,010$)*	-171,732

Mudança na distribuição dos graus em relação ao tempo foi relatado por Blonder & Dornhaus (2011) em dados experimentais para uma colônia da formiga *Temnothorax rugatulus* (Emery, 1895). Nesse trabalho os dados experimentais relativos ao número de interações recebidas pelas formigas (“out-degree”) no tempo foram ajustados para distribuição dos graus e foi observada a mudança da distribuição binomial para uma distribuição de Poisson com a evolução temporal do sistema (Figura 28). Observe que a diferença entre os AIC’s ($\Delta = |AIC_P - AIC_N|$) da distribuição normal e a de Poisson para a colônia no tempo 5800 é de 0,710 e no

Tabela 4: Valores dos parâmetros ajustados para os dados de distribuição dos graus de interação da casta das rainhas para o tempo 1000 e 5800 utilizando as distribuições de probabilidade: lei de potência, Normal, Poisson e Exponencial discreta. Os parâmetros com * indica que $p < 0,05$ e os termos entre parênteses são os erros-padrão.

Modelos	Parâmetros	Tempo 1000		Tempo 5800	
		Estimativa	AICc	Estimativa	AICc
Lei de potência	α	-1,701 ($\pm 0,055$)*	-100,704	-0,710 ($\pm 0,113$)*	-191,245
	β	0,562 ($\pm 0,010$)*		0,170 ($\pm 0,029$)*	
Normal	μ	0,998 ($\pm 0,058$)*	-75,210	3,472 ($\pm 0,038$)*	-396,275
	σ^2	0,524 ($\pm 0,042$)*		4,328 ($\pm 0,129$)*	
Poisson	λ	1,059 ($\pm 0,222$)*	-50,368	3,825 ($\pm 0,036$)*	-383,744
Expon. discreta	λ	0,846 ($\pm 0,033$)*	-89,905	0,181 ($\pm 0,021$)*	-226,579

tempo 1000 a diferença entre os AIC's é de 7,080. Segundo Burnham & Anderson (2002) quando $\Delta < 2$ indica que não existe diferença entre os modelos e quando o valor de Δ está entre 4 e 7 indica que o ajuste pela distribuição Normal é melhor do que a de Poisson, embora os dois modelos não sejam tão discrepantes.

Podemos comparar as redes das rainhas com uma rede de informação, como a que caracteriza o ambiente “**World Wide Web**”, onde a maioria dos nós está conectados a poucos nós, chamados de “hubs” (Albert & Barabási, 2002; Newman, 2003). Observamos que a estabilidade da rede de conexões deve-se aos nós “hubs”, qualquer perturbação nestes nós, por exemplo, morte da rainha, existe a possibilidade de quebra da rede em vários aglomerados e, conseqüente, o isolamento destes. No trabalho Fewell (2003) sobre redes de inseto social, os autores comentam a importância de poucos indivíduos na distribuição da informação e da importância desse tipo de topologia (rede livre de escala) na resiliência da colônia.

O período de sincronização de ativação entre as operárias foi menor do que o obtido para as rainhas, sendo que o período de sincronização da colônia está

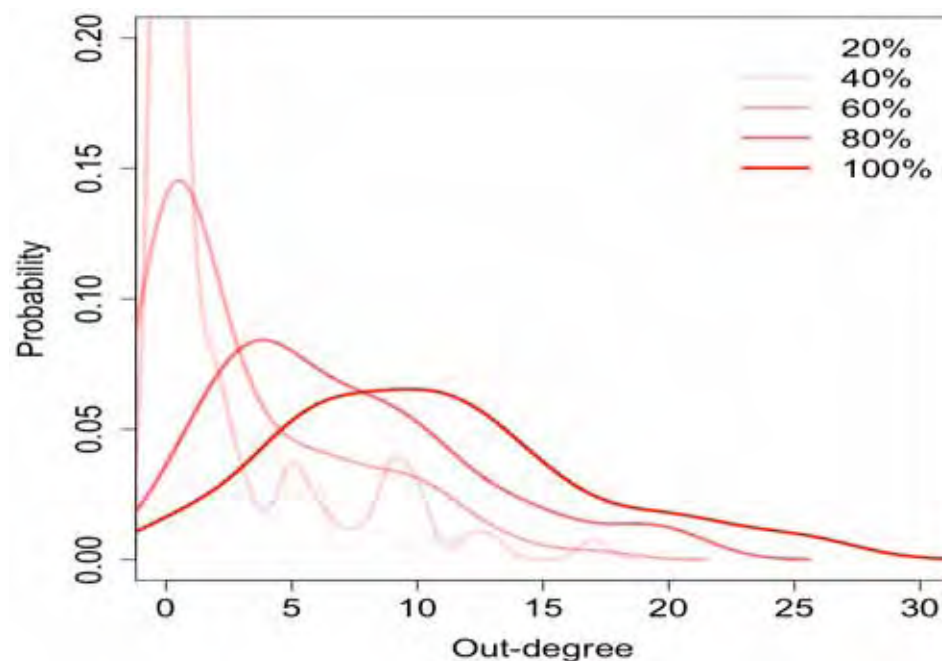
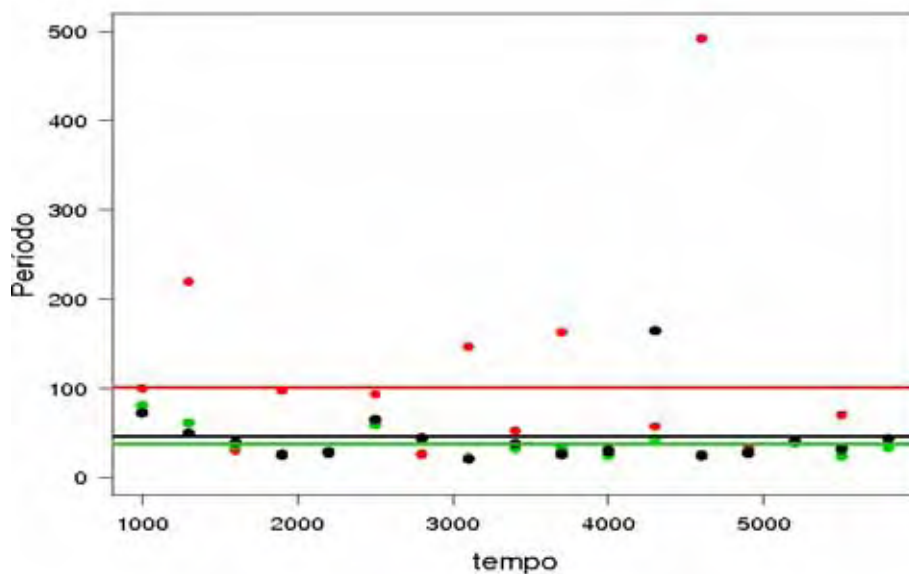


Figura 28: Distribuição dos graus com a evolução temporal da formiga *T. rugatulus*.
Fonte: Blonder & Dornhaus (2011).

mais próximo ao das operárias (Figura 29, acima), esse resultado se deve ao fato de que as operárias são mais ativas e em maior número do que as rainhas (Hölldobler & Wilson, 1990) como observado no repertório comportamental de *C. senex*, o qual apresentou cerca de 40% da colônia em estado de imobilidade, sendo que as rainhas ficaram mais tempo imóveis (47,39%) do que as operárias (39,23%) (Santos et al., 2005). Em média, a razão sinal-ruído (SNR) para as operárias ($0,434 \pm 0,074$) foi similar ao obtido pelas rainhas ($0,449 \pm 0,094$). Quando SNR é da ordem de 0,5 (Figura 29) significa que há muita variação no período de sincronização, isso acontece porque a densidade de formigas utilizada na rede é menor que 0,2 (ver Figura 23). Este resultado sugere que o experimento deve ser feito utilizando uma densidade maior para validação das conclusões obtidas.

A partir dos resultados obtidos até aqui, especula-se que os insetos sociais devem mudar a topologia das suas redes frente as perturbações, pois essa mudança funcionaria como uma estratégia evolutiva para maximizar ou minimizar



a transmissão da informação, por exemplo, na invasão na colônia por predadores a transmissão da informação tem que ser máxima, enquanto na invasão por patógenos tem que ser a mínima possível. Uma maneira que os insetos sociais têm para mudar a topologia da rede de interação entre eles é através da regulação da densidade de indivíduos na colônia. Portanto, novos experimentos devem ser elaborados com o objetivo de validar os resultados propostos. Finalmente, os resultados obtidos nesse trabalho são importantes para o entendimento da biologia dos insetos sociais, entretanto informações dos dados experimentais, principalmente em formigas, são fundamentais para o desenvolvimento e melhoria dos modelos.

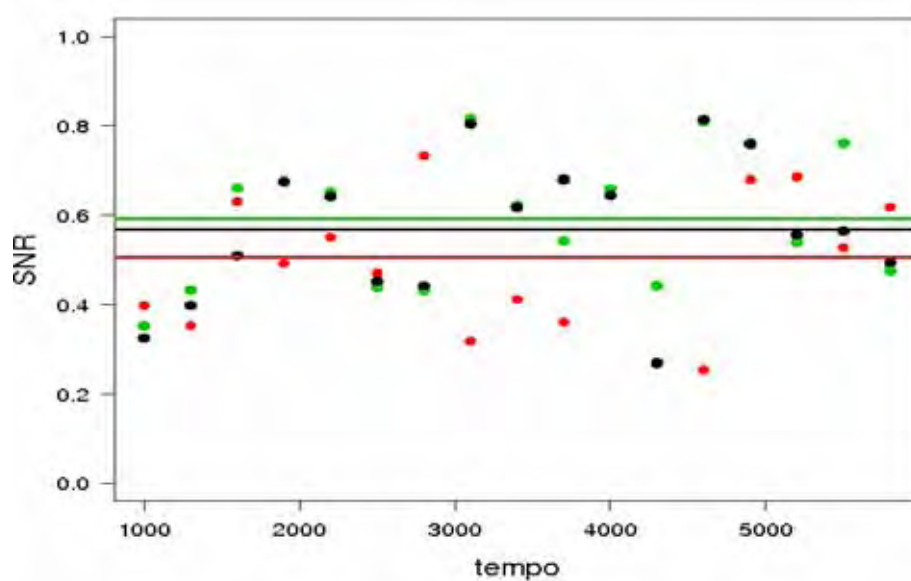


Figura 29: Relação do período de sincronização e da razão sinal-ruído como função do tempo. Os pontos pretos representam todos os indivíduos da colônia, em verde as operárias e em vermelho as rainhas. As linhas horizontais representam os valores médios do período de sincronização para cada casta e colônia.

5 CONCLUSÕES

Os resultados indicam que as colônias devem ter densidade entre 0,20 e 0,50 visto que neste intervalo a transmissão de informação na colônia é máxima. A rede de interação entre as rainhas gerada pelo ato comportamental de limpeza mútua é inicialmente livre de escala, pois a distribuição dos graus é melhor ajustada por uma lei de potência, depois ocorre mudança de topologia. Além disso, obtivemos que as operárias são mais ativas do que as rainhas e a informação distribuída pelas operárias é melhor preservada do que aquela distribuída pelas rainhas. Finalmente, sugerimos que novos experimentos sejam feitos com o objetivo de validar os resultados teóricos obtidos neste trabalho e possibilitar ajustes nos modelos propostos.

REFERÊNCIAS BIBLIOGRÁFICAS

AHMED, E. Fuzzy cellular automata models in immunobiology. **Journal of Statistical Physics**, v.85, n.1/2, p.291–294, 1996.

ALBERT, R.; BARABÁSI, A.-L. Statistical mechanics of complex networks. **Reviews of Modern Physics**, v.74, n.1, p.47–97, 2002.

ALBERT, R.; JEONG, H.; BARABÁSI, A.-L. Diameter of the world-wide-web. **Nature**, v.401, p.130–131, 1999.

BAGNOLI, F.; BEZZI, M. Species formation in simple ecosystems. **International Journal of Modern Physics C**, v.9, n.4, p.1–17, 1998.

BARABÁSI, A.-L.; ALBERT, R. Emergence of scaling in random networks. **Science**, v.286, p.509–512, 1999.

BARABÁSI, A.-L.; OLTVAI, Z. N. Network biology: understanding the cell's functional organization. **Nature Reviews Genetics**, v.5, p.101–113, 2004.

BARRAT, A.; BARTHÉLEMY, M.; VESPIGNANI, A. **Dynamical Processes on Complex Networks**. New York: Cambridge University, 2008. 347p.

BASCOMPTE, J. Networks in ecology. **Basic and Applied Ecology**, v.8, p.485–490, 2007.

BASCOMPTE, J.; JORDANO, P. Plant-animal mutualistic networks: the architecture of biodiversity. **Annual Review of Ecology, Evolution, and Systematics**, v.38, p.567–593, 2007.

BASTOLLA, U.; FORTUNA, M. A.; PASCUAL-GARCÍA, A.; FERREIRA, A.; LUQUE, B.; BASCOMPTE, J. The architecture of mutualistic networks minimizes competition and increases biodiversity. **Nature**, v.458, n.23, p.1018–1020, 2009.

BEZZI, M.; CELADA, F.; RUFFO, S.; SEIDEN, P. E. The transition between immune and disease states in a cellular automaton model of clonal immune response. **Physica A**, v.245, p.145–163, 1997.

BLONDER, B.; DORNHAUS, A. Time-ordered networks reveal limitations to information flow in ant colonies. **PLoS ONE**, v.6, n.5, p.1–8, 2011.

BOCCALETTI, S.; LATORA, V.; MORENO, Y.; CHAVEZ, M.; HWANG, D.-U. Complex networks: structure and dynamics. **Physics Reports**, v.424, p.175–308, 2006.

BONABEAU, E.; THERAULAZ, G.; DENEUBOURG, J.-L.; ARON, S.; CAMAZINE, S. Self-organization in social insects. **TREE**, v.12, n.5, p.188–193, 1997.

BURNHAM, K. P.; ANDERSON, D. R. **Model selection and multi-model inference**. 2. ed. New York: Springer-Verlag, 2002.

CANNAS, S. A.; PAEZ, S. A.; MARCO, D. E. Modeling plant spread in forest ecology using cellular automata. **Computer Physics Communications**, v.121–122, p.131–135, 1999.

COLE, B. J. Short-term activity cycles in ants: generation of periodicity by worker interaction. **The American Naturalist**, v.137, n.2, p.244–259, 1991.

COLE, B. J. Short-term activity cycles in ants: age-related changes in tempo and colony synchrony. **Behavioral Ecology and Sociobiology**, v.31, p.181–187, 1992.

DEUTSCH, A.; DORMANN, S. **Cellular automaton modeling of biological pattern formation: characterization, applications, and analysis**. Boston: Birkhäuser, 2005. 343p.

DOROGOVTSSEV, S. N.; MENDES, J. F. F. Evolution of networks. **Advances in Physics**, v.51, p.1–67, 2002.

EDELSTEIN-KESHET, L.; WATMOUGH, J.; ERMENTROUT, G. B. Trail following in ants: individual properties determine population behaviour. **Behavioral Ecology and Sociobiology**, v.36, p.119–133, 1995.

ERDÖS, P.; RÉNYI, A. On Random Graphs I. **Publicationes Mathematicae**, v.6, p.290–297, 1959.

ERDÖS, P.; RÉNYI, A. On the Evolution of Random Graphs. **Publ. Math. Inst. Hung. Acad. Sci.**, v.5, p.17–61, 1960.

ERMENTROUT, G. B.; EDELSTEIN-KESHET, L. Cellular Automata Approaches to biological modeling. **Journal Theoretical Biology**, v.160, p.97–133, 1993.

FEWELL, J. H. Social Insect Networks. **Science**, v.301, p.1867–1870, 2003.

FRANKS, N. R.; BRYANT, S.; GRIFFITHS, R. Synchronization of the behaviour within nests of the ant *Leptothorax acervorum* (Fabricius) - I. Discovering the phenomenon and its relation to the level of starvation. **Bulletin of Mathematical Biology**, v.52, n.5, p.597–612, 1990.

GULLAN, P. J.; CRANSTON, P. S. **Os insetos: um resumo de entomologia**. São Paulo: Roca, 2008. 440p.

HERZ, A. V. M. Collective phenomena in spatially extended evolutionary games. **Journal Theoretical Biology**, v.169, p.65–87, 1994.

HINDE, R. A. Interactions, relationships and social structure. **Man**, v.11, p.1–17, 1976.

HÖLLDOBLER, B.; WILSON, E. O. **The ants**. Cambridge: The Belknap Press of Harvard University Press, 1990.

- JEONG, H.; TOMBOR, B.; ALBERT, R.; OLTVAI, Z. N.; BARABÁSI, A.-L. The large-scale organization of metabolic networks. **Nature**, v.407, n.5, p.651–654, 2000.
- KRAUSE, J.; CROFT, D. P.; JAMES, R. Social network theory in the behavioural sciences potential applications. **Behavioral Ecology and Sociobiology**, v.62, p.15–27, 2007.
- LEWINSOHN, T. M.; LOYOLA, R. D.; PRADO, P. I. Matrizes, redes e ordenações: a detecção de estrutura em comunidades interativas. **Oecologia Brasiliensis**, v.10, n.1, p.90–104, 2006.
- LILJEROS, F.; EDLING, C. R.; AMARAL, L. A. N.; STANLEY, H. E.; ÅBERG, Y. The web of human sexual contacts. **Nature**, v.411, p.907–908, 2001.
- MALLET, D. G.; DE PILLIS, L. G. A cellular automata model of tumor-immune system interactions. **Journal Theoretical Biology**, v.239, p.334–350, 2006.
- MEYER-HERMANN, M. A mathematical model for the germinal center morphology and affinity maturation. **Journal Theoretical Biology**, v.216, p.273–300, 2002.
- MIRAMONTES, O.; SOLÉ, R. V.; GOODWIN, B. C. Collective behaviour of random-actived mobile cellular automata. **Physica D**, v.63, p.145–160, 1993.
- MONTOYA, J. M.; SOLÉ, R. V. Small world patterns in food webs. **Journal Theoretical Biology**, v.214, p.405–412, 2002.
- NAUG, D.; CAMAZINE, S. The role of colony organization on pathogen transmission in social insects. **Journal Theoretical Biology**, v.215, p.427–439, 2002.
- NEWMAN, M.; BARABÁSI, A.-L.; WATTS, D. J. **The Structure and Dynamics of Networks**. New Jersey: Princeton University, 2006. 582p.
- NEWMAN, M. E. J. The structure of scientific collaboration networks. **Proceedings of the National Academy of Sciences USA**, v.98, p.404–409, 2001.

- NEWMAN, M. E. J. The structure and function of complex networks. **SIAM Review**, v.45, n.2, p.167–256, 2003.
- PIE, M. R.; ROSENGAUS, R. B.; TRANIELLO, J. F. A. Nest architecture, activity pattern, worker density and the dynamics of disease transmission in social insects. **Journal of Theoretical Biology**, v.226, p.45–51, 2004.
- PROULX, S. R.; PROMISLOW, D. E. L.; PHILLIPS, P. C. Network thinking in ecology and evolution. **TRENDS in Ecology and Evolution**, v.20, n.6, p.345–353, 2005.
- R DEVELOPMENT CORE TEAM. **R: A Language and Environment for Statistical Computing**. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.
- SANTOS, J. C.; YAMAMOTO, M.; OLIVEIRA, F. R.; DEL-CLARO, K. Behavioral repertory of the weaver ant *Camponotus (Myrmobrachys) senex* (Hymenoptera: Formicidae). **Sociobiology**, v.45, p.1–11, 2005.
- SCHÖNFISCH, B. Propagation of fronts in cellular automata. **Physica D**, v.80, p.433–450, 1995.
- SOLÉ, R. V.; MIRAMONTES, O.; GOODWIN, B. C. Oscillations and chaos in ant societies. **Journal Theoretical Biology**, v.161, p.343–357, 1993.
- VIEIRA, A. S.; ANTONIALLI-JUNIOR, W. F.; FERNANDES, W. D. Modelo arquitetônico de ninhos da formiga *Ectatomma vizottoi* Almeida (Hymenoptera, Formicidae). **Revista Brasileira de Entomologia**, v.51, n.4, p.489–493, 2007.
- DE VRIES, G.; HILLEN, T.; LEWIS, M.; MÜLLER, J.; SCHÖNFISCH, B. **A Course in mathematical biology: quantitative modeling with mathematical and computational methods**. Philadelphia: SIAM, 2006.
- WATMOUGH, J.; EDELSTEIN-KESHET, L. Modelling the formation of trail networks by foraging ants. **Journal Theoretical Biology**, v.176, p.357–371, 1995.

WATTS, D. J.; STROGATZ, S. H. Collective dynamics of small-world networks. **Nature**, v.393, n.4, p.440–442, 1998.

WEY, T.; BLUMSTEIN, D. T.; SHEN, W.; JORDÁN, F. Social network analysis of animal behaviour: a promising tool for the study of sociality. **Animal Behaviour**, v.75, p.333–344, 2008.

WHITEHEAD, H. Analysing animal social structure. **Animal Behaviour**, v.53, p.1053–1067, 1997.

WILSON, E. O. **Sociobiology: The New Synthesis**. Cambridge, Massachusetts: Belknap Press, 1975.

WILSON, E. O. Causes of ecological success: the case of the ants. **Journal of Animal Ecology**, v.56, p.1–9, 1987.

WOLFRAM, S. **Theory and application of cellular automata**. Singapore: World Scientific, 1986.

YOOK, S. H.; JEONG, H.; BARABÁSI, A.-L. Modeling the Internet's large-scale topology. **Proceeding of the National Academy of Sciences USA**, v.99, p.13382–13386, 2001.

APÊNDICES

Apêndice A

```
1 #define N 625
2 #define p 0.2
3 #define A 2500 //Amaxima = N(N-1)/2 <= p*Amaxima (aleatória)
4 #define max 2 //Iterações
5 #define m 1 //número de arestas para conexão (livre de escala)
6 #define mo 4 //vértices inicial (livre de escala)
7 #define d 4 // N=100 >> d = 10 >> ln(100)=4.605 >> 1 => d/2 = 5
8 #define V 1 //Vezes que o mundo pequeno é executado
9
10 int rede[N][N], DIST[N][N], lista[N], dist[N], norm[N], T[N], k[N],
    teste[N];
11 float C[N], pk[N], randon[N], randon1[N], randon2[N];
12
13 /*****
14 Construir a Matriz de Caminho Mínimo
15 *****/
16 //Verifica se o nó x está na lista[i]
17 int verifique (int x, int n){
18     int j, i=0;
19     for (j = 0; j < n; j++) if (lista[j] == x) i++;
```

```

20     return i;
21 }
22
23 //Busca o menor caminho entre os nós s = início e t = termino
24 int busca (int s, int t){
25     int i, j, c, passo, mu, x;
26     lista[0] = s;
27     dist[0] = norm[s] = mu = 0;
28     passo = c = 1;
29     do{
30         i = lista[mu];
31         for (j = 0; j < N; j++){
32             if (rede[i][j]){
33                 x = verifique(j, c);
34                 if (!x){
35                     lista[c] = j;
36                     dist[c] = passo;
37                     norm[j] = passo;
38                     c++;
39                 }
40             }
41         }
42         if (dist[mu+1] > dist[mu]) passo++;
43         mu++;
44     } while (!(mu>N));
45     return —passo;
46 }
47
48 //Inicializa os vetores lista[i] e dist[i]
49 void limpel (){
50     int i;
51     for (i = 0; i < N; i++) lista[i] = dist[i] = norm[i] = -1;
52 }
53
54 //Inicializa a construção da matriz distância

```

```

55 void caminho (int n){
56     int i, j;
57     for (i = 0; i < n; i++){
58         for (j = i+1; j < n; j++){
59             limpe1();
60             busca(i, j);
61             DIST[j][i] = DIST[i][j] = norm[j];
62         }
63     }
64 }
65
66 /*****
67 Calcula do caminho mínimo médio
68 *****/
69 float soma1(int n){
70     int i, j, soma=0;
71     for (i = 0; i < n; i++){
72         for (j = i+1; j < n; j++) soma += DIST[i][j];
73     return (float)soma/(n*(n-1)/2);
74 }
75
76 /*****
77 Coeficiente de Agrupamento
78 *****/
79 //Calcula o Coeficiente de Agrupamento
80 float soma2 (int n){
81     int i;
82     float count=0.;
83     for (i = 0; i < n; i++) if (C[i]) count += C[i];
84     return count/n;
85 }
86
87 void limpe2 (){
88     int i;
89     for (i = 0; i < N; i++){

```

```

90     C[i] = T[i] = 0;
91 }
92 }
93
94 //Constroi o vetor com os Coeficientes de Agrupamento de cada nó
95 void Ci (int n){
96     int i;
97     for (i = 0; i < n; i++){
98         if (k[i] > 1) C[i] = (float)(2*T[i])/(k[i]*(k[i]-1));
99         else C[i] = 0;
100     }
101 }
102
103 //Constroi o vetor com o número de conexões que existem entre os
    vizinhos de i compara os nós l e c
104 void comparar (int l, int c, int n){
105     int j;
106     for (j = 0; j < n; j++) if (rede[l][j] && rede[c][j]) T[j]++;
107 }
108
109 //Verifica as conexões entre os vizinhos do nó i
110 void coef (int n){
111     int i, j;
112     for (i = 0; i < n-1; i++){
113         if (k[i] > 1){
114             for (j = i+1; j < n; j++) if (rede[i][j]) comparar(i, j, n);
115         }
116         else T[i]=0;
117     }
118 }
119
120 //Função do coeficiente de agrupamento
121 float coefagrup (int n){
122     float L;
123     limpe2();

```

```

124     coef(n);
125     Ci(n);
126     L = soma2(n);
127     return L;
128 }
129
130 /*****
131 Inicializador
132 *****/
133 void inicializar () {
134     int i, j;
135     for (i = 0; i < N; i++){
136         k[i] = C[i] = T[i] = 0;
137         for (j = 0; j < N; j++) DIST[i][j] = rede[i][j] = 0;
138     }
139 }
140
141 /*****
142 Rede Livre de Escala
143 *****/
144 //Conecta todos os nós iniciais
145 void livre2 () {
146     int i, j;
147     for (i = 0; i < mo; i++){
148         for (j = i+1; j < mo; j++) rede[i][j] = rede[j][i] = 1;
149     }
150 }
151
152 int ki (int n){
153     int i, j, count, sk=0;
154     for (i = 0; i < n; i++){
155         count = 0;
156         for (j = 0; j < n; j++){
157             if (rede[i][j]){
158                 count++;

```

```

159         sk++;
160     }
161 }
162     k[i] = count;
163 }
164     return sk;
165 }
166
167 //Calcula a frequência relativa do grau por nó
168 void probki (int sk, int n){
169     int i;
170     for (i = 0; i < n; i++) pk[i] = (float)k[i]/sk;
171 }
172
173 //Cria o vetor aleatório
174 void aleat (int n){
175     int i;
176     for (i = 0; i < n; i++) randon[i] = ((double)rand())/RAND_MAX;
177 }
178
179 //Cria um arquivo com os dados de crescimento da rede livre de escala
180 //Constroi a Rede Livre de Escala
181 void inicio3 (int c){
182     int i, j, count;
183     float K, L, Ag;
184     FILE *incid;
185     char arq1[20];
186     sprintf (arq1, "d-livre%d.txt", c);
187     incid = fopen(arq1, "w");
188     inicializar();
189     livre2();
190     fprintf (incid, "i\t<k>\t<l>\t<C>\n");
191     for (j = mo; j < N; j++){
192         K = ki(j);
193         probki(K, j);

```

```

194     K = (float)K/j;
195     caminho(j);
196     L = somal(j);
197     Ag = coefagrup(j);
198     fprintf (incid, "%d\t%f\t%f\t%f\n", j, K, L, Ag);
199     do{
200         count = 0;
201         aleat(j);
202         for (i = 0; i < j; i++){
203             if (pk[i] > randon[i]){
204                 teste[i] = 1;
205                 count++;
206             }
207             else teste[i] = 0;
208         }
209     } while (count != m);
210     for (i = 0; i < j; i++) rede[i][j] = rede[j][i] = teste[i];
211 }
212 K = (float)ki(j)/j;
213 caminho(j);
214 L = somal(j);
215 Ag = coefagrup(j);
216 fprintf (incid, "%d\t%f\t%f\t%f\n", j, K, L, Ag);
217 fclose (incid);
218 }
219
220 /*****
221 Rede Mundo Pequeno
222 *****/
223 int excluir (int i){
224     int j, count;
225     do{
226         for (j = 0; j < N; j++){
227             if (rede[i][j]) randon[j] = ((double)rand())/RANDMAX;
228             else randon[j] = 1;

```

```

229     }
230     count = 0;
231     for (j = 0; j < N; j++){
232         if (random[j] <= p){
233             count++;
234             random1[j] = -1;
235         }
236         else random1[j] = 0;
237     }
238 } while (count != m && count != 0);
239 return count;
240 }
241
242 void criar (int i){
243     int j, count;
244     do{
245         for (j = 0; j < N; j++){
246             if(rede[i][j]) random[j] = 1;
247             else random[j] = ((double)rand())/RANDMAX;
248         }
249         count = 0;
250         for (j = 0; j < N; j++){
251             if(random[j] <= p && i!=j){
252                 count++;
253                 random2[j] = 1;
254             }
255             else random2[j] = 0;
256         }
257     } while (count != m);
258 }
259
260 void realocar(){
261     int i, j, count;
262     for (i = 0; i < N; i++){
263         count = excluir(i);

```

```

264     if (count){
265         criar(i);
266         for (j = 0; j < N; j++){
267             rede[i][j] = rede[i][j] + randon1[j];
268             rede[i][j] = rede[i][j] + randon2[j];
269             rede[j][i] = rede[i][j];
270         }
271     }
272 }
273 }
274
275 void inicio5(){
276     int passo=0;
277     inicializar();
278     inicio4();
279     do{
280         realocar();
281         passo++;
282     } while (passo < V);
283 }
284
285 /*****
286 Funções de impressão em arquivo
287 *****/
288 //Cria arquivo com a lista de interacao
289 void incidencia (int c, int esc){
290     FILE *incid;
291     int i, j;
292     char arq1[20];
293     if(esc == 1) sprintf (arq1, "inc-livre%d.txt", c);
294     else if(esc == 2) sprintf (arq1, "inc-mundo%d.txt", c);
295     incid = fopen(arq1, "w");
296     for (i = 0; i < N; i++){
297         for (j = i+1; j < N; j++) if (rede[i][j]) fprintf (incid, "%d %d\\
n", i, j);

```

```

298     }
299     fclose(incid);
300 }
301
302 //Cria arquivo com a matriz distância
303 void distancia (int c, int esc){
304     FILE *incid;
305     int i, j;
306     char arq1[20];
307     if(esc == 1) sprintf(arq1, "dis-livre%d.txt", c);
308     else if(esc == 2) sprintf(arq1, "dis-mundo%d.txt", c);
309     incid = fopen(arq1, "w");
310     for (i = 0; i < N; i++){
311         for (j = 0; j < N; j++) fprintf(incid, "%d\t", DIST[i][j]);
312         fprintf(incid, "\n");
313     }
314     fclose(incid);
315 }
316
317 //Cria arquivo com os graus locais e agrupamentos locais
318 void kgraus(int c, int esc){
319     FILE *incid;
320     int i, j;
321     char arq1[20];
322     if (esc == 1) sprintf (arq1, "k-livre%d.txt", c);
323     else if (esc == 2) sprintf (arq1, "k-mundo%d.txt", c);
324     incid = fopen(arq1, "w");
325     fprintf (incid, "k\tC\n");
326     for (i = 0; i < N; i++) fprintf (incid, "%d\t%f\n", k[i], C[i]);
327     fclose(incid);
328 }
329
330 /*****
331 FUNÇÃO PRINCIPAL
332 *****/

```

```

333 int main(){
334     float L, K, Ag;
335     int escolha, c, i, j;
336     c = 0;
337     FILE *grid;
338     printf ("1 – Livre de Escala: \n2 – Rede Mundo pequeno \n");
339     scanf("%d", &escolha);
340     if (escolha == 1) grid=fopen("aleat1.txt","w");
341     else if (escolha == 2) grid=fopen("aleat2.txt","w");
342     fprintf(grid, "amostra\tGrau\tCMM\tAgrup\n");
343     do {
344         switch(escolha){
345             case 1:{
346                 inicializar();
347                 inicio1(); //Inicializa a rede livre de escala
348                 break;
349             }
350             case 2:{
351                 inicializar();
352                 inicio2(); //Inicializa a rede mundo pequeno
353                 break;
354             }
355         }
356         //Grau Medio
357         K = (float) ki(N)/N;
358         //Caminho Minimo Medio
359         caminho(N);
360         //printf ("Matriz distancia\n");
361         L = somal(N);
362         //Coeficiente de Agrupamento Medio
363         Ag = coefagrup(N);
364         fprintf (grid, "%d\t%f\t%f\t%f\n", c, K, L, Ag);
365         incidencia(c, escolha);
366         distancia(c, escolha);
367         kgraus(c, escolha);

```

```

368     c++;
369 } while (c < max);
370 fclose(grid);
371 return 0;
372 }

```

Apêndice B

```

1 #define N 10 //Tamanho da rede
2 #define d 1.00 //Densidade da população
3 #define g 0.05 //Parâmetro de ganho
4 #define pa 0.01 //Probabilidade de ativação espontânea (pa>=0)
5 #define sa 0.01 //Nível de atividade espontânea
6 #define P 500 //Número de passos
7 #define A 10 //Número de amostra
8
9 //Valores dos coeficientes acoplados
10 #define c1 1
11 #define c2 1
12 #define c3 1
13 #define c4 1
14 #define ieee 0.0000000000000001
15 #define I ((int)(d*N*N))
16
17 int m[N][N], m2[N][N], a[N][N], a2[N][N], inc[I][I]; //Matriz dos m's
18 float S1[N][N], S2[N][N]; //Matriz dos valores de atividade
19
20 /*****
21 Propriedades da rede
22 *****/

```

23 Os códigos para caracterizar a topologia das redes são os mesmos do
 Apêndice A

```

24
25 /*****
26 Funções para inicializar
27 *****/
28 void inicio () {
29     int i, j;
30     for (i = 0; i < N; i++){
31         for (j = 0; j < N; j++){
32             m2[i][j] = m[i][j] = a2[i][j] = a[i][j] = 0;
33             S2[i][j] = S1[i][j] = ieee;
34         }
35     }
36 }
37
38 void distr () {
39     int i, j, c;
40     c = 0;
41     do {
42         i = rand() % N;
43         j = rand() % N;
44         if (a[i][j] == 0) {
45             S1[i][j] = (((double)rand())/RANDMAX) + rand()%2 - 1;
46             if (S1[i][j] > ieee) m[i][j] = 1;
47             c++;
48             a[i][j] = c;
49         }
50     } while (c < d*N*N);
51 }
52
53 /*****
54 Atualização da Matriz S1 de S2
55 *****/
56 void atualiz () {

```

```

57  int i, j;
58  for (i = 0; i < N; i++){
59      for (j = 0; j < N; j++){
60          S1[i][j] = S2[i][j];
61          S2[i][j] = ieee;
62          if (S1[i][j] > ieee) m[i][j] = 1;
63          else m[i][j] = 0;
64      }
65  }
66 }
67
68 /*****
69 Funções para a condição de contorno periódica
70 *****/
71 int cci (int i){
72     if (i < 0) return (N - 1);
73     if (i > (N - 1)) return 0;
74     return i;
75 }
76
77 int ccj (int j){
78     if (j < 0) return (N - 1);
79     if (j > (N - 1)) return 0;
80     return j;
81 }
82
83 /*****
84 Função para os coeficientes acoplados
85 *****/
86 int Coe (i, j){
87     if (i > 0){
88         if (j > 0) return c1;
89         else return c3;
90     } else {
91         if (j > 0) return c2;

```

```

92     else return c4;
93 }
94 }
95
96 /*****
97 Função para o valor de atividade (S)
98 *****/
99 float J (i, j){
100     float act;
101     act = tanh(g*(Coe(i - 1, j - 1)*S1[cci(i - 1)][ccj(j - 1)] + Coe(i -
        1, j)*S1[cci(i - 1)][j] + Coe(i - 1, j + 1)*S1[cci(i - 1)][ccj(j +
        1)] + Coe(i, j - 1)*S1[i][ccj(j - 1)] + Coe(i, j)*S1[i][j] + Coe(
        i, j + 1)*S1[i][ccj(j + 1)] + Coe(i + 1, j - 1)*S1[cci(i + 1)][ccj
        (j - 1)] + Coe(i + 1, j)*S1[cci(i + 1)][j] + Coe(i + 1, j + 1)*S1[
        cci(i + 1)][ccj(j + 1)]));
102     return act;
103 }
104
105 /*****
106 Função Secundária
107 *****/
108 void atividade (){
109     int i, j;
110     for (i = 0; i < N; i++){
111         for (j = 0; j < N; j++){
112             if (a[i][j]) S2[i][j] = J(i, j);
113         }
114     }
115     atualiz();
116 }
117
118 /*****
119 Função de atualização do MOVE()
120 *****/
121 void atualmove (){

```

```

122     int i, j;
123     for (i = 0; i < N; i++){
124         for (j = 0; j < N; j++){
125             a[i][j] = a2[i][j];
126             m[i][j] = m2[i][j];
127             S1[i][j] = S2[i][j];
128             a2[i][j] = m2[i][j] = 0;
129             S2[i][j] = ieee;
130         }
131     }
132 }
133
134
135 /*****
136 Função de movimento
137 *****/
138 void move () {
139     int i, j;
140     int l, c, count, z;
141     for (i = 0; i < N; i++){
142         for (j = 0; j < N; j++){
143             if (a[i][j]){
144                 if (m[i][j]){
145                     z = count = 0;
146                     do{
147                         l = rand()%3 - 1;
148                         c = rand()%3 - 1;
149                         z++;
150                     if (a[cci(i + l)][ccj(j + c)] == 0 && a2[cci(i + l)][ccj(j
                        + c)] == 0){
151                         a2[cci(i + l)][ccj(j + c)] = a[i][j];
152                         m2[cci(i + l)][ccj(j + c)] = m[i][j];
153                         a[i][j] = m[i][j] = 0;
154                         S2[cci(i + l)][ccj(j + c)] = S1[i][j];
155                         S1[i][j] = ieee;

```

```

156         count++;
157     }
158     } while (count == 0 && z < 6);
159     if (z >= 5){
160         a2[i][j] = a[i][j];
161         m2[i][j] = m[i][j];
162         a[i][j] = m[i][j] = 0;
163         S2[i][j] = S1[i][j];
164         S1[i][j] = ieee;
165     }
166     } else {
167         a2[i][j] = a[i][j];
168         m2[i][j] = m[i][j];
169         a[i][j] = m[i][j] = 0;
170         S2[i][j] = S1[i][j];
171         S1[i][j] = ieee;
172     }
173     }
174 }
175 }
176 atualmove();
177 }
178
179 /*****
180 Função de ativação espontânea
181 *****/
182 void prob (){
183     int i, j;
184     float ra;
185     for (i = 0; i < N; i++){
186         for (j = 0; j < N; j++){
187             if (S1[i][j] < ieee && a[i][j] > 0){
188                 ra = ((double)rand())/RAND_MAX;
189                 if (ra < pa){
190                     S1[i][j] = sa;

```

```

191         m[i][j] = 1;
192     }
193 }
194 }
195 }
196 }
197
198 /*****
199 Função soma de ativação espontânea
200 *****/
201 float som1 () {
202     int i, j;
203     float s;
204     s = 0.0;
205     for (i = 0; i < N; i++){
206         for (j = 0; j < N; j++){
207             s += S1[i][j];
208         }
209     }
210     return s;
211 }
212
213 /*****
214 Função soma de objetos ativos
215 *****/
216 int som2 () {
217     int i, j;
218     int s = 0;
219     for (i = 0; i < N; i++){
220         for (j = 0; j < N; j++){
221             s += m[i][j];
222         }
223     }
224     return s;
225 }

```

```

226
227 /*****
228 Construção da matriz de incidência
229 *****/
230 void pinc (int NP)
231 {
232     FILE *ar3;
233     int i, j;
234     char arq3[20];
235     sprintf (arq3,"%dincidencia%d.txt", (int)(d*100), NP);
236     ar3 = fopen(arq3,"w");
237     for (i = 0; i < I; i++){
238         for (j = i; j < I; j++){
239             if (inc[i][j]) fprintf (ar3, "%d\t%d\n",i ,j);
240         }
241     }
242 }
243
244 void incid(){
245     int i, j, l, c;
246
247     //Meio
248     for (i = 1; i < N-1 ; i++){
249         for (j = 1; j < N-1; j++){
250             if (a[i][j] && m[i][j]){
251                 for (l = i-1; l <= i+1; l++){
252                     for (c = j-1; c <= j+1; c++){
253                         if (a[i][j] != a[l][c] && a[l][c] && m[l][c])    inc[a[i][j]
                                ]-1][a[l][c]-1] = 1;
254                     }
255                 }
256             }
257         }
258     }
259

```

```

260 //Coluna 0
261 j = N-1;
262 for (i = 1; i < N-1; i++){
263     if (a[i][0] && m[i][0]){
264         for (l = i-1; l <= i+1; l++){
265             if (a[l][j] && m[l][j])    inc[a[i][0]-1][a[l][j]-1] = 1;
266             for (c = 0; c <= 1; c++){
267                 if (a[i][0] != a[l][c] && a[l][c] && m[l][c])    inc[a[i]
                    ][0]-1][a[l][c]-1] = 1;
268             }
269         }
270     }
271 }
272
273 //Coluna N-1
274 j = 0;
275 for (i = 1; i < N-1; i++){
276     if (a[i][N-1] && m[i][N-1]){
277         for (l = i-1; l <= i+1; l++){
278             if (a[l][j] && m[l][j])    inc[a[i][N-1]-1][a[l][j]-1] = 1;
279             for (c = N-2; c <= N-1; c++){
280                 if (a[i][N-1] != a[l][c] && a[l][c] && m[l][c])    inc[a[i][N
                    -1]-1][a[l][c]-1] = 1;
281             }
282         }
283     }
284 }
285
286 //Linha 0
287 i = N-1;
288 for (j = 1; j < N-1; j++){
289     if (a[0][j] && m[0][j]){
290         for (c = j-1; c <= j+1; c++){
291             if (a[i][c] && m[i][c])    inc[a[0][j]-1][a[i][c]-1] = 1;
292             for (l = 0; l <= 1; l++){

```

```

293         if (a[0][j] != a[1][c] && a[1][c] && m[1][c]) inc[a[0][j]
294             ]-1[a[1][c]-1] = 1;
295     }
296 }
297 }
298
299 //Linha N-1
300 i = 0;
301 for (j = 1; j < N-1; j++){
302     if (a[N-1][j] && m[N-1][j]){
303         for (c = j-1; c <= j+1; c++){
304             if (a[i][c] && m[i][c]) inc[a[N-1][j]-1][a[i][c]-1] = 1;
305             for (l = N-2; l <= N-1; l++){
306                 if (a[N-1][j] != a[l][c] && a[l][c] && m[l][c]) inc[a[N-1][j]
307                     ]-1[a[l][c]-1] = 1;
308             }
309         }
310     }
311
312 //Pontas
313 if (a[0][0] && m[0][0]){
314     i = N-1;
315     if (a[N-1][N-1] && m[N-1][N-1])
316         inc[a[0][0]-1][a[i][i]-1] = 1;
317     for (l = 0; l <= 1; l++){
318         if (a[l][i] && m[l][i]) inc[a[0][0]-1][a[l][i]-1] = 1;
319         if (a[i][l] && m[i][l]) inc[a[0][0]-1][a[i][l]-1] = 1;
320         for (c = 0; c <= 1; c++){
321             if (a[0][0] != a[l][c] && a[l][c] && m[l][c]) inc[a
322                 ][0][0]-1[a[l][c]-1] = 1;
323         }
324     }

```

```

325  if (a[0][N-1] && m[0][N-1]){
326      i = N-1;
327      if (a[N-1][0] && m[N-1][0])    inc[a[0][N-1]-1][a[N-1][0]-1] = 1;
328      for (c = N-2; c <= N-1; c++){
329          if (a[i][c] && m[i][c])    inc[a[0][N-1]-1][a[i][c]-1] = 1;
330      }
331      for (l = 0; l <= 1; l++){
332          if (a[l][0] && m[l][0])    inc[a[0][N-1]-1][a[l][0]-1] = 1;
333          for (c = N-2; c <= N-1; c++){
334              if (a[0][N-1] != a[l][c] && a[l][c] && m[l][c])    inc[a[0][N-1]-1][a[l][c]-1] = 1;
335          }
336      }
337  }
338  if (a[N-1][0] && m[N-1][0]){
339      i = 0;
340      if (a[0][N-1] && m[0][N-1])    inc[a[N-1][0]-1][a[0][N-1]-1] = 1;
341      for (c = 0; c <= 1; c++){
342          if (a[i][c] && m[i][c])    inc[a[N-1][0]-1][a[i][c]-1] = 1;
343      }
344      for (l = N-2; l <= N-1; l++){
345          if (a[l][N-1] && m[l][N-1])    inc[a[N-1][0]-1][a[l][N-1]-1] = 1;
346          for (c = 0; c <= 1; c++){
347              if (a[N-1][0] != a[l][c] && a[l][c] && m[l][c])    inc[a[N-1][0]-1][a[l][c]-1] = 1;
348          }
349      }
350  }
351  if (a[N-1][N-1] && m[N-1][N-1]){
352      i = 0;
353      if (a[0][0] && m[0][0])    inc[a[N-1][N-1]-1][a[0][0]-1] = 1;
354      for (c = N-2; c <= N-1; c++){
355          if (a[i][c] && m[i][c])    inc[a[N-1][N-1]-1][a[i][c]-1] = 1;
356      }
357      for (l = N-2; l <= N-1; l++){

```

```

358         if (a[l][i] && m[l][i]) inc[a[N-1][N-1]-1][a[l][i]-1] = 1;
359     for (c = N-2; c <= N-1; c++){
360         if (a[N-1][N-1] != a[l][c] && a[l][c] && m[l][c]) inc[a[N-1][N-1]-1][a[l][c]-1] = 1;
361     }
362 }
363 }
364 }
365
366 /*****
367 Função principal
368 *****/
369 int main () {
370     int i, t;
371     char arq1[20], arq2[20], arq3[20];
372     FILE *ar1, *ar2, *ar3;
373
374     sprintf (arq3, "%dpropor.txt", (int)(d*100));
375     ar3 = fopen(arq3, "w");
376     fprintf (ar3, "<k>\t<CMM>\t<Agrup>\n");
377     for (t = 0; t < A; t++){
378         sprintf(arq1, "%dautomato%d.txt", (int)(d*100), t);
379         sprintf(arq2, "%dtemporal%d.txt", (int)(d*100), t);
380         ar1 = fopen(arq1, "w");
381         ar2 = fopen(arq2, "w");
382         inicio();
383         distr();
384         fprintf(ar1, "%d\t%.14f\n", 0, som1());
385         fprintf(ar2, "%d\t%.14f\n", 0, som2());
386         for (i = 1; i < P; i++){
387             incid();
388             atividade();
389             move();
390             prob();
391             fprintf(ar1, "%d\t%.14f\n", i, som1());

```

```

392     fprintf(ar2, "%d\t%d\n", i, som2());
393 }
394 caminho();
395 fprintf(ar3, "%d\t%f\t%f\t%f\n", t, ki(), sl(), coefagrup());
396 pinc(t);
397 fclose(ar1);
398 fclose(ar2);
399 }
400 fclose(ar3);
401 return 0;
402 }

```

Apêndice C

```

1 #define N 40 //Tamanho da rede
2 #define d 0.125 //Densidade da populacao
3 #define d1 0.90 //Densidade da casta 1
4 #define g 0.05 //Parametro de ganho
5 #define pa 0.01 //Probabilidade de ativacao espontanea (pa>=0)
6 #define sa 0.01 //Nivel de atividade espontanea
7 #define P 1000 //Numero de passos
8 #define Am 10 //Numero de amostra
9
10 //Valores dos coeficientes acoplados
11 #define c1 1
12 #define c2 1
13 #define c3 1
14 #define c4 1
15 #define ieee 0.0000000000000001
16 #define I ((int)(d*N*N))
17 #define V ((int)(P/r))

```

```

18
19 //Probabilidade de conexao
20 #define p11 0.0123
21 #define p12 0.0086
22 #define p21 0.0036
23
24 int m[N][N], m2[N][N], cast1[N][N], cast2[N][N], a[N][N], a2[N][N], inc
    [I][I];
25 float S1[N][N], S2[N][N]; //Matrizes dos valores de atividade
26
27 /*****
28 Funcoes para inicializar
29 *****/
30 void inicio () {
31     int i, j;
32     for (i = 0; i < N; i++){
33         for (j = 0; j < N; j++){
34             cast2[i][j] = cast1[i][j] = a2[i][j] = a[i][j] = m2[i][j] = m[i
                ][j] = 0;
35             S2[i][j] = S1[i][j] = ieee;
36         }
37     }
38 }
39
40 void distr () {
41     int i, j, c;
42     c = 0;
43     //Inicializa para Casta 1
44     do {
45         i = rand()%N;
46         j = rand()%N;
47         if (a[i][j] == 0) {
48             S1[i][j] = (((double)rand())/RANDMAX) + (rand()%2) - 1;
49             if (S1[i][j] > ieee) m[i][j] = 1;
50             c++;

```

```

51     a[i][j] = c;
52     cast1[i][j] = 1;
53 }
54 } while (c < (d1*d*N*N));
55 //Inicializa para Casta 2
56 do{
57     i = rand()%N;
58     j = rand()%N;
59     if (a[i][j] == 0){
60         S1[i][j]= (((double)rand())/RAND_MAX) + rand()%2 - 1;
61         if (S1[i][j] > ieee) m[i][j] = 1;
62         c++;
63         a[i][j] = c;
64         cast1[i][j] = 2;
65     }
66 } while (c < d*N*N);
67 }
68
69 /*****
70 Atualizacao da Matriz S1 de S2
71 *****/
72 Mesma função do Apêndice B
73 void atualiz (){}
74
75 /*****
76 Funcoes para as margens
77 *****/
78 Mesmas funções do Apêndice B
79 int cci (int i){}
80 int ccj (int j){}
81
82 /*****
83 Funcoes atualizacao do MOVE()
84 *****/
85 void atualmove (){

```

```

86  int i, j;
87  for (i = 0; i < N; i++){
88      for (j = 0; j < N; j++){
89          a[i][j] = a2[i][j];
90          m[i][j] = m2[i][j];
91          S1[i][j] = S2[i][j];
92          S2[i][j] = ieee;
93          cast1[i][j] = cast2[i][j];
94          cast2[i][j] = m2[i][j] = a2[i][j] = 0;
95      }
96  }
97 }
98
99 /*****
100 Funcoes de movimento
101 *****/
102 void move () {
103     int i, j, l, c, count, z;
104     for (i = 0; i < N; i++){
105         for (j = 0; j < N; j++){
106             if (a[i][j]){
107                 if (m[i][j]){
108                     count = z = 0;
109                     do{
110                         l = rand()%3 - 1;
111                         c = rand()%3 - 1;
112                         z++;
113                     if (a[cci(i + 1)][ccj(j + c)] == 0 && a2[cci(i + 1)][ccj(j +
114                         c)] == 0){
115                         a2[cci(i+1)][ccj(j+c)] = a[i][j];
116                         m2[cci(i+1)][ccj(j+c)] = m[i][j];
117                         S2[cci(i+1)][ccj(j+c)] = S1[i][j];
118                         cast2[cci(i+1)][ccj(j+c)] = cast1[i][j];
119                         m[i][j] = a[i][j] = cast1[i][j] = 0;
120                         S1[i][j] = ieee;

```

```

120         count++;
121     }
122     } while (count == 0 && z < 6);
123     if (z >= 5){
124         a2[i][j] = a[i][j];
125         m2[i][j] = m[i][j];
126         S2[i][j] = S1[i][j];
127         cast2[i][j] = cast1[i][j];
128         m[i][j] = a[i][j] = cast1[i][j] = 0;
129         S1[i][j] = ieee;
130     }
131     } else {
132         a2[i][j] = a[i][j];
133         m2[i][j] = m[i][j];
134         S2[i][j] = S1[i][j];
135         cast2[i][j] = cast1[i][j];
136         m[i][j] = a[i][j] = cast1[i][j] = 0;
137         S1[i][j] = ieee;
138     }
139     }
140     }
141     }
142     atualmove();
143 }
144
145 /*****
146 Função de ativação espontânea
147 *****/
148 Mesma função do Apêndice B
149 void prob (){}
150
151 /*****
152 Função soma de ativação espontânea
153 *****/
154 float som1 (int ca){

```

```

155     int i, j;
156     float s;
157     s = 0.0;
158     for (i = 0; i < N; i++){
159         for (j = 0; j < N; j++){
160             if (cast1[i][j] == ca) s += S1[i][j];
161         }
162     }
163     return s;
164 }
165
166 /*****
167 Função soma de objetos ativos
168 *****/
169 int som2 (int ca){
170     int i, j, s;
171     s = 0;
172     for (i = 0; i < N; i++){
173         for (j = 0; j < N; j++){
174             if (cast1[i][j]==ca) s += m[i][j];
175         }
176     }
177     return s;
178 }
179
180 /*****
181 Função para os coeficientes acoplados
182 *****/
183 Mesmas funções do Apêndice B
184 float Coe1 (i,j){}
185 float Coe2 (i,j){}
186
187 /*****
188 Funções para o valor de atividade (S)
189 *****/

```

```

190 float J(i, j, zc){
191     float act, z1, z2, z3, z4, z5, z6, z7, z8, z9;
192     if (m[i][j]){
193         if (zc != cast1[cci(i-1)][ccj(j-1)] || (zc == 1)) z1 = Coe1(cci(i
            -1), ccj(j-1))*S1[cci(i-1)][ccj(j-1)];
194         else z1 = 0;
195         if (zc != cast1[cci(i-1)][j] || (zc == 1)) z2 = Coe1(cci(i-1),
            j)*S1[cci(i-1)][j];
196         else z2 = 0;
197         if (zc != cast1[cci(i-1)][ccj(j+1)] || (zc == 1)) z3 = Coe1(cci(i
            -1), ccj(j+1))*S1[cci(i-1)][ccj(j+1)];
198         else z3 = 0;
199         if (zc != cast1[i][ccj(j-1)] || (zc == 1)) z4 = Coe1(i, ccj(j
            -1))*S1[i][ccj(j-1)];
200         else z4 = 0;
201         z5 = c1*S1[i][j];
202         if (zc != cast1[i][ccj(j+1)] || (zc == 1)) z6 = Coe1(i, ccj(j
            +1))*S1[i][ccj(j+1)];
203         else z6 = 0;
204         if (zc != cast1[cci(i+1)][ccj(j-1)] || (zc == 1)) z7 = Coe1(cci(i
            +1), ccj(j-1))*S1[cci(i+1)][ccj(j-1)];
205         else z7 = 0;
206         if (zc != cast1[cci(i+1)][j] || (zc == 1)) z8 = Coe1(cci(i+1),
            j)*S1[cci(i+1)][j];
207         else z8 = 0;
208         if (zc != cast1[cci(i+1)][ccj(j+1)] || (zc == 1)) z9 = Coe1(cci(i
            +1), ccj(j+1))*S1[cci(i+1)][ccj(j+1)];
209         else z9 = 0;
210         act = tanh(g*(z1 + z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9));
211     } else {
212         if (zc != cast1[cci(i-1)][ccj(j-1)] || (zc == 1)) z1 = Coe2(cci(i
            -1), ccj(j-1))*S1[cci(i-1)][ccj(j-1)];
213         else z1 = 0;
214         if (zc != cast1[cci(i-1)][j] || (zc == 1)) z2 = Coe2(cci(i-1),
            j)*S1[cci(i-1)][j];

```

```

215     else z2 = 0;
216     if (zc != cast1[cci(i-1)][ccj(j+1)] || (zc == 1))    z3 = Coe2(cci(i
        -1), ccj(j+1))*S1[cci(i-1)][ccj(j+1)];
217     else z3 = 0;
218     if (zc != cast1[i][ccj(j-1)] || (zc == 1))    z4 = Coe2(i, ccj(j
        -1))*S1[i][ccj(j-1)];
219     else z4 = 0;
220                                     z5 = c4*S1[i][j];
221     if (zc != cast1[i][ccj(j+1)] || (zc == 1))    z6 = Coe2(i, ccj(j
        +1))*S1[i][ccj(j+1)];
222     else z6 = 0;
223     if (zc != cast1[cci(i+1)][ccj(j-1)] || (zc == 1))    z7 = Coe2(cci(i
        +1), ccj(j-1))*S1[cci(i+1)][ccj(j-1)];
224     else z7 = 0;
225     if (zc != cast1[cci(i+1)][j] || (zc == 1))    z8 = Coe2(cci(i+1),
        j)*S1[cci(i+1)][j];
226     else z8 = 0;
227     if (zc != cast1[cci(i+1)][ccj(j+1)] || (zc == 1))    z9 = Coe2(cci(i
        +1), ccj(j+1))*S1[cci(i+1)][ccj(j+1)];
228     else z9 = 0;
229     act = tanh(g*(z1 + z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9));
230 }
231 return act;
232 }
233
234 /*****
235 Função Secundária
236 *****/
237 void atividade () {
238     int i, j;
239     for (i = 0; i < N; i++){
240         for (j = 0; j < N; j++){
241             if (a[i][j]) S2[i][j] = J(i, j, cast1[i][j]);
242         }
243     }

```

```

244     atualiz();
245 }
246
247 /*****
248 Construção da matriz de incidência
249 *****/
250 void incid () {
251     int i, j, l, c;
252     float rincid;
253
254     //Meio
255     for (i = 1; i < N-1; i++){
256         for (j = 1; j < N-1; j++){
257             if (a[i][j] && m[i][j] && (cast1[i][j] == 1)){
258                 for (l = i-1; l <= i+1; l++){
259                     for (c = j-1; c <= j+1; c++){
260                         if (a[i][j] != a[l][c] && a[l][c] && m[l][c]){
261                             rincid = ((double)rand())/RANDMAX;
262                             if (cast1[l][c] == 1 && (rincid <= p11)){
263                                 inc[a[i][j]-1][a[l][c]-1] = 1;
264                                 inc[a[l][c]-1][a[i][j]-1] = 1;
265                             } else if (cast1[l][c] == 2 && (rincid <= p12)){
266                                 inc[a[i][j]-1][a[l][c]-1] = 1;
267                                 inc[a[l][c]-1][a[i][j]-1] = 1;
268                             }
269                         }
270                     }
271                 }
272             } else if (a[i][j] && m[i][j] && cast1[i][j] == 2){
273                 for (l = i-1; l <= i+1; l++){
274                     for (c = j-1; c <= j+1; c++){
275                         rincid = ((double)rand())/RANDMAX;
276                         if ((a[i][j] != a[l][c]) && a[l][c] && m[l][c] &&
277                             cast1[i][j] != cast1[l][c] && (rincid <= p21)){

```

```

278             inc[a[l][c]-1][a[i][j]-1] = 1;
279         }
280     }
281 }
282 }
283 }
284 }
285
286 //Coluna 0
287 j = N-1;
288 for (i = 1; i < N-1; i++){
289     if (a[i][0] && m[i][0] && cast1[i][0] == 1){
290         for (l = i-1; l <= i+1; l++){
291             if (a[l][j] && m[l][j]){
292                 rincid = ((double)rand())/RANDMAX;
293                 if (cast1[l][j] == 1 && (rincid <= p11)) {
294                     inc[a[l][j]-1][a[i][0]-1] = 1;
295                     inc[a[i][0]-1][a[l][j]-1] = 1;
296                 } else if (cast1[l][j] == 2 && (rincid <= p12)){
297                     inc[a[l][j]-1][a[i][0]-1] = 1;
298                     inc[a[i][0]-1][a[l][j]-1] = 1;
299                 }
300             }
301         }
302         for (c = 0; c <= 1; c++){
303             if (a[i][0] != a[l][c] && a[l][c] && m[l][c]){
304                 rincid = ((double)rand())/RANDMAX;
305                 if (cast1[l][c] == 1 && rincid <= p11){
306                     inc[a[l][c]-1][a[i][0]-1] = 1;
307                     inc[a[i][0]-1][a[l][c]-1] = 1;
308                 } else if (cast1[l][c] == 2 && rincid <= p12){
309                     inc[a[l][c]-1][a[i][0]-1] = 1;
310                     inc[a[i][0]-1][a[l][c]-1] = 1;
311                 }
312             }

```

```

313     }
314 } else if (a[i][0] && m[i][0] && cast1[i][0] == 2){
315     for (l = i-1; l <= i+1; l++){
316         rincid = ((double)rand())/RANDMAX;
317         if (a[l][j] && m[l][j] && cast1[i][0] != cast1[l][j] && (rincid
318             <= p21)){
319             inc[a[l][j]-1][a[i][0]-1] = 1;
320             inc[a[i][0]-1][a[l][j]-1] = 1;
321         }
322     }
323     for (c = 0; c <= 1; c++){
324         rincid = ((double)rand())/RANDMAX;
325         if (a[i][0] != a[l][c] && a[l][c] && m[l][c] && cast1[i][0] !=
326             cast1[l][c] && (rincid <= p21)){
327             inc[a[l][c]-1][a[i][0]-1]=1;
328             inc[a[i][0]-1][a[l][c]-1]=1;
329         }
330     }
331 }
332 //Coluna N-1
333 j = 0;
334 for (i = 1; i < N-1; i++){
335     if (a[i][N-1] && m[i][N-1] && cast1[i][N-1] == 1){
336         for (l = i-1; l <= i+1; l++){
337             if (a[l][j] && m[l][j]){
338                 rincid = ((double)rand())/RANDMAX;
339                 if (cast1[l][j] == 1 && rincid <= p11){
340                     inc[a[l][j]-1][a[i][N-1]-1] = 1;
341                     inc[a[i][N-1]-1][a[l][j]-1] = 1;
342                 } else if (cast1[l][j] == 2 && rincid <= p12){
343                     inc[a[l][j]-1][a[i][N-1]-1] = 1;
344                     inc[a[i][N-1]-1][a[l][j]-1] = 1;
345                 }

```

```

346     }
347     for (c = N-2; c <= N-1; c++){
348         if (a[i][N-1] != a[l][c] && a[l][c] && m[l][c]){
349             rincid = ((double)rand())/RAND_MAX;
350             if (cast1[l][c] == 1 && (rincid <= p11)){
351                 inc[a[i][N-1]-1][a[l][c]-1] = 1;
352                 inc[a[l][c]-1][a[i][N-1]-1] = 1;
353             } else if (cast1[l][c] == 2 && rincid <= p12){
354                 inc[a[i][N-1]-1][a[l][c]-1] = 1;
355                 inc[a[l][c]-1][a[i][N-1]-1] = 1;
356             }
357         }
358     }
359 }
360 } else if (a[i][N-1] && m[i][N-1] && cast1[i][N-1] == 2){
361     for (l = i-1; l <= i+1; l++){
362         rincid = ((double)rand())/RAND_MAX;
363         if (a[l][j] && m[l][j] && cast1[i][N-1] != cast1[l][j] &&
364             rincid <= p21){
365             inc[a[i][N-1]-1][a[l][j]-1] = 1;
366             inc[a[l][j]-1][a[i][N-1]-1] = 1;
367         }
368     }
369     for (c = N-2; c <= N-1; c++){
370         rincid = ((double)rand())/RAND_MAX;
371         if (a[i][N-1] != a[l][c] && a[l][c] && m[l][c] && cast1[i][
372             N-1] != cast1[l][c] && rincid <= p21){
373             inc[a[l][c]-1][a[i][N-1]-1] = 1;
374             inc[a[i][N-1]-1][a[l][c]-1] = 1;
375         }
376     }
377 }
378 //Linha 0

```

```

379  i = N-1;
380  for (j = 1; j < N-1; j++){
381      if (a[0][j] && m[0][j] && cast1[0][j] == 1){
382          for (c = j-1; c <= j+1; c++){
383              if (a[i][c] && m[i][c]){
384                  rincid = ((double)rand())/RANDMAX;
385                  if (cast1[i][c] == 1 && rincid <= p11){
386                      inc[a[0][j]-1][a[i][c]-1] = 1;
387                      inc[a[i][c]-1][a[0][j]-1] = 1;
388                  } else if (cast1[i][c] == 2 && rincid <= p12){
389                      inc[a[0][j]-1][a[i][c]-1] = 1;
390                      inc[a[i][c]-1][a[0][j]-1] = 1;
391                  }
392              }
393          for (l = 0; l <= 1; l++){
394              if (a[0][j] != a[l][c] && a[l][c] && m[l][c]){
395                  rincid = ((double)rand())/RANDMAX;
396                  if (cast1[l][c] == 1 && rincid <= p11){
397                      inc[a[0][j]-1][a[l][c]-1] = 1;
398                      inc[a[l][c]-1][a[0][j]-1] = 1;
399                  } else if (cast1[l][c] == 2 && rincid <= p12){
400                      inc[a[0][j]-1][a[l][c]-1] = 1;
401                      inc[a[l][c]-1][a[0][j]-1] = 1;
402                  }
403              }
404          }
405      }
406      } else if (a[0][j] && m[0][j] && cast1[0][j] == 2){
407          for (c = j-1; c <= j+1; c++){
408              rincid = ((double)rand())/RANDMAX;
409              if (a[i][c] && m[i][c] && cast1[0][j] != cast1[i][c] &&
410                  rincid <= p21){
411                  inc[a[0][j]-1][a[i][c]-1] = 1;
412                  inc[a[i][c]-1][a[0][j]-1] = 1;
413              }

```

```

413         for (l = 0; l <= 1; l++){
414             rincid = ((double)rand())/RANDMAX;
415             if (a[0][j] != a[l][c] && a[l][c] && m[l][c] && cast1[0][j]
                != cast1[l][c] && rincid <= p21){
416                 inc[a[0][j]-1][a[l][c]-1] = 1;
417                 inc[a[l][c]-1][a[0][j]-1] = 1;
418             }
419         }
420     }
421 }
422 }
423
424 //Linha N-1
425 i = 0;
426 for (j = 1; j < N-1; j++){
427     if (a[N-1][j] && m[N-1][j] && cast1[N-1][j] == 1){
428         for (c = j-1; c <= j+1; c++){
429             if (a[i][c] && m[i][c]){
430                 rincid = ((double)rand())/RANDMAX;
431                 if (cast1[i][c] == 1 && rincid <= p11){
432                     inc[a[N-1][j]-1][a[i][c]-1] = 1;
433                     inc[a[i][c]-1][a[N-1][j]-1] = 1;
434                 } else if (cast1[i][c] == 2 && rincid <= p12){
435                     inc[a[N-1][j]-1][a[i][c]-1] = 1;
436                     inc[a[i][c]-1][a[N-1][j]-1] = 1;
437                 }
438             }
439         }
440         for (l = N-2; l <= N-1; l++){
441             if (a[N-1][j] != a[l][c] && a[l][c] && m[l][c]){
442                 rincid = ((double)rand())/RANDMAX;
443                 if (cast1[l][c] == 1 && rincid <= p11){
444                     inc[a[N-1][j]-1][a[l][c]-1] = 1;
445                     inc[a[l][c]-1][a[N-1][j]-1] = 1;
446                 } else if (cast1[l][c] == 2 && rincid <= p12){

```

```

447         inc[a[l][c]-1][a[N-1][j]-1] = 1;
448     }
449 }
450 }
451 }
452 } else if (a[N-1][j] && m[N-1][j] && cast1[N-1][j] == 2){
453     for (c = j-1; c <= j+1; c++){
454         rincid = ((double)rand())/RANDMAX;
455         if (a[i][c] && m[i][c] && cast1[N-1][j] != cast1[i][c] &&
456             rincid <= p21){
457             inc[a[i][c]-1][a[N-1][j]-1] = 1;
458             inc[a[N-1][j]-1][a[i][c]-1] = 1;
459         }
460     }
461     for (l = N-2; l <= N-1; l++){
462         rincid=((double)rand())/RANDMAX;
463         if (a[N-1][j] != a[l][c] && a[l][c] && m[l][c] && cast1[N-1][j] != cast1[l][c] && rincid <= p21){
464             inc[a[l][c]-1][a[N-1][j]-1] = 1;
465             inc[a[N-1][j]-1][a[l][c]-1] = 1;
466         }
467     }
468 }
469
470 //Pontas
471 if (a[0][0] && m[0][0] && cast1[0][0] == 1){
472     i = N-1;
473     if (a[N-1][N-1] && m[N-1][N-1]){
474         rincid = ((double)rand())/RANDMAX;
475         if (cast1[N-1][N-1] == 1 && rincid <= p11){
476             inc[a[0][0]-1][a[i][i]-1] = 1;
477             inc[a[i][i]-1][a[0][0]-1] = 1;
478         } else if (cast1[N-1][N-1] == 2 && rincid <= p12){
479             inc[a[0][0]-1][a[i][i]-1] = 1;

```

```

480         inc[a[i][i]-1][a[0][0]-1] = 1;
481     }
482 }
483 for (l = 0; l <= 1; l++){
484     if (a[l][i] && m[l][i]){
485         rincid = ((double)rand())/RANDMAX;
486         if (cast1[l][i] == 1 && rincid <= p11){
487             inc[a[0][0]-1][a[l][i]-1] = 1;
488             inc[a[l][i]-1][a[0][0]-1] = 1;
489         } else if (cast1[l][i] == 2 && rincid <= p12){
490             inc[a[0][0]-1][a[l][i]-1] = 1;
491             inc[a[l][i]-1][a[0][0]-1] = 1;
492         }
493     }
494     if (a[i][l] && m[i][l]){
495         rincid = ((double)rand())/RANDMAX;
496         if (cast1[i][l] == 1 && rincid <= p11){
497             inc[a[0][0]-1][a[i][l]-1] = 1;
498             inc[a[i][l]-1][a[0][0]-1] = 1;
499         } else if (cast1[i][l] == 2 && rincid <= p12){
500             inc[a[0][0]-1][a[i][l]-1] = 1;
501             inc[a[i][l]-1][a[0][0]-1] = 1;
502         }
503     }
504     for (c = 0; c <= 1; c++){
505         if (a[0][0] != a[l][c] && a[l][c] && m[l][c]){
506             rincid = ((double)rand())/RANDMAX;
507             if (cast1[l][c] == 1 && rincid <= p11){
508                 inc[a[0][0]-1][a[l][c]-1] = 1;
509                 inc[a[l][c]-1][a[0][0]-1] = 1;
510             } else if (cast1[l][c] == 2 && rincid <= p12){
511                 inc[a[0][0]-1][a[l][c]-1] = 1;
512                 inc[a[l][c]-1][a[0][0]-1] = 1;
513             }
514         }

```

```

515     }
516 }
517 } else if (a[0][0] && m[0][0] && cast1[0][0] == 2){
518     i = N-1;
519     rincid = ((double)rand())/RAND_MAX;
520     if (a[N-1][N-1] && m[N-1][N-1] && cast1[0][0] != cast1[N-1][N
521         -1] && rincid <= p21){
522         inc[a[0][0]-1][a[i][i]-1] = 1;
523         inc[a[i][i]-1][a[0][0]-1] = 1;
524     }
525     for (l = 0; l <= 1; l++){
526         rincid = ((double)rand())/RAND_MAX;
527         if (a[l][i] && m[l][i] && cast1[0][0] != cast1[l][i] &&
528             rincid <= p21){
529             inc[a[0][0]-1][a[l][i]-1] = 1;
530             inc[a[l][i]-1][a[0][0]-1] = 1;
531         }
532         rincid = ((double)rand())/RAND_MAX;
533         if (a[i][l] && m[i][l] && cast1[0][0] != cast1[i][l] &&
534             rincid <= p21){
535             inc[a[0][0]-1][a[i][l]-1] = 1;
536             inc[a[i][l]-1][a[0][0]-1] = 1;
537         }
538     }
539     for (c = 0; c <= 1; c++){
540         rincid = ((double)rand())/RAND_MAX;
541         if (a[0][0] != a[l][c] && a[l][c] && m[l][c] && cast1[0][0] !=
542             cast1[l][c] && rincid <= p21){
543             inc[a[0][0]-1][a[l][c]-1] = 1;
544             inc[a[l][c]-1][a[0][0]-1] = 1;
545         }
546     }
547 }
548 }
549 if (a[0][N-1] && m[0][N-1] && cast1[0][N-1] == 1){
550     i = N-1;

```

```

546         if (a[N-1][0] && m[N-1][0]) {
547             rincid = ((double)rand())/RANDMAX;
548             if (cast1[N-1][0] == 1 && rincid <= p11){
549                 inc[a[0][N-1]-1][a[N-1][0]-1] = 1;
550                 inc[a[N-1][0]-1][a[0][N-1]-1] = 1;
551             } else if (cast1[N-1][0] == 2 && rincid <= p12){
552                 inc[a[0][N-1]-1][a[N-1][0]-1] = 1;
553                 inc[a[N-1][0]-1][a[0][N-1]-1] = 1;
554             }
555         }
556         for (c = N-2; c <= N-1; c++){
557             if (a[i][c] && m[i][c]) {
558                 rincid = ((double)rand())/RANDMAX;
559                 if (cast1[i][c] == 1 && rincid <= p11){
560                     inc[a[0][N-1]-1][a[i][c]-1] = 1;
561                     inc[a[i][c]-1][a[0][N-1]-1] = 1;
562                 } else if (cast1[i][c] == 2 && rincid <= p12){
563                     inc[a[0][N-1]-1][a[i][c]-1] = 1;
564                     inc[a[i][c]-1][a[0][N-1]-1] = 1;
565                 }
566             }
567         }
568         for (l = 0; l <= 1; l++){
569             if (a[l][0] && m[l][0]) {
570                 rincid = ((double)rand())/RANDMAX;
571                 if (cast1[l][0] == 1 && rincid <= p11){
572                     inc[a[0][N-1]-1][a[l][0]-1] = 1;
573                     inc[a[l][0]-1][a[0][N-1]-1] = 1;
574                 } else if (cast1[l][0] == 2 && rincid <= p12){
575                     inc[a[0][N-1]-1][a[l][0]-1] = 1;
576                     inc[a[l][0]-1][a[0][N-1]-1] = 1;
577                 }
578             }
579             for (c = N-2; c <= N-1; c++){
580                 if (a[0][N-1] != a[l][c] && a[l][c] && m[l][c]) {

```

```

581         rincid = ((double)rand())/RANDMAX;
582         if (cast1[l][c] == 1 && rincid <= p11){
583             inc[a[0][N-1]-1][a[l][c]-1] = 1;
584             inc[a[l][c]-1][a[0][N-1]-1] = 1;
585         } else if (cast1[l][c] == 2 && rincid <= p12){
586             inc[a[0][N-1]-1][a[l][c]-1] = 1;
587             inc[a[l][c]-1][a[0][N-1]-1] = 1;
588         }
589     }
590 }
591 }
592 } else if (a[0][N-1] && m[0][N-1] && cast1[0][N-1] == 2){
593     i = N-1;
594     rincid = ((double)rand())/RANDMAX;
595     if (a[N-1][0] && m[N-1][0] && cast1[0][N-1] != cast1[N-1][0] &&
        rincid <= p21){
596         inc[a[0][N-1]-1][a[N-1][0]-1]=1;
597         inc[a[N-1][0]-1][a[0][N-1]-1]=1;
598     }
599     for (c = N-2; c <= N-1; c++){
600         rincid=((double)rand())/RANDMAX;
601         if (a[i][c] && m[i][c] && cast1[0][N-1] != cast1[i][c] &&
            rincid <= p21){
602             inc[a[0][N-1]-1][a[i][c]-1] = 1;
603             inc[a[i][c]-1][a[0][N-1]-1] = 1;
604         }
605     }
606     for (l = 0; l <= 1; l++){
607         rincid = ((double)rand())/RANDMAX;
608         if (a[l][0] && m[l][0] && cast1[0][N-1] != cast1[l][0] &&
            rincid <= p21){
609             inc[a[0][N-1]-1][a[l][0]-1]=1;
610             inc[a[l][0]-1][a[0][N-1]-1]=1;
611         }
612         for (c = N-2; c <= N-1; c++){

```

```

613         rincid = ((double)rand())/RANDMAX;
614         if (a[0][N-1] != a[1][c] && a[1][c] && m[1][c] && cast1[0][N
        -1] != cast1[1][c] && rincid <= p21){
615             inc[a[0][N-1]-1][a[1][c]-1] = 1;
616             inc[a[1][c]-1][a[0][N-1]-1] = 1;
617         }
618     }
619 }
620 }
621 if (a[N-1][0] && m[N-1][0] && cast1[N-1][0] == 1){
622     i = 0;
623     if (a[0][N-1] && m[0][N-1]){
624         rincid = ((double)rand())/RANDMAX;
625         if (cast1[0][N-1] == 1 && rincid <= p11){
626             inc[a[N-1][0]-1][a[0][N-1]-1] = 1;
627             inc[a[0][N-1]-1][a[N-1][0]-1] = 1;
628         } else if (cast1[0][N-1] == 2 && rincid <= p12){
629             inc[a[N-1][0]-1][a[0][N-1]-1] = 1;
630             inc[a[0][N-1]-1][a[N-1][0]-1] = 1;
631         }
632     }
633     for (c = 0; c <= 1; c++){
634         if (a[i][c] && m[i][c]){
635             rincid=((double)rand())/RANDMAX;
636             if (cast1[i][c] == 1 && rincid <= p11){
637                 inc[a[N-1][0]-1][a[i][c]-1] = 1;
638                 inc[a[i][c]-1][a[N-1][0]-1] = 1;
639             } else if (cast1[i][c] == 2 && rincid <= p12){
640                 inc[a[N-1][0]-1][a[i][c]-1] = 1;
641                 inc[a[i][c]-1][a[N-1][0]-1] = 1;
642             }
643         }
644     }
645     for (l = N-2; l <= N-1; l++){
646         if (a[l][N-1] && m[l][N-1]){

```

```

647     rincid = ((double)rand())/RAND_MAX;
648     if (cast1[l][N-1] == 1 && rincid <= p11){
649         inc[a[N-1][0]-1][a[l][N-1]-1] = 1;
650         inc[a[l][N-1]-1][a[N-1][0]-1] = 1;
651     } else if (cast1[l][N-1] == 2 && rincid <= p12){
652         inc[a[N-1][0]-1][a[l][N-1]-1] = 1;
653         inc[a[l][N-1]-1][a[N-1][0]-1] = 1;
654     }
655 }
656 for (c = 0; c <= 1; c++){
657     if (a[N-1][0] != a[l][c] && a[l][c] && m[l][c]){
658         rincid = ((double)rand())/RAND_MAX;
659         if (cast1[l][c] == 1 && rincid <= p11){
660             inc[a[N-1][0]-1][a[l][c]-1] = 1;
661             inc[a[l][c]-1][a[N-1][0]-1] = 1;
662         } else if (cast1[l][c] == 2 && rincid <= p12){
663             inc[a[N-1][0]-1][a[l][c]-1] = 1;
664             inc[a[l][c]-1][a[N-1][0]-1] = 1;
665         }
666     }
667 }
668 }
669 } else if (a[N-1][0] && m[N-1][0] && (cast1[N-1][0] == 2)){
670     i = 0;
671     rincid = ((double)rand())/RAND_MAX;
672     if (a[0][N-1] && m[0][N-1] && cast1[N-1][0] != cast1[0][N-1] &&
        rincid <= p21){
673         inc[a[N-1][0]-1][a[0][N-1]-1] = 1;
674         inc[a[0][N-1]-1][a[N-1][0]-1] = 1;
675     }
676 }
677 for (c = 0; c <= 1; c++){
678     rincid = ((double)rand())/RAND_MAX;
679     if (a[i][c] && m[i][c] && cast1[N-1][0] != cast1[i][c] &&
        rincid <= p21){

```

```

680         inc [a[N-1][0]-1][a[i][c]-1] = 1;
681         inc [a[i][c]-1][a[N-1][0]-1] = 1;
682     }
683 }
684 for (l = N-2; l <= N-1; l++){
685     rincid = ((double)rand())/RAND_MAX;
686     if (a[l][N-1] && m[l][N-1] && cast1[N-1][0] != cast1[l][N
        -1] && rincid <= p21){
687         inc [a[N-1][0]-1][a[l][N-1]-1] = 1;
688         inc [a[l][N-1]-1][a[N-1][0]-1] = 1;
689     }
690     for (c = 0; c <= 1; c++){
691         rincid = ((double)rand())/RAND_MAX;
692         if (a[N-1][0] != a[l][c] && a[l][c] && m[l][c] && cast1[N
            -1][0] != cast1[l][c] && rincid <= p21){
693             inc [a[N-1][0]-1][a[l][c]-1] = 1;
694             inc [a[l][c]-1][a[N-1][0]-1] = 1;
695         }
696     }
697 }
698 }
699 if (a[N-1][N-1] && m[N-1][N-1] && cast1[N-1][N-1] == 1){
700     i = 0;
701     if (a[0][0] && m[0][0]){
702         rincid = ((double)rand())/RAND_MAX;
703         if (cast1[0][0] == 1 && rincid <= p11){
704             inc [a[N-1][N-1]-1][a[0][0]-1] = 1;
705             inc [a[0][0]-1][a[N-1][N-1]-1] = 1;
706         } else if (cast1[0][0] == 2 && rincid <= p12){
707             inc [a[N-1][N-1]-1][a[0][0]-1] = 1;
708             inc [a[0][0]-1][a[N-1][N-1]-1] = 1;
709         }
710     }
711     for (c = N-2; c <= N-1; c++){
712         if (a[i][c] && m[i][c]){

```

```

713     rincid = ((double)rand())/RANDMAX;
714     if (cast1[i][c] == 1 && rincid <= p11){
715         inc[a[N-1][N-1]-1][a[i][c]-1] = 1;
716         inc[a[i][c]-1][a[N-1][N-1]-1] = 1;
717     } else if (cast1[i][c] == 2 && rincid <= p12){
718         inc[a[N-1][N-1]-1][a[i][c]-1] = 1;
719         inc[a[i][c]-1][a[N-1][N-1]-1] = 1;
720     }
721     }
722 }
723 for (l = N-2; l <= N-1; l++){
724     if (a[l][i] && m[l][i]){
725         rincid = ((double)rand())/RANDMAX;
726         if (cast1[l][i] == 1 && rincid <= p11){
727             inc[a[N-1][N-1]-1][a[l][i]-1] = 1;
728             inc[a[l][i]-1][a[N-1][N-1]-1] = 1;
729         } else if (cast1[l][i] == 2 && rincid <= p12){
730             inc[a[N-1][N-1]-1][a[l][i]-1] = 1 ;
731             inc[a[l][i]-1][a[N-1][N-1]-1] = 1 ;
732         }
733     }
734     for (c = N-2; c <= N-1; c++){
735         if (a[N-1][N-1] != a[l][c] && a[l][c] && m[l][c]){
736             rincid = ((double)rand())/RANDMAX;
737             if (cast1[l][c] == 1 && rincid <= p11){
738                 inc[a[N-1][N-1]-1][a[l][c]-1] = 1;
739                 inc[a[l][c]-1][a[N-1][N-1]-1] = 1;
740             } else if (cast1[l][c] == 2 && rincid <= p12){
741                 inc[a[N-1][N-1]-1][a[l][c]-1] = 1;
742                 inc[a[l][c]-1][a[N-1][N-1]-1] = 1;
743             }
744         }
745     }
746 }
747 } else if (a[N-1][N-1] && m[N-1][N-1] && cast1[N-1][N-1] == 2){

```

```

748         i = 0;
749         rincid = ((double)rand())/RAND_MAX;
750         if (a[0][0] && m[0][0] && cast1[N-1][N-1] != cast1[0][0] &&
            rincid <= p21){
751             inc[a[N-1][N-1]-1][a[0][0]-1] = 1;
752             inc[a[0][0]-1][a[N-1][N-1]-1] = 1;
753         }
754         for (c = N-2; c <= N-1; c++){
755             rincid = ((double)rand())/RAND_MAX;
756             if (a[i][c] && m[i][c] && cast1[N-1][N-1] != cast1[i][c] &&
                rincid <= p21){
757                 inc[a[N-1][N-1]-1][a[i][c]-1] = 1;
758                 inc[a[i][c]-1][a[N-1][N-1]-1] = 1;
759             }
760         }
761         for (l = N-2; l <= N-1; l++){
762             rincid = ((double)rand())/RAND_MAX;
763             if (a[l][i] && m[l][i] && cast1[N-1][N-1] != cast1[l][i]
                && rincid <= p21){
764                 inc[a[N-1][N-1]-1][a[l][i]-1] = 1;
765                 inc[a[l][i]-1][a[N-1][N-1]-1] = 1;
766             }
767             for (c = N-2; c <= N-1; c++){
768                 rincid = ((double)rand())/RAND_MAX;
769                 if (a[N-1][N-1] != a[l][c] && a[l][c] && m[l][c] && cast1[N-1][N-1] != cast1[l][c] && rincid <= p21){
770                     inc[a[N-1][N-1]-1][a[l][c]-1] = 1;
771                     inc[a[l][c]-1][a[N-1][N-1]-1] = 1;
772                 }
773             }
774         }
775     }
776 }
777
778 /*****

```

779 **Função principal**

```

780 *****/
781 int main () {
782     int i, t;
783     char arq1[20], arq2[20];
784     FILE *ar1, *ar2;
785     for (t = 0; t < Am; t++){
786         sprintf(arq1, "%dautomato%d.txt", (int)(d*100), t);
787         sprintf(arq2, "%dtemporal%d.txt", (int)(d*100), t);
788         ar1 = fopen(arq1, "w");
789         ar2 = fopen(arq2, "w");
790         inicio();
791         distr();
792         fprintf(ar1, "%d\t%.14f\t%.14f\n", 0, som1(1), som1(2));
793         fprintf(ar2, "%d\t%d\t%d\n", 0, som2(1), som2(2));
794         incid();
795         for (i = 1; i < P; i++){
796             atividade();
797             move();
798             prob();
799             fprintf(ar1, "%d\t%.14f\t%.14f\n", i, som1(1), som1(2));
800             fprintf(ar2, "%d\t%d\t%d\n", i, som2(1), som2(2));
801         }
802         fclose(ar1);
803         fclose(ar2);
804     }
805     return 0;
806 }
```