Carlos Pietro C.L.G. Rebouças

Cantando no Tempo e Espaço:

Um Modelo Computacional da Dinâmica Evolutiva de Passeriformes

Dissertação apresentada ao
Programa de Pós-Graduação em Física da
Universidade Federal da Bahia

Flora Souza Bacelar (Orientadora)
Hilton F. Japyassu (Coorientador)
Vitor Passos Rios (Coorientador)

Salvador, BA
2019

Resumo

Os cantos dos Passeriformes são vocalizações longas e complexas produzidas pelos machos na temporada de acasalamento que possuem duas funções principais: atrair a fêmea para o acasalamento e defender o território do macho. Esta ordem é composta por duas subordens, os Suboscines, que possuem pouco efeito de aprendizagem, e os Oscines, que apresentam aprendizagem socialmente mediada. A evolução no canto dos Passeriformes é um caso de seleção sexual por escolha pela fêmea, porém nos Oscines a componente social do aprendizado também influencia no processo evolutivo. Nesse trabalho construímos um modelo computacional baseado em agentes para avaliar como diferenças no aprendizado e discriminação de cantos entre Oscines e Subocines impactam sobre sua dinâmica evolutiva. Para isso foi desenvolvido um modelo onde a aptidão não é calculada explicitamente e depende apenas da própria dinâmica social, em um ambiente homogêneo e desprovido de barreiras geográficas. Três aspectos da dinâmica social foram considerados: i) o processo de seleção de parceiro sexual pela fêmea através do canto, ii) o aprendizado socialmente mediado do canto dos filhotes machos de Oscines e iii) a competição intraespecífica. Para a avaliação dos resultados foram geradas redes de reconhecimento, onde os nós representam os Passeriformes machos e fêmeas e as arestas representam os acasalamentos possíveis. Avaliamos diferenças nas redes geradas por Oscines e Suboscines com relação ao número de componentes, que relacionamos com o fenômeno de especiação, e a modularidade, que relacionamos com a formação de dialetos comuns a pássaros de uma mesma comunidade.

Palavras-chave: Modelagem Baseada em Agentes, Redes Complexas, Plasticidade do Canto, Evolução.

Abstract

The songs of Passeriformes are long complex vocalisations produced by males in breeding season that have two mainly functions: attract a mate and defend the territory. This order are composed by two suborders, the Suboscines, that don't have so much learning effect, and the Oscines, that exhibit socially mediated learning. The evolution of song in Passeriformes is a case of sexual selection by female choice, however in the Oscines the social component of learning also influences the evolutionary process. On this work we make a model where the fitness isn't explicitly calculated and depends only of the own social dynamics, in a homogeneous environment with no geographical barriers. We consider three aspects of social dynamics: i) the process of selection of mate by female through song, ii) the socially mediated learning of song by Oscines male offsprings and iii) the intraspecific competition. For evaluate the results we build recognition networks, where the nodes represent the male and female Passeriformes, and the edges represent the possible matings. We evaluated differences in the networks generated by Oscines and Suboscines in relation to the number of components, related to the phenomenon of speciation, and modularity, which we relate to the formation of common dialects to birds of the same community.

Keywords: Agents Based Modeling, Complex Networks, Plasticity of Bird Song, Evolution.

Agradecimentos

Primeiramente gostaria de agradecer à minha avó Raquel, que mesmo tendo apenas a 3ª série do fundamental, foi responsável pela alfabetização de quase todas as crianças do Cedro na época em que foi professora, e ao meu avô Josino, que sempre foi um exemplo de pai para a família Leal e todas as demais famílias do Cedro. À minha falecida avó Arlinda, matriarca da família Rebouças, que ao mesmo tempo sonhava que eu crescesse logo pra me formar, e que eu não crescesse nunca, pra ficar pequenininho para sempre. Caso ela estivesse aqui hoje, veria que de certa maneira os dois sonhos se realizaram.

Gostaria de agradecer aos meus pais, Deide e Zé de Arlinda, que são meus amores, meu porto seguro, me incentivaram a ter gosto pelos estudos e investiram tempo e dinheiro em minha formação acadêmica. Este trabalho é tanto uma realização de vocês quanto minha.

Gostaria de agradecer à minha família, que sempre me incentivou e apoiou com todo amor e carinho. Em especial a Simone, que é minha segunda mãe desde que eu era bebê, a Tia Carmem, que é minha segunda mãe desde quando eu era criança, e a Tia Jô, que também é minha segunda mãe hoje. Gostaria de fazer um agradecimento especial também ao meu padrinho tio Josias e a meu primo e irmão do coração Danilo, com quem sempre posso contar para tudo.

Agradeço também a Dayse, que é minha irmã de sangue e do coração, e que sempre arruma um jeito de tornar minha vida mais feliz. Agradeço também a Tiago e Murilo, que são meus irmãos do coração, e que por mais tempo que fiquemos sem se encontrar, sempre parece que nunca saímos um do lado do outro. À Beatriz Cravo, por todo amor e por todo incentivo e compreensão nesse último semestre.

Gostaria de agradecer a todos os meus amigos por todo carinho e suporte emocional que vocês me proporcionam todos os dias. Agradeço também aos meus colegas do Instituto de Física da UFBA, em especial a Zezinho, Lucas e Pedro, que junto comigo compunham o Trio Nordestino (eramos 4 porque Pedro foi suplente quando Lucas viajou, reza a lenda que teve até edital de seleção), e a Raíssa, Aline, Fernanda e Bárbara, que cursaram inúmeras disciplinas junto comigo e foram as melhores companhias que eu poderia querer.

Gostaria de agradecer à minha orientadora, Profa. Flora Bacelar, que me apresentou a ideia desse projeto que me chamou muito a atenção numa época em que eu estava bem confuso quanto ao que gostaria de fazer, que me ensinou sobre diversas ferramentas de modelagem e que foi uma orientadora excelente, eficiente e sobretudo muito paciente.

Gostaria de agradecer aos meus coorientadores, o Prof. Hilton Japyassu, que me orientou com muita eloquência sobre os aspectos fenomenológicos do sistema e apresentou ótimos *insights* sobre quais poderiam ser simplificado no modelo, e o Prof. Vitor Rios que me ajudou muito com a fenomenologia, com a implementação, com o tratamento de dados e com uma revisão muito rica dessa dissertação, sempre com muito tato e eficiência.

Gostaria também de agradecer a todo corpo docente do Instituto de Física da UFBA, em especial ao Prof. Garcia Vivas pelas disciplinas de introdução a modelagem baseada em agentes que tornaram possível a realização desse trabalho e ao corpo de servidores técnico-administrativos.

Gostaria de agradecer a CAPES pelo apoio financeiro.

Pois se for o ka, virá como um vento, e seus planos resistirão a ele tanto quanto um celeiro a um ciclone.

Stephen King
A Torre Negra vol. IV:
"Mago e Vidro"

Dedico essa dissertação aos meus pais e avós,
que sonharam com ela muito antes de mim.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este trabalho é uma inciativa criada pela interseção entre o projeto temático 11 "Genes, canto e socialidade: diferentes abordagens no estudo da variação geográfica", coordenado por H. F. Japyassú e H. Batalha Filho (UFBA) e o projeto integrador 1 "Modelagem matemática, computacional e estatística aplicada à ecologia e evolução", coordenado por F. S. Bacelar (UFBA) e P. I. Prado (USP), pertencentes ao INCT IN-TREE, INCT em Estudos Interdisciplinares e Transdisciplinares em Ecologia e Evolução. A ideia por trás dessa colaboração é o desenvolvimento de um modelo evolutivo e cultural, utilizando ferramentas da física estatística e sistemas complexos, para avaliar a influência do canto na dinâmica evolutiva de Passeriformes.

Os cantos dos Passeriformes são "vocalizações longas e complexas produzidas pelos machos na temporada de acasalamento"(CATCHPOLE; SLATER, 2008). Eles possuem duas funções principais, **atrair a fêmea para o acasalamento** e **defender o território**. A ordem dos Passeriformes é formada por duas subordens, os Suboscines, que possuem pouco efeito de aprendizagem, e os Oscines, que apresentam aprendizagem socialmente mediada. Além disso os Oscines possuem uma maior plasticidade e uma menor discriminação quanto aos cantos pertencentes a sua espécie (FREEMAN; MONTGOMERY; SCHLUTER, 2017).

A evolução no canto dos Passeriformes é explicada como um caso de seleção sexual por escolha pela fêmea, porém nos Oscines a componente social do aprendizado também influencia no processo evolutivo. Aprendizagem socialmente mediada pode acelerar a evolução genética e a especiação, pois cantos aprendidos são frutos de evolução genética assim como de evolução cultural, sendo essa última mais rápida pode aumentar as taxas de mudança (VERZIJDEN et al., 2012; CHEBIB; MARRIOTT, 2016; LACHLAN; SERVEDIO, 2004). Porém o contrário também pode acontecer, por exemplo, em cenários onde a seleção natural ou sexual impulsionam a divergência genética, o aprendizado socialmente mediado pode mascarar esses efeitos e inibir a especiação (FREEMAN; MONTGOMERY; SCHLUTER, 2017; VERZIJDEN et al., 2012).

Sistemas evolutivos são em geral sistemas complexos, pois são compostos por um grande número de agentes interagentes e exibem emergência, que é um comportamento coletivo auto-organizado, muitas vezes não intuitivo, e independente de um controlador central (BOCCARA, 2010). Nosso sistema apresenta comportamentos emergentes a nível cultural (formação de dialetos) e a nível evolutivo (especiação). Podemos então construir um modelo através de uma bordagem *botton-up*, que além de ser utilizada na mecânica estatística também tem se mostrado eficaz para diversos outros sistemas complexos, como tráfego, *flocking*, mercado de ações (BONABEAU, 2002).

Dentre as ferramentas disponíveis para modelagem *botton-up*, escolhemos para esse trabalho construir um modelo baseado em agentes (MBA). Agentes autônomos são entidades situadas dentro de um ambiente, que percebem e agem continuamente sobre ele seguindo seus próprios objetivos (FRANKLIN; GRAESSER, 1996). Um MBA é construído de uma maneira bem intuitiva, gerando computacionalmente agentes que simulem o comportamento individual dos componentes do sistema, suas interações e o ambiente onde estão inseridos.

Construímos nesse trabalho um modelo onde a aptidão não é calculado explicitamente e depende apenas da própria dinâmica social dos Passeriformes em um ambiente homogêneo e desprovido de barreiras geográficas. Três aspectos da dinâmica social foram considerados: i) o processo de seleção do parceiro sexual pela fêmea através do canto, ii) o aprendizado socialmente mediado do canto dos filhotes machos de Oscines e iii) a competição intraespecífica (que gera heterogeneidade na distribuição espacial dos agentes em nosso modelo). Foram desconsiderados o papel desempenhado pelo canto na defesa de territórios e a aprendizagem na preferência das fêmeas.

Para investigarmos as propriedades emergentes do nosso sistema, construímos para cada instante de tempo uma rede, usando o critério de espécie por reconhecimento para acasalamento de Paterson e McEvey (1993). Nos concentramos em avaliar duas propriedades da rede: o número de componentes, que relacionamos com o fenômeno de especiação, e a modularidade, que relacionamos com a formação de dialetos diferentes entre comunidades.

No capítulo 2 apresentamos a fenomenologia do sistema que modelamos. Ele é divido em duas subseções: a subseção 2.1 contém uma introdução geral sobre teoria da evolução, em especial os conceitos de seleção sexual, deriva gênica, espécie e especiação que desempenham papeis centrais nesse modelo; já a subseção 2.2 apresenta as características específicas do canto dos Passeriformes, principalmente sobre o mecanismo de aprendizado do canto presente nos Oscines, tomando como base um **modelo do padrão auditivo** de tentilhões (*Fringilla coelebs*) (CATCHPOLE; SLATER, 2008), e o papel do canto na seleção sexual dos Passeriformes.

No capítulo 3 encontramos os materiais e métodos utilizados na construção do modelo. Ele começa com uma pequena introdução a abordagem de modelagem *botton-up*, mostrando como modelagem baseada em agentes e mecânica estatística são diferentes técnicas que utilizam uma mesma estrutura. Na seção 3.1 discutimos a definição de agente, características fenomenológicas que levam a construção de um modelo baseado em agentes e como é a estrutura e construção de um modelo desse tipo. Na seção 3.2 temos uma introdução as redes complexas e a explicação dos conceitos de conectividade, comunidades e modularidade, que utilizamos no tratamento de dados.

O capítulo 4 quatro contém a **visão geral** do modelo, descrita segundo o protocolo ODD (GRIMM et al., 2006), e os resultados originais obtidos nas simulações seguindo uma estrutura de artigo. Os **conceitos de design** e os **detalhes** do modelo foram descritos no apêndice A, e o código implementado em C++ está disponível no apêndice B.

E por fim, o capítulo 5 discorre sobre modificações que podem ser implementadas no nosso modelo para investigações futuras.

Capítulo 2

Evolução e Canto

Nesse capítulo abordaremos a fenomenologia do sistema, destacando as características necessárias para construção e compreensão do modelo. Faremos isso em duas seções, uma delas dedicada a uma discussão geral sobre a teoria da evolução e a outra sobre o canto, seu aprendizado, e seu papel na dinâmica evolutiva dos Passeriformes.

2.1 Evolução

A biologia evolutiva é uma grande ciência, possuindo uma ampla variedade de áreas de concentração, como genética molecular, morfologia e embriologia, com diferentes objetos de estudo e abordagens capazes de testar a ideia de evolução por seleção natural. Esta é uma das ideias mais poderosas e elegantes da ciência e ajuda a dar sentido a fenômenos biológicos em diferentes escalas, dos microrganismos presentes em uma gota de chuva a manadas de grandes animais. (RIDLEY, 2006)

Em 1859, Darwin se referia a evolução como "descendência com modificação". Podemos definir evolução de uma maneira mais precisa como uma "mudança ao longo do tempo por descendência com modificação"(HARRISON, 2001). Essas definições evocam a ideia da transformação das espécies ao longo do tempo, porém esta não é a única ideia por trás do que usualmente chamamos de teoria da evolução. Conforme Meyer e El-Hani (2005), o que chamamos de teoria da evolução é na verdade um conjunto de 5 teorias inter-relacionadas.

Primeiramente temos a teoria de que a evolução ocorre, ou seja, que as espécies não são imutáveis e variam no tempo. Esta é uma teoria que antecede Darwin, que em "A Origem das Espécies" apresenta um compilado maciço de evidências anteriores que a sustentam. Hoje existe um número ainda maior de evidências a favor da evolução, que podem ser divididas em 3 grupos: observações diretas em pequena escala, semelhanças homólogas entre grupos de seres vivos, e a ordenação dos principais grupos no registro fóssil (RIDLEY, 2006). As evidências não deixam espaço para dúvida de que os seres vivos evoluíram e continuam evoluindo.

A segunda teoria é a de que os seres vivos partilham ancestrais comuns. Novas espécies surgem de espécies ancestrais através de um processo denominado especiação, que será explicado na subseção 2.1.2. Isso gera um padrão de ramificação nas linhagens de espécies para o qual se pode fazer uma analogia com uma árvore, onde cada ramo representaria uma linhagem de gerações que surge por especiação, conforme podemos ver no diagrama desenhado por Darwin (1859) reproduzido na figura

2.1. Dessa forma caminhando para trás no tempo (ou seguindo a analogia, descendo aos galhos mais baixos) é possível encontrar ancestrais comuns entre quaisquer espécies. Em algum nível, todos os galhos estão conectados entre si. (MEYER; EL-HANI, 2005)

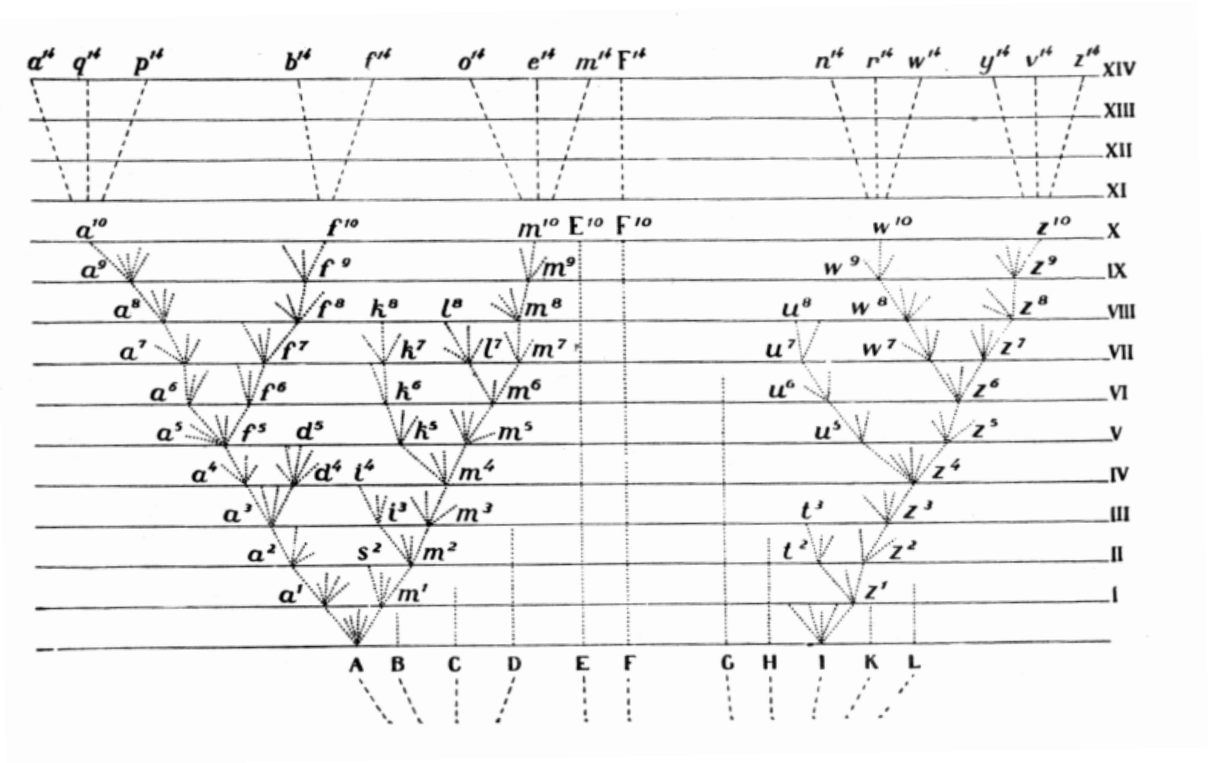


Fig. 2.1: Diagrama de gerações representando a árvore evolutiva desenhado por Darwin (1859)

A terceira teoria se refere ao fato de que a variação dentro de uma espécie origina diferença entre espécies, e oferece uma maneira de explicar de onde surgem as diferenças que levam a especiação (MEYER; EL-HANI, 2005). As diferenças observadas entre populações, espécies e táxons de maior hierarquia diferem em escala mas possuem a mesma natureza.

A quarta teoria se refere ao fato de que a evolução é gradual. As grandes mudanças evolutivas e diferenças entre espécies se dão pelo acúmulo gradual de mudanças menores. Esse ponto encontrou forte resistência a aceitação, principalmente devido às grandes lacunas que existem no mundo natural. Darwin (1859) contornava essas críticas argumentando que essas lacunas eram formadas por falhas no registro fóssil. Assim, os organismos parecem surgir abruptamente nos registros porque os fósseis de seus antepassados que continham os passos intermediários não foram preservados. (MEYER; EL-HANI, 2005)

A quinta e última teoria é também provavelmente a mais controversa: a seleção natural como principal mecanismo por trás da mudança evolutiva (MEYER; EL-HANI, 2005). A ideia da seleção natural foi mais revolucionária do que a da evolução em si, pois a ideia de evolução já estava no ar na época de Darwin (1859), mas foram ele e Wallace os primeiros a propor um mecanismo que controlasse seu funcionamento. Esse mecanismo não previa um curso definido para a evolução, mas sim um fortemente dependente das características locais do ambiente e do surgimento de variações

entre os indivíduos. Os processos que geravam, armazenavam e transmitiam essas varações ainda não eram conhecidos na época de Darwin, mas hoje são conhecidos a nível molecular com algum grau de detalhamento, além de como determinar as proporções de características discretas e contínuas transmitidas as novas gerações através das genéticas mendeliana e quantitativa. O mecanismo da seleção natural será explicado em detalhes a seguir na subseção 2.1.1. As explicações de conceitos básicos discutido nessa seção sobre evolução se basearam principalmente na obra "Evolução. 3ª edição" de autoria de Ridley (2006).

2.1.1 Seleção Natural, Deriva Gênica e Seleção Sexual

Seleção Natural

A seleção natural é o principal mecanismo subjacente a mudança evolutiva. Para compreender a sua natureza vamos tomar como mecanismo de descrição de dinâmica populacional as leis descritas por Turchin (2001). A primeira delas, a lei do crescimento exponencial, é enunciada como "uma população vai crescer (ou decrescer) exponencialmente enquanto o ambiente experimentado por todos os indivíduos da população permanecer constante" (TURCHIN, 2001). Essa lei representa um caso ideal onde o ambiente permanece constante, o que nunca pode ser considerado para grandes períodos de tempo, pois nenhum ambiente pode suportar um aumento populacional exponencial indefinidamente sem apresentar variações que afetem a taxa de nascimento e morte dos indivíduo (TURCHIN, 2001). Podemos observar isso tomando o exemplo clássico do elefantes, nas palavras de Darwin:

> De todos os animais conhecidos, o elefante, assim se julga, é o que se reproduz mais lentamente. Fiz alguns cálculos para avaliar qual seria provavelmente o valor mínimo do seu aumento em número. Pode, sem temor de errar, admitir-se que começa a reproduzir-se na idade de trinta anos, e que continua até aos noventa; neste intervalo, produz seis filhos, e vive por si mesmo até à idade de cem anos. Ora, admitindo estes números, em setecentos e quarenta ou setecentos e cinquenta anos, haveria dezenove milhões de elefantes vivos, todos descendentes do primeiro casal. (DARWIN, 1859)

De maneira análoga aos elefantes, todos os demais organismos cuja dinâmica populacional pode ser descrita pela primeira lei proposta por Turchin, caso pudessem se multiplicar indefinidamente, rapidamente cobririam a superfície da terra. Assim, a segunda lei proposta por Turchin se refere ao fato de que nenhum crescimento populacional pode continuar indefinidamente: "têm de haver um limite superior acima do qual a densidade populacional não pode aumentar"(TURCHIN, 2001). Essa limitação se dá porque o aumento da população provoca mudanças na interação entre o organismo e o ambiente em relação a diversos fatores, como escassez de recursos e competição intraespecífica. Para termos uma situação estacionária, para o caso de reprodução sexuada, cada casal deve produzir em média apenas um casal de descendentes reprodutivamente ativos, pois qualquer outra situação levaria a um crescimento (ou decrescimento) exponencial que seria insustentável a longo prazo.

Porém isso não significa que a prole de um casal será composta por apenas dois indivíduos, na verdade para a maioria dos seres vivos esse número é em média maior, sendo muito superior em algumas espécies, como é o caso do bacalhau do Ártico (*Gadus callarias*), no qual uma fêmea produz em média 2 milhões de ovos em um período reprodutivo. O que temos em geral na natureza é um "excesso de fecundidade", com fêmeas gerando uma prole maior do que o número de indivíduos que

sobrevive de fato até a idade reprodutiva. Há então em cada geração o que Darwin (1859) chamou de **luta pela sobrevivência**, uma disputa entre os seres vivos pela possibilidade de chegarem à idade adulta e se reproduzirem. Essa luta pela sobrevivência não deriva apenas da competição direta e indireta entre indivíduos de uma mesma espécie, mas também de uma complexa rede de interações ecológicas na qual cada organismo está inserido.

Em uma população qualquer, os organismos apresentam variação com relação a diferentes características que podem ser hereditárias, como forma, fisiologia e comportamento. Com essa luta pela sobrevivência os indivíduos com maior **aptidão**, ou seja, com as características que os tornam mais propensos a reprodução e sobrevivência, conseguem gerar um número maior de descendentes. Caso essas características sejam hereditárias temos então um aumento na proporção dos indivíduos que as possuem, gerando a mudança entre gerações que é como definimos a evolução. A esse princípio, onde uma variação é conservada e perpetuada na população caso seja útil na luta pela sobrevivência, Darwin (1859) deu o nome de **seleção natural**. A seleção natural conduz a população ao aumento da capacidade de sobrevivência e reprodução dos indivíduos no ambiente onde estão inseridos, a essa capacidade Darwin (1859) se referia como **adaptação**, que para ele era o problema central a ser resolvido por qualquer teoria evolutiva.

Também é possível compreender a seleção natural através de um argumento lógico, onde estabelecemos quatro condições que quando satisfeitas implicam na ocorrência de seleção natural (RIDLEY, 2006). Essas condições são:

1. Reprodução. As entidades devem se reproduzir para formar uma nova geração.

2. Hereditariedade. A progênie deve tender a lembrar os seus progenitores [...].

3. Variação nos caracteres individuais entre os membros da população [...].

4. Variação na aptidão do organismo com relação a um carácter herdável. [...] essa condição significa que um indivíduo da população com alguns caracteres deve ter uma maior probabilidade de reproduzir-se (i.e, ter um maior aptidão) do que outros. [...]

Quando as quatro condições são verificadas com relação a alguma característica de uma população, temos a ocorrência de evolução por seleção natural. Quando uma ou mais delas não é verificada, a seleção natural não ocorre. Podemos tomar como exemplo de entidade que não evolui por seleção natural os softwares maliciosos conhecido como *worms*. Os *worms* são softwares semelhantes aos vírus de computador, porém com uma capacidade de autorreplicação independente. Esse processo de autorreplicação pode ser considerado análogo a reprodução em seres vivos, portanto eles satisfazem a primeira condição. Eles possuem hereditariedade, pois as cópias geradas são iguais aos seus progenitores, cumprindo assim a segunda condição. Temos porém que não há variação alguma entre caracteres individuais em uma "população" de *worms* (ao menos para os tipos mais comuns destes softwares), pois as cópias são sempre absolutamente iguais, o que faz com que também não haja variação de aptidão entre os indivíduos. Portanto as condições três e quatro não são satisfeitas e podemos concluir que os *worms* não evoluem por seleção natural.

Com relação a vida, podemos verificar que ela cumpre essas quatro condições. A primeira e a segunda são as mais diretas, pois a maioria dos organismos se reproduz se puder e apresenta herança de algumas características, transmitidas através dos genes. Nem todas as características são transmitidas

por herança, porém aquelas que são podem ser moldadas pela seleção natural caso as condições 3 e 4 sejam satisfeitas.

A condição 3, como pode ser observado, é largamente satisfeita na natureza. Morfologicamente é possível ver que as populações apresentam variação individual para praticamente qualquer característica que possa ser medida, desde características de variação contínua, como tamanho corporal, até características de variação discreta, como o sexo. Também é possível encontrar variações a nível celular, como no número e estrutura dos cromossomos, variações a nível bioquímico, como no caso de populações que possuam proteínas que existem em diferentes formas, e variações a nível de DNA, onde proteínas que possuem a mesma forma apresentam variação com relação a sequencia de bases que as codificam. Além disso variação adicional é fornecida continuamente pelos processos de mutação e recombinação, porém preciso destacar que essa variação têm caráter aleatório, não possuindo assim qualquer tendencia a apresentarem melhorias evolutivas.

Apesar de as variações não serem necessariamente benéficas, algumas delas aumentam as probabilidades de sobrevivência e/ou reprodução, implicando no aumento da aptidão dos indivíduos que a apresentam com relação aos demais membros daquela população, o que supriria a condição 4. Assim inferimos que a seleção natural pode atuar sobre a vida.

Agora convém diferenciar as maneiras pelas quais a seleção pode agir sobre um carácter de variação contínua, como os que serão utilizados com relação ao canto dos Passeriformes nesse modelo. Podemos diferenciar três tipos de seleção a depender de onde se encontram os picos de aptidão, que são os valores assumidos por um fenótipo que maximizam a capacidade de reprodução e sobrevivência, com relação ao valor médio da distribuição fenotípica de uma população.

A seleção pode ser **direcional**, quando indivíduos portadores de uma característica de valor maior ou menor que a média possuem maior aptidão. A seleção direcional produz um deslocamento do valor médio assumido pela característica selecionada na direção de valores com maior aptidão.

A seleção pode ser **estabilizadora**, quando indivíduos portadores de uma característica de valor próximo a média possuem maior aptidão. A seleção estabilizadora age contra a mudança da distribuição fenotípica, fazendo com que a população se mantenha estável ao longo do tempo.

Por fim, a seleção pode ser **perturbadora**, quando qualquer valor distante da média, tanto os maiores quanto os menores, possuem uma maior aptidão. A seleção disruptiva gera uma distribuição bimodal ou polimodal, com picos nas regiões de maior aptidão.

Deriva Gênica

Existe também a possibilidade da característica analisada não influenciar na aptidão, como por exemplo uma população onde os indivíduos podem ter uma coloração mais clara ou mais escura em um ambiente onde coloração não tenha importância sobre as probabilidades de sobrevivência e reprodução. Nesse caso a seleção natural não irá atuar sobre essa característica indiferente com relação a aptidão, mas ainda poderá haver variação na frequência dos genótipos e consequentemente dos fenótipos observados pois os genes que compõe uma nova geração são amostras aleatórias da população anterior, e a essa variação aleatória da-se o nome de **deriva gênica**. É importante destacar que mesmo que uma característica seja vantajosa, ainda assim uma outra pode predominar na população devido ao efeito do acaso, principalmente se a contribuição dessa característica para a sobrevivência for pequena (MEYER; EL-HANI, 2005). O efeito da deriva é especialmente relevante quando se está trabalhando com populações menores.

Seleção Sexual

Existe um tipo específico de seleção natural que vamos abordar por ser o único tipo presente em nosso modelo, que é a **seleção sexual**. Através dela podemos explicar a existência e a formação de caracteres sexuais secundários, que são aqueles que diferenciam externamente machos e fêmeas e não estão ligados diretamente a cópula. Caracteres sexuais secundários costumam ser mais comuns no sexo masculino, pois a seleção sexual é muito mais poderosa em espécies poligâmicas, onde um macho pode se destacar dentre dos demais e acasalar com varias fêmeas. Muitos desses caracteres são deletérios, pois aparentemente diminuem a aptidão dos indivíduos que as possuem. Podemos tomar o exemplo clássico da cauda do pavão, ela é um órgão sexual secundário e apesar de o fato de ser deletério nunca ter sido explicitamente demonstrado, ela reduz a mobilidade do pavão, sua capacidade de voar e deve ter um custo energético associado a sua produção, o que significa que certamente é deletéria.

Existem dois tipos de seleção sexual. A **competição entre machos** é a mais simples dentre os dois, e se refere a competição dos machos pelo acesso as fêmeas. Ela justificaria a existência de certas características como a juba do leão e as esporas do galo de briga, que podem ser úteis diretamente na luta contra outros machos da mesma espécie (DARWIN, 1859).

Há porém características cuja existência não pode ser explicada pela competição entre machos, como é o caso do exemplo mencionado da cauda dos pavões. Ela não serve para lutar e nem possui qualquer outra função competitiva. Segundo Darwin a cauda do pavão existe apenas porque as fêmeas preferem acasalar com pavões que possuam caudas grandes e bonitas. A esse tipo de seleção sexual se dá o nome de **escolha pela fêmea**. Diferente do caso de seleção por competição entre machos, o motivo pelo qual uma fêmea prefere acasalar com um macho detentor de um carácter custoso não é tão claro e existem duas ideias sobre como isso acontece.

Na teoria de Fisher (1999), caso surja uma fêmea mutante na população que seja menos seletiva com relação a cauda do macho, ela acabaria por acasalar com machos que possuem caudas mais curtas e menos ostensivas e assim sua prole também teria caudas menos ostensivas. Dessa maneira a prole teria dificuldades de acasalar e propagar o gene que torna a fêmea menos seletiva, pois ele existe em uma frequência muito baixa na população. Fisher também discutiu como essa característica pode surgir inicialmente. Para ele a cauda, que hoje é um carácter deletério, pode ter sido vantajosa no início do desenvolvimento, assim não apenas os machos com caudas mas longas possuíam maior aptidão como as fêmeas que possuíam preferência por caudas mais longas também. As caudas mais longas nos machos e a preferência de cruzamento das fêmeas evoluíram num sistema de reforço mútuo, que Fisher denominou **escapamento** (*runaway*), até que as caudas ficaram tão grandes a ponto de ultrapassarem o pico de aptidão, como podemos ver na figura 2.2.

Já para Zahavi (1975), as características sexuais secundárias deletérias surgem como uma espécie de marcador para bons genes. Segundo ele existem genes que conferem maior aptidão em uma população, assim as fêmeas que acasalam com os machos que possuem esses bons genes geram uma prole que também possui maior aptidão. Assim caso exista um marcador que seja desvantajoso para o macho, apenas aqueles com os bons genes, e consequentemente maior aptidão, poderiam ostenta-lo e as fêmeas poderiam escolher corretamente com qual macho acasalar. O marcador necessariamente precisaria ser deletério para que os machos com genes ruins não acabem enganando as fêmeas, caso um macho apresente ao mesmo tempo o traço deletério e genes ruins eles não conseguem sobreviver e reproduzir devido ao acúmulo de desvantagens.

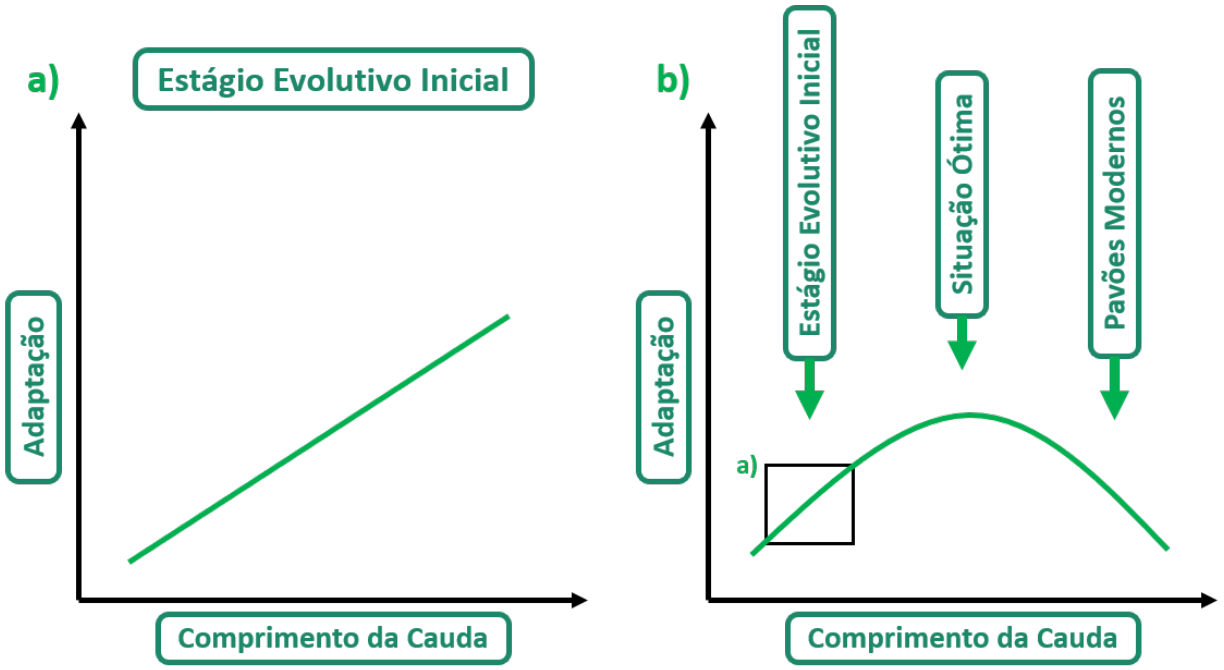


Fig. 2.2: Em (a) está representado o estágio evolutivo inicial do comprimento da cauda em pavões, onde um maior comprimento confere uma maior adaptação. Já em (b) a evolução fez o comprimento ultrapassar seu valor ótimo através do escapamento. (Figura adaptada de Ridley (2006))

2.1.2 Especiação

A especiação é o processo responsável pela formação de novas espécies, gerando ramificações na árvore evolutiva. O objetivo desse trabalho é avaliar como diferenças na aprendizagem entre Oscines e Suboscines impactam sobre sua dinâmica evoltiva, e por esse motivo nessa subseção vamos tratar sobre os diferentes tipos de especiação e analisar como ocorrem.

Para discutir especiação primeiramente é necessário definir o conceito de espécie. Existe uma discordância muito grande entre os biólogos sobre o que seria exatamente uma espécie, pois existem vários conceitos diferentes. Esses conceitos podem ser tanto verticais, quando tem como objetivo definir quais organismos pertencem a cada espécie o tempo todo, quanto horizontais, quando visam definir qual organismo presentem em cada espécie em um determinado momento. Vamos examinar 3 diferentes conceitos horizontais de espécie, o biológico, o ecológico e o fenético, que são aqueles que os biólogos estão normalmente mais interessados.

Vamos começar com o **conceito fenético** de espécie. Nele uma espécie é definida através de caracteres definidores específicos, então uma espécie seria basicamente um conjunto de indivíduos com caracteres semelhantes. Existem varias propostas que seguem esse conceito, como o "conceito tipológico de espécie", que define uma espécie como todos os indivíduos semelhantes a um espécime-tipo que deve ser disponibilizado em uma coleção pública, e o conceito filogenético de espécie, que é definido por Nixon e Wheeler (1990) como "a menor agregação de populações (sexuadas) ou de linhagens (assexuadas) que é diagnosticável através de uma combinação exclusiva de modos de expressão de caracteres, nos indivíduos comparáveis".

Há também o **conceito ecológico** de espécie, que surge porque espécies diferentes podem tolerar condições ecológicas diferentes e possuem diferentes papeis no sistema ecológico. Essa combinação de condições e do papel no sistema ecológico formam o **nicho** de uma espécie, e em geral espécies só conseguem se sobrepor no ambiente caso ocupem nichos diferentes. Assim o conceito ecológico de espécie define espécie com um conjunto de indivíduos que exploram o mesmo nicho.

Por fim temos o **conceito biológico** de espécie, que define espécies em termos de intercruzamento. Conforme a definição de Mayr et al. (1963) temos que "espécies são grupos de populações naturais que se intercruzam e estão reprodutivamente isoladas de outros grupos desse tipo". Segundo o Ridley (2006), hoje em dia esse é o conceito mais aceito, ao menos entre os zoólogos. Ele é importante para genética de populações, pois um grupo de indivíduos que só intercruza entre si configura um conjunto gênico, que é a unidade onde podem mudar as frequências genicas.

Um conceito de espécie muito parecido com o conceito biológico e que será usado nesse modelo é o **conceito de espécie por reconhecimento** de Paterson e McEvey (1993), que é citado por Ridley (2006). Segundo esse conceito uma espécie é "um conjunto de indivíduos que compartilham um sistema específico de reconhecimento para acasalamento (SMRS[1])", sendo esse sistema um método sensorial que permite reconhecer parceiros em potencial. Veremos que no modelo construído neste trabalho é fácil estabelecer esse sistema de reconhecimento específico para acasalamento, e como não modelamos um grande número de características, nem nichos ecológicos, o conceito de espécie por reconhecimento se apresenta bastante adequado a ser utilizado.

Agora vamos discutir o fenômeno da especiação em si, tomando como conceito de espécie o conceito biológico. Como nesse conceito uma espécie é definida como populações naturais que se intercruzam entre si e estão reprodutivamente isoladas, descrever como a especiação ocorre é descrever como ocorre o isolamento reprodutivo. Primeiramente precisamos distinguir entre isolamento reprodutivo pré e pós-zigótico. Isolamento pré-zigótico ocorre por diferenças nos comportamentos de cortejo, escolha de parceiros diferentes, por épocas de acasalamento diferentes ou por morfologias incompatíveis. Já no isolamento pós-zigótico o cruzamento ocorre, porém os híbridos possuem baixa fertilidade ou viabilidade. A distinção é importante porque diferentes teorias de especiação lidam com diferentes tipos de isolamento. Já que nosso modelo não possui diferença entre viabilidade ou fertilidade nos híbridos, nossos agentes não pode desenvolver isolamento reprodutivo pós-zigótico, o único que pode surgir em nosso modelo é o isolamento pré-zigótico, formado por grupos de agentes isolados devido a escolhas de parceiro diferentes.

Geograficamente podemos distinguir 3 tipos de especiação, conforma ilustrado na figura 2.3:

- **especiação alopátrica**: a nova espécie evolui geograficamente isolada da espécie ancestral;
- **especiação parapátrica**: a nova espécie evolui em uma população contígua;
- **especiação simpátrica**: a nova espécie evolui no mesmo espaço da espécie ancestral;

Na especiação alopátrica o isolamento reprodutivo pode surgir como um subproduto da divergência evolutiva entre populações isoladas geograficamente. Quando duas populações de uma mesma espécie são isoladas, o fluxo gênico entre elas é cortado e elas passam a evoluir de maneira independente, e assim podem fixar alelos diferentes, seja através do processo de deriva (que é aleatório), ou

[1]Em inglês, *specific mate recognition system, SMRS.*

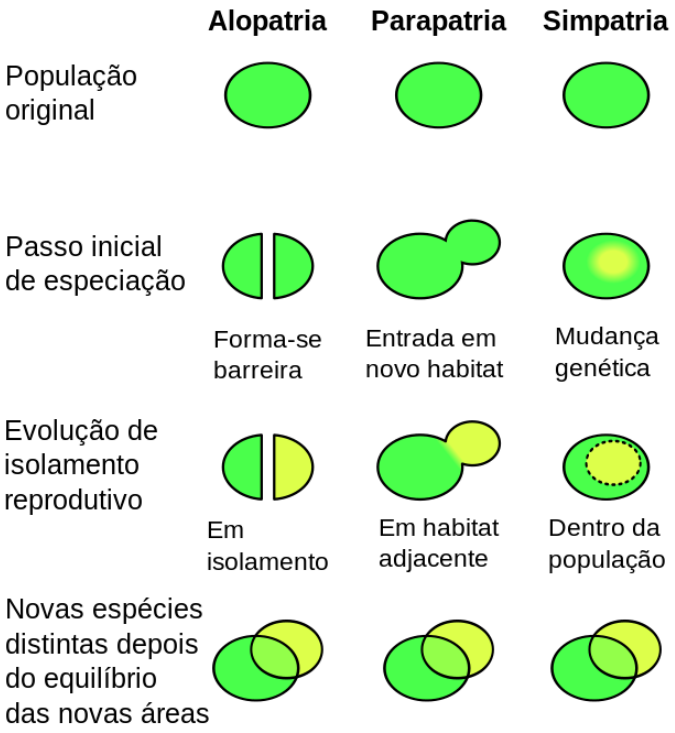


Fig. 2.3: Três principais tipos de especiação, distinguidos de acordo com as relações geográficas entre as populações envolvidas. (Modificada a partir de: GoEThe1 - Obra do próprio, Domínio público, https://commons.wikimedia.org/w/index.php?curid=3900860)

seja através da fixação de alelos que proporcionem maior aptidão aos diferentes ambientes ao qual cada população está submetida.

Um dos experimentos que corrobora a ideia do isolamento reprodutivo como subproduto de divergência evolutiva é o experimento de Dodd (1989), representado na figura 2.4. Ela dividiu moscas (*Drosophila pseudoobscura*) capturadas em uma mesma localização em oito populações, e colocou 4 delas em um meio nutritivo a base de amido e as outras 4 em um meio nutritivo a base de maltose. As populações foram cultivadas nesses meios diferentes por varias gerações, até que se surgissem diferenças detectáveis em suas enzimas digestivas, fruto da adaptação aos dois meios diferentes. Em seguida, após marcar machos de fêmeas de cada população, ela colocou indivíduos de diferentes populações dentro de caixas para acasalar e notou que moscas criadas no meio a base de amido preferiam acasalar com outras também criadas a base de amido e moscas criadas no meio a base de maltose preferiam acasalar com moscas também criadas a base de maltose. Assim, o experimento de Dodd (1989) detectou um isolamento reprodutivo pré-zigótico parcial que evoluiu como subproduto em uma situação de alopatria. O isolamento não foi selecionado diretamente, e sim as mudanças que ocorreram durante a adaptação das enzimas digestivas levaram a ele.

Existem dois mecanismos que podem contribuir com o surgimento do isolamento reprodutivo como um subproduto. Um deles é o **efeito carona**, que acontece quando a seleção favorece um gene de um loco, e assim acaba alterando a frequência de genes em locos ligados a ele. No caso do experimento de Dodd (1989), o que pode ter acontecido é que os genes que codificavam as enzimas

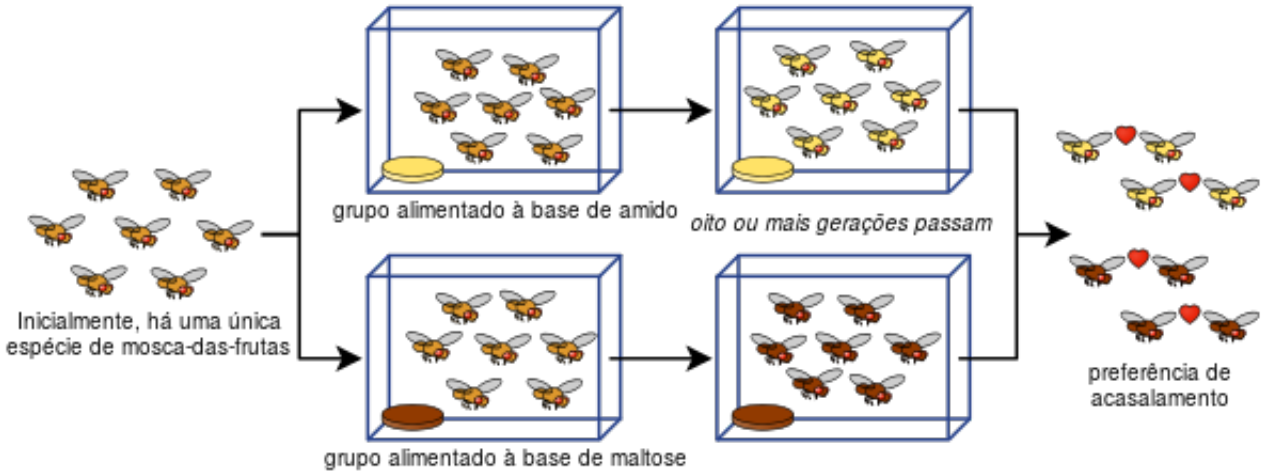


Fig. 2.4: Representação do experimento de Dodd (1989). (Por Fastfission (traduzido para o português por Spoladore) - Imagem:Drosophila speciation experiment.svg, Domínio público, https://commons.wikimedia.org/w/index.php?curid=4858676)

digestivas eram ligados a genes que influenciavam o cortejo, assim ao aumentar a frequência do gene que codificava uma enzima digestiva específica o cortejo pode ter sido influenciado por efeito carona. O outro mecanismo é a **pleiotropia**, que acontece quando um gene influencia outras características além da que está sendo selecionada. Como exemplo temos as aves, cujo tamanho do bico está relacionado com o tamanho do alimento que consomem, mas que também está relacionado com os cantos que elas são capazes de produzir, que por sua vez estão relacionados com a escolha de parceiros, como veremos na seção 2.2.3. Populações que evoluam em ambientes com disponibilidade de alimentos de tamanhos diferentes podem desenvolver dialetos que como subproduto da evolução de bicos adaptados aos alimentos de sua região, e assim apresentarem isolamento reprodutivo pré-zigótico.

Além de como subproduto da divergência evolutiva, o isolamento pode ser favorecido diretamente por um mecanismo chamado **reforço** (*feedback*). O reforço atua quando há duas formas genéticas em uma população e a forma hibrida entre as duas possui um valor adaptativo menor que as formas puras. Teoricamente a seleção natural beneficiaria os indivíduos que cruzassem apenas com outros que possuem a mesma forma genética, levando assim a população a um isolamento pré-zigótico. A teoria do reforço contudo tem sofrido criticas, pois as condições que levam a ela (duas formas genéticas cujo cruzamento seja desvantajoso) não parecem estáveis a longo prazo na maioria das situações e os testes empíricos tem se mostrado inconclusivos.

A especiação parapátrica está intimamente relacionada com o reforço Na especiação parapátrica as populações não estariam isoladas, mas assumiriam um padrão de "clina escalonada", que poderia ser gerado por uma variação brusca no ambiente por exemplo. Caso os lados da clina possuam formas que possam ser claramente diferenciadas temos o que se chama de **zona híbrida**, e caso essa zona híbrida seja composta por híbridos que são evolutivamente desvantajosos temos o que se chama de **zona de tensão**. Em uma zona de tensão as condições onde o reforço poderia atuar, levando ao isolamento pré-zigótico e dele para a especiação. Contudo não esperamos encontrar especiação parapátrica em nosso modelo, pois o isolamento pós-zigótico, que é um do pré requisitos para o

reforço, não é comportado por ele.

A especiação simpátrica também pode acontecer, porém da mesma maneira que a especiação parapátrica ela depende do mecanismo de reforço. Enquanto na especiação parapátrica a população possui um polimorfismo espacial, na especiação simpátrica as diferentes variedades da espécie se sobrepõe no espaço. Caso os híbridos tenham valor adaptativo mais baixo do que as formas puras, novamente temos uma situação onde o reforço pode atuar.

Por fim, a seleção sexual, que será o único mecanismo de seleção em nosso modelo, também é importante para a especiação. Uma das situações onde podemos verificar a ocorrência de seleção sexual é quando há escolha de parceiros pela fêmea, como vimos na subseção 2.1.1. Os mecanismos que as fêmeas usam para selecionar os machos podem contribuir ou determinar o isolamento pré-zigótico. Podemos entender melhor como isso ocorre tomando como exemplo o experimento de Dodd (1989). Nas diferentes populações, a seleção favorece os indivíduos que estão adaptados a viver com uma dieta a base de amido (ou de maltose), mas também favorece as fêmeas que escolhem machos mais bem adaptados a viverem nessas condições. Assim as fêmeas que vivem na população mantida a base de amido podem ter sido selecionada para escolher machos que estejam melhor adaptados a viver a base de amido, e o mesmo acontece com as fêmeas que vivem na população mantida a base de maltose.

2.2 Canto dos Passeriformes

2.2.1 Introdução

Nessa seção iremos abordar a fenomenologia do sistema que nosso modelo se propõe a representar, que é o canto dos Passeriformes, como ele se desenvolve, suas funções ecológicas e seu impacto evolutivo. Para isso nos basearemos principalmente na obra "Bird Song - Biological Themes and Variation" de autoria de Catchpole e Slater (2008). Começaremos com uma introdução básica e estabelecimento da terminologia utilizada no estudo do canto dos pássaros.

Os cantos emitidos pelos Passeriformes são estruturas especiais usadas unicamente para a comunicação, o que chamamos de **sinais**. Podemos saber que houve comunicação entre dois animais quando o sinal enviado por um modifica o comportamento do animal que o recebeu (SLATER, 1983). Essa é uma definição restrita de comunicação, que exclui a possibilidade de detecção passiva de um sinal, mas é útil para numerosos experimentos, que ao lado da observação, gravação e analise, são usados para estudar o canto dos pássaros. Um tipo de experimento muito relevante é o feito a partir de playbacks, que consiste em tocar sons para os pássaros e observar se eles reagem e como reagem. Esses sons podem ser cantos e chamados gravados na natureza ou sons sintéticos.

Os Passeriformes possuem dois tipos de vocalizações, os cantos e os chamados. Os cantos são definidos por Catchpole e Slater (2008) como "tendem a ser vocalizações longas e complexas produzidas pelos machos na temporada de acasalamento", e tem duas funções principais, **atrair a fêmea para o acasalamento** e **defender o território**. Já os chamados são vocalizações mais curtas, produzidas por ambos os sexos ao longo do ano com diversos tipos de funções específicas, como alertar para perigos. Em algumas espécies existe uma superposição entre os cantos mais simples e os chamados mais complexos, o que enfatiza o carácter arbitrário dessa divisão, mas mesmo nessas situações a distinção pode se fazer útil. As vocalizações também podem ter outras funcionalidades, como iden-

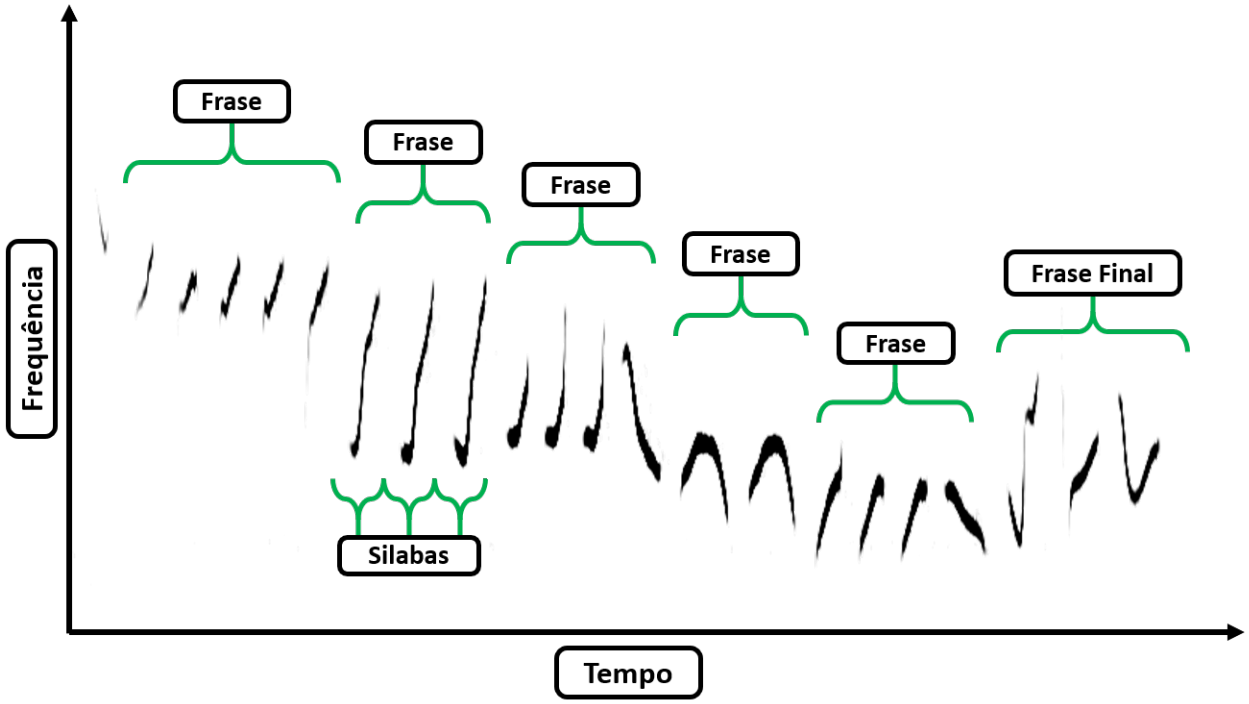


Fig. 2.5: Sonograma do canto de uma Felosa-Musical (*Phylloscopus trochilus*). O canto está marcado para representar os conceitos de sílaba, frase e frase final. (Modificada a partir de: Mysid - Feito pelo próprio em baudline e Gimp de Imagem:Phylloscopus rochilus.ogg., Domínio Público, https://commons.wikimedia.org/w/index.php?curid=2129188)

tificar sua própria espécie, vizinhos, e indivíduos específicos, como os filhotes ou o parceiro no caso de espécies monogâmicas.

Dentre as funções citadas o papel do canto no acasalamento terá destaque nesse trabalho. Ele é fundamental, tanto para atrair a fêmea, como foi citado, quanto para servir como parâmetro das fêmeas para a escolha do macho. Dessa maneira o canto desemprenha um papel central na determinação do acasalamento ou não de dois indivíduos, e serve como sistema específico de reconhecimento para acasalamento, conforme o conceito de espécie de Paterson e McEvey (1993), mencionado na subseção 2.1.2. Vamos ver melhor como a escolha da fêmea funciona na subseção 2.2.3.

É necessário definir também outros termos que serão usados posteriormente. A maioria dos Passeriformes tem mais de um canto da espécie, e alguns tem muitos. Para o conjunto de cantos de um indivíduo damos o nome de **repertório**. Além disso podemos dividir cada canto em unidades menores, denominadas **frases**, **sílabas** e **elementos**, que estão representadas no sonograma da figura 2.5. As frases são seções distintas de um canto formadas por uma série de unidades que ocorrem em algum padrão particular. Essas unidades são chamadas de sílabas, que podem ser simples ou complexas. Caso sejam complexas, podemos dividi-las em unidades menores denominadas elementos.

Mas porque usar o som e a audição ao invés de outros meios e canais de comunicação? O motivo é que os sons apresentam muitas vantagens interessantes. Os Passeriformes não possuem olfato bem desenvolvido, o que torna a audição e a visão seus sentidos mais importantes. Com relação a visão, o som apresenta varias vantagens. Continua funcionando perfeitamente a noite e em situações de

baixa luminosidade em geral. Além disso o som possui uma difração muito mais expressiva que a luz, permitindo que as ondas contornem obstáculos e que a se propague por grandes distâncias em situações pouco favoráveis para a visão, como em uma floresta densa. Além disso o som é uma forma de comunicação rápida e transiente, ou seja, ele só é produzido quando se quer produzir e leva uma grande quantidade de informação rapidamente através de grandes distâncias.

A ordem dos Passeriformes é formada por duas subordens, os Oscines e os Suboscines. A principal distinção entre as duas subordens se dá com relação a aprendizagem: ambas apresentam aprendizagem individual, mas apenas os Oscines apresentam aprendizagem socialmente mediada, o que será explicado na subseção 2.2.2. Já os estudos com Suboscines indicam que o desenvolvimento do seu canto sofre pouco efeito de aprendizagem, como no experimento de Kroodsma (1984), onde ele criou filhotes de duas espécies de Suboscines, papa-moscas de salgueiro (*Empidonax traillii*) e de papa-moscas de amieiro (*Empidonax alnorum*), colocando-os para ouvir playbacks com cantos uma da outra. Em seu experimento, Kroodsma encontrou que ambas desenvolveram os cantos característicos de suas próprias espécies, um sinal de que eles não apresentam aprendizagem socialmente mediada. Além disso os Suboscines possuem uma maior discriminação com relação aos cantos que pertencem a própria espécie e uma menor plasticidade com relação aos cantos que são capazes de executar (FREEMAN; MONTGOMERY; SCHLUTER, 2017).

2.2.2 O Desenvolvimento do Canto

Os cantos não surgem prontos nos Passeriformes adultos, eles são fruto de um processo de desenvolvimento que nos Oscines está atrelado a aprendizagem socialmente mediada. Pra descrever esse processo vamos analisar alguns experimentos sobre desenvolvimento do canto em tentilhões (*Fringilla coelebs*). Em seguida, vamos destrinchar um modelo de aprendizado do canto feito com base nas observação de diversas espécies de Oscines. A generalização do modelo de aprendizado pode parecer excessiva, mas devido a imensa variação que existe entre as espécies de Oscines ela pode servir como um bom ponto de partida para nosso trabalho.

Em um experimento realizado por Thorpe (1958), ele comparou os sonogramas dos cantos de tentilhões selvagens com os de tentilhões criados isoladamente, que cresceram sem a oportunidade de escutar outros cantos. Os tentilhões selvagens desenvolveram cantos semelhantes ao padrão da espécie, que é um trinado composto por algumas frases, cada uma delas formada sílabas que se repetem com um formato constante, e uma frase final. Já os cantos desenvolvidos pelos tentilhões criados isoladamente, apesar de apresentarem aproximadamente a mesma frequência, duração, e divisões em silabas e elementos os, apresentavam silabas muito irregulares uma em relação a outra, tornando difícil separar o canto em frases. Além disso a silaba final era muito diferente entre os selvagens e os isolados.

A diferença entre os sonogramas mostra que escutar o canto dos adultos é importante para que tentilhões jovens desenvolvam seu próprio canto. Thorpe (1958) também realizou diversos outros experimentos e constatou alguns fenômenos interessantes:

- Tentilhões jovens que escutam playbacks de cantos de adultos em diferentes estágios do primeiro ano de vida desenvolvem o canto normal da espécie;
- Tentilhões jovens criados em grupo sem adultos desenvolvem cantos mais simples, assim como os criados isolados, porém similares aos dos colegas de grupo;

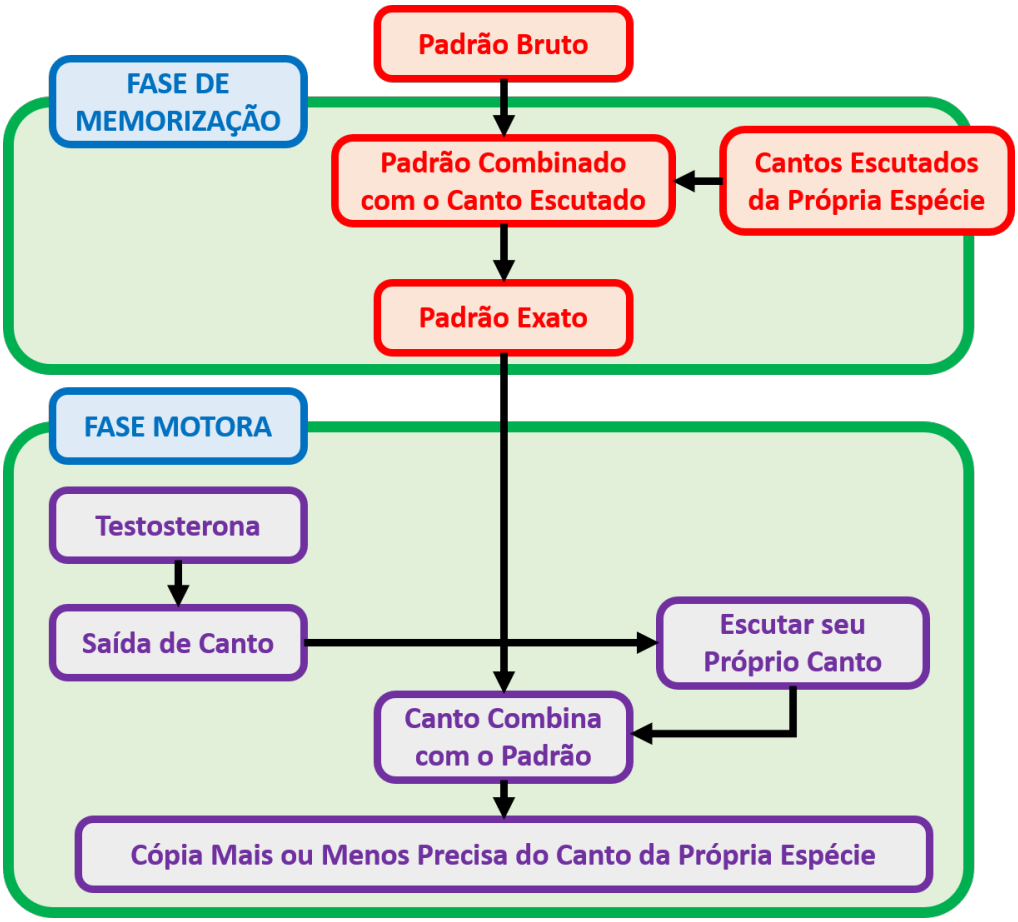


Fig. 2.6: Modelo do padrão auditivo do desenvolvimento do canto. (Figura adaptada de Catchpole e Slater (2008))

- Tentilhões jovens não aprendem qualquer coisa que escutam, como cantos de outras espécies e melodias tocadas com um apito, a menos que o som seja muito similar ao canto dos tentilhões, como é o caso do canto da petinha-das-árvores (*Anthus trivialis*);
- Tentilhões jovens desenvolvem cantos condizentes com a da sua espécie mesmo que as escutem muitos meses antes de começarem a cantar;
- Uma vez que o canto está plenamente desenvolvido (o que chamamos de **cristalizado**) os tentilhões não o modificam mais, mesmo que ele seja um canto subdesenvolvido;

Outro experimento interessante sobre o desenvolvimento do canto em tentilhões, útil para elucidar como ocorre a cristalização, foi feito por Nottebohm (1969). Ele castrou um tentilhão jovem, o que fez com que este não desenvolvesse seu canto no tempo habitual. O canto só se desenvolveu quando Nottebohm ministrou nele hormônio sexual masculino, o que indicava que os níveis de testosterona no sangue tinham forte relação com o desenvolvimento do canto.

Apesar da grande variabilidade da subordem dos Oscines dificultarem uma generalização, muitos dessas observações foram também confirmadas para outras espécies. Com elas foi criado um **modelo**

do padrão auditivo para o desenvolvimento do canto dos Oscines, que está representado na figura 2.6 e servirá de base para a criação do nosso modelo. No modelo do padrão auditivo os pássaros jovens nascem com um 'padrão bruto' de como deve ser o canto da espécie e o canto passa por um processo de desenvolvimento do canto composto por duas fazes, a fase de memorização e a fase motora.

Na fase de memorização os Oscines jovens escutam todo tipo de sons. O padrão bruto da espécie serve então como um filtro, selecionando o que deve ou não ser aprendido, impedindo que os jovens Oscines aprendam qualquer som escutado. Os sons que combinam com o padrão bruto são então assimilados por eles, e contribuem para a formação do padrão exato de como é o canto da espécie. Isso justifica que tentilhões criados isoladamente desenvolvam cantos similares aos de sua espécie com relação a algumas características, mas muito diferentes em relação a outras: eles possuem apenas o padrão bruto de como deveriam cantar, pois não passaram pelo aprendizado social que refina esse padrão.

Já a fase motora começa para a maioria das espécies na primavera, quando o macho começa a produzir testosterona, que como vimos no experimento de Nottebohm (1969) induz o pássaro a cantar. De inicio o canto dele não se parece tanto com o do padrão exato ele aprendeu, porém ele vai escutando e ajustando o próprio canto até que ele se torne parecido com esse padrão, quando se cristaliza.

2.2.3 Seleção Sexual no Canto dos Passeriformes

O canto é uma característica sexual secundária presente majoritariamente nos Passeriformes machos, o que sugere que a evolução do canto é controlada por um mecanismo de seleção sexual. Iremos apresentar alguns experimentos que mostram que essa seleção de fato acontece e em seguida apontaremos como as fêmeas podem ser beneficiadas direta ou indiretamente por essa escolha.

Vamos nos restringir a experimentos que mostram duas coisas importantes: que as fêmeas são atraídas pelo canto dos macho e que elas escutam vários machos antes de escolher o seu favorito, ao invés de ficar com o primeiro que encontram.

O experimento de Eriksson e Wallin (1986), feito com duas espécies de papa-moscas (*Ficedula hypoleuca* e *F. albicollis*), fornece evidência de que as fêmeas são atraídas pelo canto dos machos. Nesse experimento os papa-moscas se reproduziam em caixas-ninho. Eriksson e Wallin instalaram redes de captura automática na entrada dessas caixas, para capturar fêmeas que viessem inspecionalas, colocaram bonecos imitando o macho da espécie e playbacks do canto das espécies para tocar dentro de metade delas. Nove de dez fêmeas capturadas foram atraídas para caixas com playbacks, o que indica que o canto desempenha um papel na atração das fêmeas para o acasalamento. Resultados semelhantes foram encontrados para outras espécies de Passeriformes.

Para avaliar se o mecanismo de seleção sexual por escolha da fêmea está por trás da dinâmica evolutiva do canto é preciso verificar se as fêmeas escolhem ativamente os seus machos, ou se são apenas atraídas pelo primeiro canto que escutam. Para isso iremos citar o experimento de Bensch e Hasselquist (1992), no qual eles usaram transmissores de rádio para rastrear fêmeas de rouxinol-grande-dos-caniços (*Acrocephalus arundinaceus*). Eles descobriram que as fêmeas dessa espécie visitam em média seis machos antes de retornar para acasalar com um macho em particular, o que indica uma escolha ativa por parte da fêmea. Resultados semelhantes também foram encontrados para outras espécies de Passeriformes.

Temos ainda resultados que corroboram a existência nas fêmeas, assim como nos machos, de um

padrão bruto do canto da espécie, conforme reportado no modelo do padrão auditivo da figura 2.6, que é responsável por servir de parâmetro para que as fêmeas determinem se o canto que elas estão escutando pertence a sua própria espécie (WHALING et al., 1997; NELSON, 2000). Contudo, dentre os cantos reconhecidos como pertencentes a própria espécie aparentemente não há preferências de um sobre o outro (LYNCH, 1996).

Identificar bem quais cantos pertencem a sua espécie é muito importante para as fêmeas do que para os machos. Caso um macho cometa um erro de avaliação de um canto, ele terá apenas que gastar parte do seu tempo e energia para lidar com um problema, como um confronto causado pela invasão do território de outro macho. As fêmeas, por outro lado, tem muito mais a perder, pois um julgamento errado sobre a espécie pode comprometer o potencial reprodutivo daquela fêmea em uma temporada de acasalamento.

Além disso a quantidade de canto pode servir como um indicador da qualidade do macho ou do terreno que ele defende. O experimento feito por Wasserman e Cigliano (1991) mostra que as fêmeas preferem machos que cantem mais (seja por cantarem mais frequentemente ou por produzirem cantos mais longos). Eles manipularam playbacks implementando esses dois tipos de modificações, e em ambos os casos as fêmeas preferiam os playbacks alterados com maiores quantidade de canto.

Aumentar a frequência com que se canta é custoso para os machos, tanto do ponto de vista energético quanto por aumentar o risco de exposição a predadores, o canto é uma característica deletéria. Conforme a teoria de Zahavi (1975) para a seleção sexual, que vimos na subseção 2.1.1, características deletérias podem servir como marcadores para bons genes, pois apenas os machos com maiores valores de aptidão conseguirão ostenta-las. Ao escolher machos que cantam muito, as fêmeas podem estar indiretamente escolhendo bons genes e por esse motivo a quantidade de canto é uma característica que as fêmeas deveriam levar em conta na escolha do macho.

Outro custo associado a quantidade de canto é menos direto. Os pássaros precisam dedicar tempo a outras atividade além do canto, como ao forrageio e a alimentação, e quanto mais tempo se dedica a essas atividades, menos tempo eles podem despender com o canto. Assim, machos que conseguem defender um território melhor em termos de recursos podem cantar mais, pois precisam dedicar menos tempo em média ao forrageio. Manter uma alta frequência de canto poderia então ser um indicativo de que o macho possui um bom território. O experimento de Radesäter et al. (1987), feito com toutinegras de salgueiro (*Phylloscopus trochilus*), parece confirmar que existe uma correlação entre a quantidade de comida disponível e a média de canto por minuto.

2.3 Conclusão

Na seção 2.1 apresentamos o que é a teoria da evolução e diversos conceitos básicos relacionados a ela que foram necessários para a construção e análise do modelo. Ao introduzirmos o conceito de seleção natural, definimos aptidão e adaptação e diferenciamos seleção direcional, estabilizadora e perturbadora. Apresentamos os conceitos de deriva gênica e seleção sexual, que são fundamentais em nosso trabalho, pois são as únicas forças por trás das mudanças evolutivas no nosso modelo. Em seguida definimos espécie usando diferentes conceitos e determinamos qual deles seria utilizado nesse trabalho, o de espécie por reconhecimento para acasalamento de Paterson e McEvey (1993). Como este trabalho trata da dinâmica evolutiva de Passeriformes, tratamos do fenômeno de especiação descrevendo mecanismos que possibilitam o desenvolvimento de isolamento reprodutivo em situações

de alopatria, parapatria e simpatria, e destacamos que nosso modelo apenas comporta isolamento pré-zigótico.

Na seção 2.2 explicamos o que são cantos, quem os produz, suas funções, seus componentes, suas vantagens frente a outras formas de comunicação e as diferenças entre eles e os chamados, que são outros tipos de vocalizações dos Passeriformes. Diferenciamos Oscines e Suboscines, ambas subordens dos Passeriformes, através da existência de aprendizagem socialmente mediada nos Oscines e da maior discriminação e menor plasticidade no canto nos Suboscines. Apresentamos um modelo do padrão auditivo, utilizado na construção do nosso próprio modelo, que fornece uma descrição generalizada de como ocorre o desenvolvimento do canto nos Oscines. Mostramos experimentos que corroboram que as fêmeas são atraídas pelos cantos dos machos, e que escutam vários machos antes de escolher um parceiro.

Capítulo 3

Materiais e Métodos

Sistemas evolutivos são em geral sistemas complexos. Não existe uma definição universal de sistema complexo, porém os autores que abordam o tema parecem concordar que eles compartilham algumas propriedades entre si. Segundo Boccara (2010) elas são:

1. Eles são constituídos de um grande número de agentes interagentes.

2. Eles exibem emergência; isto é, um comportamento coletivo auto-organizado difícil de prever a partir do conhecimento do comportamento individual dos agentes.

3. Seu comportamento emergente não resulta da existência de um controlador central.

Podemos checar que esses elementos estão presentes em nosso sistema evolutivo. O grande número de agentes interagentes é dado pela população de Passeriformes virtuais, que interagem uns com os outros. Existem duas propriedades emergentes em nosso sistema, a nível social temos a formação de dialetos, que definimos como cantos similares entre si que são reconhecidos proporcionalmente por um número maior de pássaros dentro de uma comunidade de agentes que fora dela, e a formação de novas espécies (aqui usando o critério de espécie por reconhecimento), que se dá quando comunidades de Passeriformes desenvolvem dialetos tão diferentes que os membros de uma não reconhecem mais nenhum dos membros de outra como pertencentes a mesma espécie. Além disso não existe nenhum tipo de controlador central coordenando a formação de dialetos ou de novas espécies, esses comportamentos surgem de forma espontânea, através das interações de curto alcance entre os Passeriformes virtuais.

Para a analise do nosso sistema usaremos um modelo computacional baseado em agentes. Em linhas gerais, um modelo é uma representação simplificada do mundo real. Pesquisadores frequentemente desejam estudar sistemas reais para os quais uma analise direta seria muito lenta, muito complexa ou impossível, como geralmente é o caso de sistemas biológicos. Usando um modelo matemático ou computacional, um sistema pode ser simplificado e então analisado matematicamente ou através de experimentos *in silico*.

Tornar um sistema mais simples envolve ignorar alguns dos componentes e processos que o compõe. Porém como decidir quais aspectos devem ser incluídos e quais devem ser ignorados? Essa decisão está diretamente ligada à pergunta que queremos responder. Existem diferentes abordagens, e para entender melhor a modelagem baseada em agentes, que iremos usar, vamos apresentar as

diferenças entre duas concepções gerais de modelagem, a abordagem reducionista newtoniana (normalmente usada na física) e a abordagem usada para tratar sistemas complexos.

Segundo Janssen (2012) uma das principais diferenças está no fato de que a abordagem reducionista newtoniana é mecanicista, ou seja, em posse do modelo correto do sistema é possível fazer uma previsão sobre a evolução dele. Já a abordagem de sistemas complexos assume que a trajetória completa do sistema geralmente não pode ser determinada, ao invés disso o sistema se comporta se ajustando continuamente a novos contextos.

Os modelos reducionistas dão conta de resolver algumas classes de problemas, como o movimento de corpos em sistemas com poucos graus de liberdade, e por isso essa abordagem foi predominante durante muito tempo nas ciências, começando pela física e passando em seguida para as ciências biológicas, sociais e econômicas. Porém existem aspectos de descrições reducionistas que as tornam inadequadas para responderem determinadas perguntas. Na física, por exemplo, temos a mecânica newtoniana, o estudo do movimento dos corpos, que é reducionista e completamente **reversível**, ou seja, sistemas descritos usando esse formalismo continuam obedecendo a mesma dinâmica caso o tempo seja revertido. Já na termodinâmica, o estudo da relação entre fenômenos térmicos e caloríficos, a segunda lei quebra a noção de reversibilidade para sistemas macroscópicos fechados, pois neles a entropia sempre cresce ou se mantém constante. Isso tem diversas implicações, como a impossibilidade do calor fluir de uma fonte para uma outra de temperatura maior por um processo reversível. Assim, por mais que a mecânica newtoniana consiga descrever o movimento de todas as partículas de um gás, ela não consegue dar conta da irreversibilidade que emerge em sistemas macroscópicos. Além disso a conexão entre modelo e fenômeno nesses sistemas é comprometida pela impossibilidade experimental de determinar com precisão o estado inicial a partir do qual ele evolui. Então para fazer uma conexão entre a descrição microscópica do movimento das partículas (seja clássica ou quântica) e as propriedades termodinâmicas de um gás é preciso usar mecânica estatística.

A física estatística é uma abordagem de modelagem *botton-up* não reducionista onde a descrição microscópica do sistema e a estatística se juntam para criar um modelo que descreve um todo que é maior que a soma das partes. Segundo Salinas (2013), uma análise mecânico-estatística de um sistema físico pode ser resumida nas seguintes etapas:

1. Especificação dos estados microscópicos do sistema (que formam um conjunto denominado *ensamble* estatístico);

2. Estabelecimento de um postulado estatístico básico e utilização da teoria de probabilidades. No caso de um sistema com energia total fixa, utilizamos a hipótese simplificadora das probabilidades iguais a priori, que conduz à identificação do *ensamble* microcanônico;

3. Estabelecimento de uma conexão com a termodinâmica, ou seja, com as variáveis mensuráveis direta ou indiretamente do mundo macroscópico;

De uma maneira geral, o desenvolvimento de modelos baseados em agentes para sistemas complexos segue etapas semelhantes às de uma análise mecânico-estatística. A especificação dos estados microscópicos do sistema da etapa 1 depende da descrição de seus componentes individuais. Na análise mecânico estatística, no caso de um gás, usamos mecânica clássica ou a mecânica quântica para especificar os microestados, que tanto descrevem o comportamento de uma partícula individual, quanto servem de base para hipóteses geralmente simplificadas de interação. Na modelagem baseada

em agentes, os componentes individuas do modelo são os próprios agentes, que possuem comportamentos e interações que buscam mimetizar os do sistema real.

Na etapa 2 da análise mecânico estatística a teoria de probabilidades é utilizada para fazer a ponte matemática entre o comportamento microscópico e o macroscópico do sistema. Já os modelos baseado em agentes são computacionais, e por esse motivo essa ponte é construída através da realização de experimentos *in silico*.

E por fim na etapa 3 da análise mecânico-estatística, temos o estudo de propriedades emergentes feito através do estabelecimento de uma conexão com a termodinâmica. De maneira semelhante, os modelos baseados em agentes, como veremos na seção 3.1.2, tem como um de seus benefícios a capacidade de capturar fenômenos emergentes que surgem através interação entre as entidades individuais que os compõem.

A seção 3.1 aborda as principais questões relacionadas a modelagem baseada em agentes: o que é um agente, sua estrutura interna e a estrutura do modelo baseado em agentes como um todo. Ja a seção 3.2 aborda o assunto de redes complexas, que serão utilizadas no tratamento dos dados gerados pelo nosso modelo.

3.1 Modelagem Baseada em Agentes

3.1.1 O que é um Agente?

Todos nós estamos de alguma maneira familiarizados com o conceito de agentes. Grimm et al. (2005) traça um paralelo bem intuitivo sobre entre o conceito de agente autônomo e os agentes secretos, com James Bond. Segundo ele esses dois tipos de agente apresentam algumas características em comum: ambos possuem um objetivo claro, possuem autonomia sobre quais decisões tomar para atingir esse objetivo e adaptam essas decisões a contextos em contante mudança.

O conceito de agentes surgiu inicialmente na ciência da computação, mais especificamente no ramo de inteligência artificial, onde desempenha um papel central. Apesar disso, não houve muita consideração sobre a síntese do conceito de agentes até a década de 80, quando surgiu um grande interesse no assunto em toda ciência da computação e em áreas afins, como a robótica.(WOOLDRIDGE; JENNINGS, 1995)

Assim como para sistemas complexos, não há uma definição universal e precisa do que é um agente. Ao invés disso diferentes autores aparentemente cunharam suas próprias definições a partir dos exemplos de agentes que conheciam (FRANKLIN; GRAESSER, 1996). Para Wooldridge e Jennings (1995), a pergunta **o que é um agente?** encontra na comunidade de computação baseada em agentes o mesmo problema que a pergunta **o que é inteligência?** encontra na comunidade de inteligência artificial, que é o fato de que muitas áreas estritamente correlacionadas fazem livre uso do termo, o que torna difícil produzir uma definição universalmente aceita. Isso não deveria ser necessariamente um problema, porém existe o perigo de abuso e mal uso do termo. Assim uma definição mais precisa é necessária.

Para propor uma definição geral, Franklin e Graesser (1996) apresentam uma revisão de diversas definições encontradas na literatura, mostrando convergências e divergências entre elas. A partir dessas definições eles elaboraram uma lista das propriedades que normalmente são relacionadas aos agentes, listadas na tabela 3.1. Eles propõem então uma definição de agente autônomo que contém

Propriedades	Outros Nomes	Definição
Reatividade	(Detecção e Atuação)	responde em tempo hábil a mudanças no ambiente
Autonomia		exerce controle sobre suas próprias ações
Orientação a Objetivo	Propósito Pró-ativo	não simplesmente age em resposta ao ambiente
Continuidade Temporal		é um processo em contínua execução
Comunicatividade	Aptidão Social	se comunica com outros agentes, talvez inclusive pessoas
Aprendizagem	Adaptação	muda seu comportamento com base em experiência prévia
Mobilidade		capacidade de se transportar no ambiente onde está inserido
Flexibilidade		ações não roteirizadas
Personificação		personalidade "verossímil"e estado emocional

Tab. 3.1: Propriedades possivelmente presentes em um agente. As quatro primeiras estão inclusas na definição de agente autônomo. As demais são facultativas. (Tabela adaptada de Franklin e Graesser (1996))

as quatro primeiras propriedades da tabela 3.1, assumindo que as demais propriedades são opcionais. Além disso eles também propões uma taxonomia para a classificação dos agentes. A definição de agente autônomo de Franklin e Graesser (1996) é:

> Um **agente autônomo** é um sistema que está situado dentro de um ambiente e é parte dele, capaz de sentir o ambiente e agir sobre ele ao longo do tempo, seguindo sua própria agenda e buscando realizar o que ela prevê para o futuro. (FRANKLIN; GRAESSER, 1996)

Existem algumas ideias sobre agentes contidas nessa definição. Primeiramente, muitas outras definições partem da ideia do agente como uma entidade que age em nome de uma outra (como por exemplo uma interface de usuário que age em nome do usuário de um computador), mas a fim de preservar seu caráter geral, a definição de Franklin e Graesser (1996) segue o caminho de definir agente focando apenas na entidade que age, uma vez que essa definição engloba também a possibilidade de agir em nome de outra pessoa. Ela é construída com base em elementos comuns entre entidades que agem de uma maneira geral, como humanos, animais, robôs e agentes de software. Todas essas entidades estão inseridas em um ambiente, podem senti-lo de alguma forma e agir de forma autônoma e contínua sobre ele (seguindo seus próprios objetivos). Além disso, nessa definição um sistema é ou não um agente a depender do ambiente onde está inserido. Um robô que conta apenas com sensores visuais em um ambiente completamente desprovido de luz não pode ser considerado um agente.

3.1.2 Porque construir Modelos Baseados em Agentes

A modelagem baseada em agentes (MBA) é uma técnica muito poderosa que tem encontrado inúmeras aplicações na área de sistemas complexos. Nessa abordagem um sistema é modelado com-

putacionalmente como uma coleção de agentes inseridos em um determinado ambiente e capazes de apresentar comportamentos e interações condizentes com o sistema representado. Ela apresenta alguns benefícios em relação a outras abordagens para a modelagem de sistemas complexos, que Bonabeau (2002) resume nos seguintes pontos:

1. MBA é capaz de capturar fenômenos emergentes;
2. MBA provê uma descrição natural do sistema;
3. MBA é flexível;

Os fenômenos emergentes surgem da interação de entidades individuais, assim a modelagem baseada em agentes é capaz de captura-los pois inclui tanto as características e processos individuais quanto as interações. Algumas características de um sistema que podem levar a existência de um possível fenômeno emergente é: comportamento individual não linear (como com regras do tipo se-então, e acoplamento não linear); comportamento individual com memória, dependente do caminho, com histerese, ou com correlações temporais, que podem incluir aprendizado ou adaptação (dois fenômenos importantes no nosso sistema de evolução de Oscines); interação entre as entidades individuais heterogêneas capaz de gerar efeitos de rede (que também é o caso do nosso sistema); e comportamento que não pode ser suavizado por um calculo médio, como quando o sistema maximiza flutuações e faz com que elas desempenhem um papel central na dinâmica. (BONABEAU, 2002)

Com relação ao ponto 2, em algumas situações a MBA é a linguagem mais natural para a construção de um modelo, como por exemplo: quando o comportamento dos indivíduos não pode ser descrito claramente através de taxas de transição; quando o comportamento do indivíduo é complexo (pois a princípio podemos modelar tudo com equações, porém quanto mais complexo o comportamento normalmente mais complicadas se tornam as equações que o descrevem); ou quando existe estocasticidade no comportamento do agente (BONABEAU, 2002).

Sobre o ponto 3, a flexibilidade do uso de agentes na modelagem se apresenta de diversas maneiras. O número de agentes utilizado pode ser controlado, podemos estabelecer os mais variados tipos de comportamentos, ações e sensações nos agentes, podemos variar as características do ambiente onde estão inseridos, e podemos mudar os níveis de descrição ou agregação, representando agentes agregados, grupos de agentes, agentes solitários e partes de agentes. (BONABEAU, 2002)

Quando então devemos usar modelagem baseada em agentes? Com base nesses benefícios podemos apresentar diversas situações nas quais a estrutura e as características de um sistema tornam o uso da modelagem baseada em agentes interessante (MACAL; NORTH, 2014; BONABEAU, 2002). Elas são as seguintes:

- Quando as interações entre os agentes são complexas, não lineares, descontínuas ou discretas;
- Quando os agentes não possuem posições fixas e o comportamento e as interações são espacialmente dependentes;
- Quando a população de agentes é heterogênea e cada indivíduo é potencialmente diferente;
- Quando a topologia das interações é heterogênea, complexa ou dinâmica, com relacionamentos que podem dissolver ou se formar;

- Quando os agentes possuem comportamento discreto ou complexo, incluindo aprendizagem, adaptação e memoria;
- Quando os agentes se organizam em estruturas nas quais comportamentos e interações discretas ou complexas são relevantes;

3.1.3 Design de um MBA

A construção de um modelo baseado em agentes segue a mesma agenda da construção de qualquer outro tipo de modelo matemático e computacional. Uma vez que sabemos a pergunta que o modelo se propõe a responder, e as principais características do sistema que ele visa representar, entramos em uma sequência de passos que são comuns ao desenvolvimento de qualquer modelo, acrecidos de outros dois passos que estão estritamente correlacionados com a modelagem baseada em agentes (MACAL; NORTH, 2014). Estes passos estão representados na figura 3.1.

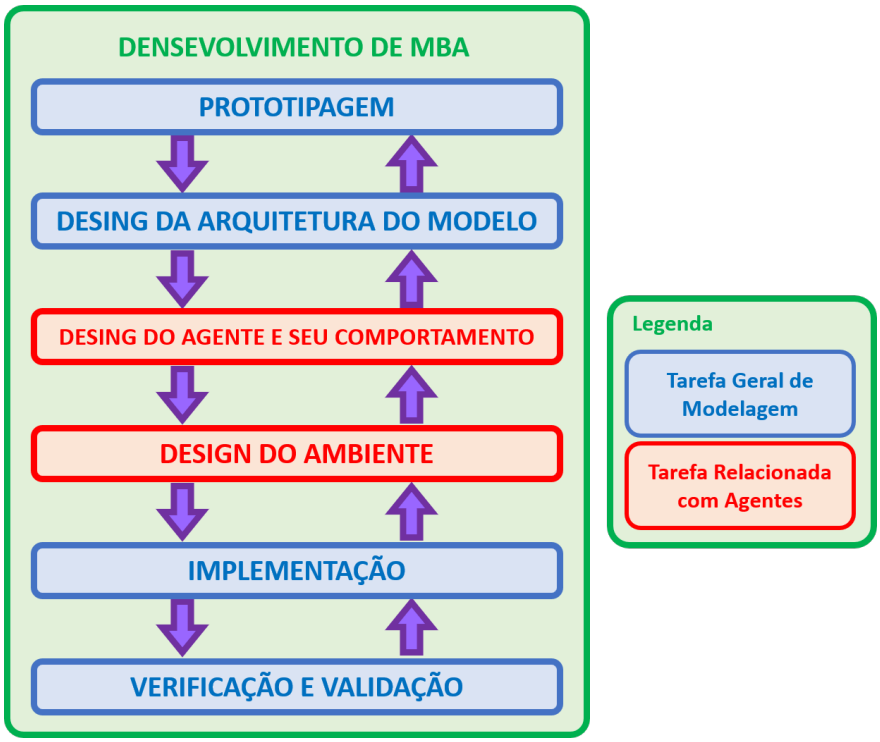


Fig. 3.1: Fluxograma com a representação do processo de desenvolvimento de um modelo baseado e agentes. As tarefas em vermelhos são específicas de modelo baseados em agentes. (Figura adaptada de Macal e North (2014))

Vamos focar nos dois passos em vermelho da figura 3.1 e falaremos de alguns pontos relacionados a implementação. Primeiramente vamos abordar o design dos agentes, descrevendo diferentes arquiteturas para sua estrutura interna e em seguida vamos falar sobre o design do ambiente, apresentando algumas topologias que determinam as possíveis interações. Por fim vamos fazer algumas considerações sobre como o software de um modelo deve ser estruturado usando programação orientada a objetos e falar sobre duas diferentes maneiras de implementação para as atualizações temporais.

O Design de um Agente

Vimos na definição de agente autônomo de Franklin e Graesser (1996) que "um agente autônomo é um sistema que está situado dentro de um ambiente e é parte dele, capaz de sentir o ambiente e agir sobre ele[...]". Segundo essa definição os agentes devem ser capazes de sentir e de agir, e como ela contempla o tipo mais simples de agente podemos deduzir que os agentes mais sofisticados também vão estar aparelhados para desempenhar essas duas funções. Os meios através dos quais os agentes são capazes de sentir os ambiente no qual estão inseridos são denominados **sensores**, e os que lhes possibilitam agir são denominados **atuadores**.

Além disso a definição de Franklin e Graesser (1996) ainda coloca que os agentes sentem e agem sobre o ambiente "ao longo do tempo, seguindo sua própria agenda e buscando realizar o que ela prevê para o futuro". Isso implica que os agentes possuem algumas propriedades, entre elas autonomia (que dá origem ao nome **agentes autônomos**), que se refere a capacidade dos agentes de tomarem as próprias decisões. Então alem de sentir e agir os agentes também dispõem de meios de determinar como vão agir com base nas informações coletadas a partir de seus sensores, e existem diferentes arquiteturas com diferentes ciclos operacionais capazes de desempenhar esse papel (GUDWIN, 2010).

O tipo mais simples de agente é o **agente reflexivo** (também chamado de reativo), cujo ciclo operacional está representado na figura 3.2. Nele as ações tomadas pelos atuadores são definidas através de uma função direta das entradas captadas nos sensores. Essa função pode ser uma função matemática, camadas ocultas de uma rede neural ou um conjunto de regras condicionais. Os objetivos desse tipo de agente são definidos implicitamente através da escolha de uma boa função $f(x)$.

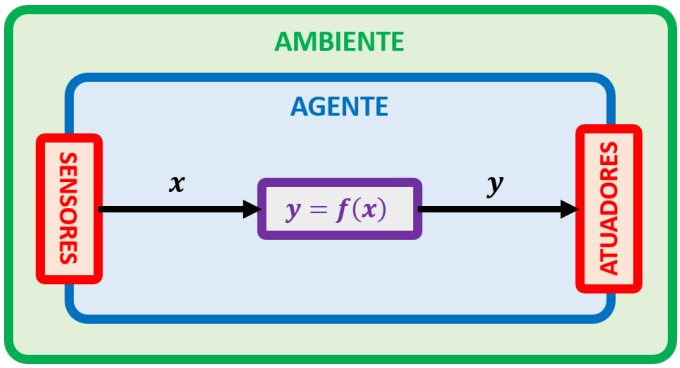


Fig. 3.2: Representação do ciclo operacional de um agente reativo (reflexivo). (Figura adaptada de Gudwin (2010))

Uma arquitetura um pouco mais sofisticada é a de **agentes comportamentais**, criada por Brooks (1991), cujo ciclo operacional está representado na figura 3.3. Na época as pesquisas relacionadas a inteligência artificial possuíam um grande enfoque na representação simbólica de informação, e os trabalhos de Brooks (1991) com robótica visavam mostrar que a representação simbólica não era necessária para a exibição de comportamento inteligente. Para isso ele elaborou uma arquitetura para agentes constituída por módulos independentes de percepção e ação que funcionavam paralelamente. O módulo de ação dos agentes comportamentais possui um conjunto pré definido de comportamentos, e a principal diferença entre estes e os agentes reativos é que o comportamento executado pode ser tanto selecionado de maneira independente pelo módulo de ação como selecionado por influência do módulo de percepção.

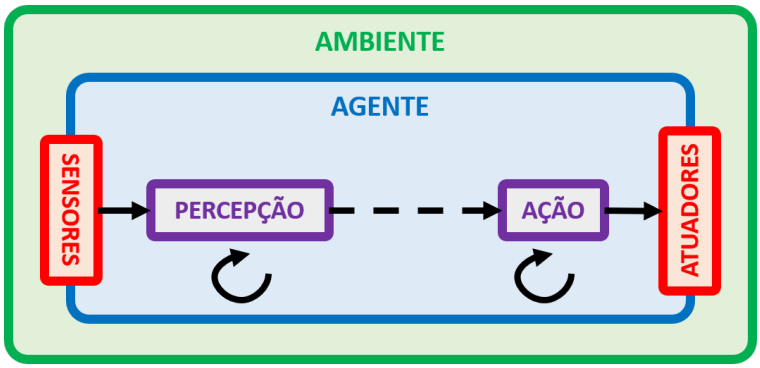


Fig. 3.3: Representação do ciclo operacional de um agente comportamental. (Figura adaptada de Gudwin (2010))

Agora veremos o tipo de arquitetura utilizada no modelo dessa dissertação, a de **agentes deliberativos** (ou planejadores), cujo ciclo operacional está representado na figura 3.4. Essa arquitetura possui um componente extra com relação aos agentes reativos, que é o **modelo de mundo**, responsável por armazenar informações e características do ambiente e situar o agente dentro dele. O componente de atuação pode funcionar de maneira independente, como no caso dos agentes comportamentais, ou apenas de forma dependente da percepção, porém eles não se comunicam diretamente. As informações obtidas pelos sensores são processadas e/ou armazenadas no modelo de mundo e então usadas para decidir quais ações serão desempenhadas pelos atuadores.

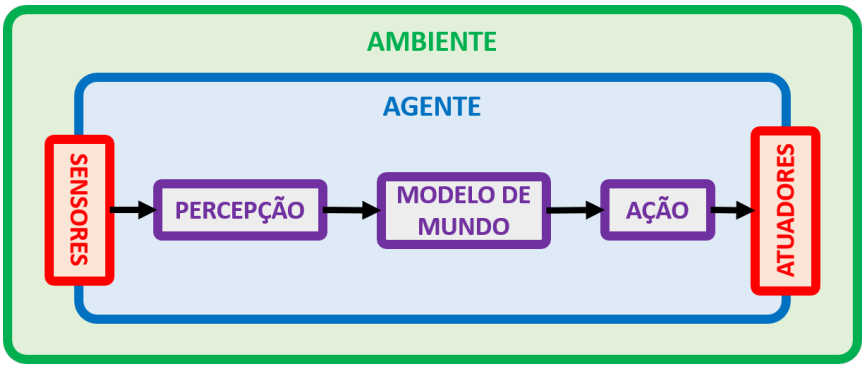


Fig. 3.4: Representação do ciclo operacional de um agente deliberativo (planejador). (Figura adaptada de Gudwin (2010))

A definição de agente de Franklin e Graesser (1996) tem implícita nela as quatro primeiras características da tabela 3.1, que estão presentes nos agentes reativos, nos comportamentais e nos deliberativos, porém as características de aprendizagem e flexibilidade exigem a presença de um modelo de mundo, e por esse motivo são próprias apenas dos agentes deliberativos.

Modelos de mundo podem ter uma grande variabilidade de características e propriedades. O modelo de mundo pode ser inteiramente conhecido pelo agente ou construído com base nas informações coletadas pelos sensores. Ele pode conter informações sobre a geografia ou sobre a física do ambiente (caso o ambiente possua alguma dessas características). Pode possuir memória, com informações referentes ao ambiente, aos estados internos de próprio agente ou informações coletadas sobre

outros agentes. Pode possuir aprendizado e a capacidade de planejar, antecipando o impacto que suas próprias ações terão sobre o mundo e elaborando planos que podem ser avaliados e descartados no processo de tomada de decisão.

Uma outra arquitetura é a de **agentes emocionais**, cujo ciclo operacional está representado na figura 3.5. Com ela é possível construir agentes que possuem **personificação**, a ultima propriedade da tabela 3.1. Um outro componente aparece no ciclo operacional, responsável por armazenar o **sistema de valores** do agente, assim como seu estado emocional. Ele atua paralelamente ao modelo de mundo, processando as informações e interferindo nas decisões de uma maneira mais subjetiva. O sistema de valores pode armazenar o estado emocional do agente com relação a diferentes emoções (como medo, desejo, dor), pode avaliar quão bem o objetivo de um agente está sendo cumprido e alterar o estado emocional ou interferir no processo de tomada de decisão. Também é possível fazer com que o modelo de mundo seja capaz de alterar o estado emocional do agente ao antecipar possíveis estados futuros.

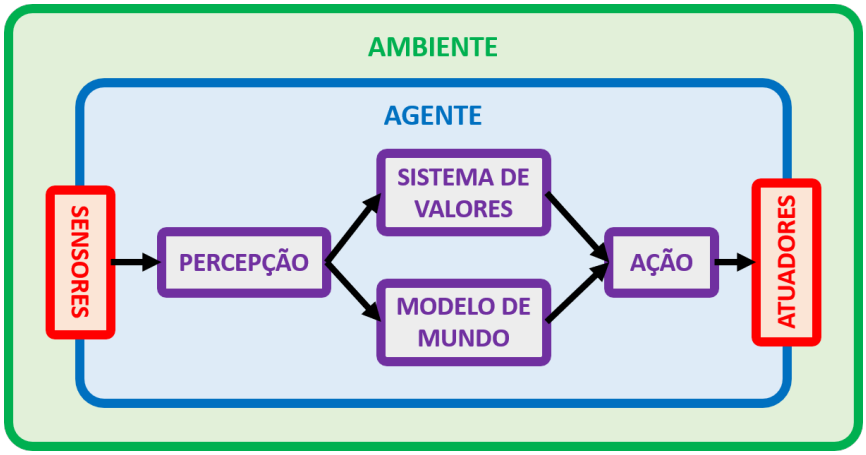


Fig. 3.5: Representação do ciclo operacional de um agente emocional. (Figura adaptada de Gudwin (2010))

Das propriedades da tabela 3.1 a única que não falamos ainda é a comunicatividade. Caso os agentes disponham de um canal de comunicação direto com outros agentes nós os chamamos de **agentes comunicativos**, porém existem outras maneiras de realizar comunicação entre agentes. Os agentes emocionais podem dispor de algum mecanismo capaz de externar seu estado emocional e captar o de outros, o que pode se configurar como um tipo de comunicação. Além disso alguns agentes estão inseridos em ambientes suficientemente complexos para que um canal de comunicação direta não seja necessário, sendo a informação transmitida através do próprio ambiente. Esses agentes são chamados de **agentes semióticos**, que normalmente possuem sensores e atuadores mais complexos.

O Design do Ambiente

Quando falamos de ambiente em modelagem baseada em agentes não necessariamente estamos falando de uma região espacialmente explícita bidimensional ou tridimensional onde os agentes possuem um corpo físico. Antes disso o ambiente é a entidade na qual os agentes estão inseridos, que pode possuir diferentes topologias que determinam as interações possíveis. Além disso ele pode ser

dotado de diferentes condições de contorno, ser estático ou dinâmico, homogêneo ou heterogêneo e pode possuir diferentes tipos de agentes ou objetos com os quais os agentes podem interagir.

Modelos baseados em agentes comumente buscam descrever fenômenos emergentes que surgem da interação dos componentes de um sistema, e por esse motivo a topologia escolhida para o modelo desempenha um papel muito importante. Determinar a topologia correta é determinar corretamente quais interações são possíveis. Segundo Macal e North (2014), as topologias mais comuns são:

1. **Sopa:** um modelo não espacial onde os agentes não possuem atributos locais;
2. **Grade ou treliça:** um automato celular representando padrões de interação de agentes e informação local disponível através de uma grade ou treliça. Células imediatamente em volta de um agente são sua **vizinhança**. A localização de um agente é dada pelo índice da célula da grade onde ele se encontra;
3. **Espaço euclidiano:** Os agentes vagam em espaços 2D ou 3D. A localização de um agente é sua coordenada relativa ou geoespacial;
4. **Sistema de informação geográfica:** Agentes se movimentam e interagem com *patches* realistas de paisagens geoespaciais. A localização de um agente é uma unidade geográfica (por exemplo, código postal) ou coordenadas geoespaciais;
5. **Redes:** As redes podem ser estáticas (com arestas pré-especificadas) ou dinâmicas (com arestas determinadas por mecanismos de relacionamento). A localização de um agente é a localização relativa do nó na rede;

Para cada uma dessas topologias é necessário ainda ficar atento a escolha do tamanho do mundo (com exceção da sopa de agentes, para o qual esse conceito não se aplica). Em mundos pequenos podem surgir efeitos de tamanho finito interferindo na captação dos fenômenos emergentes, já mundos grandes podem tornar a análise computacional mais lenta sem qualquer contribuição significativa. As condições de contorno também devem ser observadas para as topologias de grade, espaço euclidiano e sistemas de informação geográfica, sendo as mais comuns delas a condição de contorno aperiódica, e as condições de contorno periódica esférica e toroidal.

O Design do Software

O desenvolvimento de software para modelos basados em agentes possui uma relação natural com a programação orientada a objetos (MACAL; NORTH, 2014). Agentes e objetos não são a mesma coisa. Programação orientada a objetos é um padrão de desenvolvimento de software cuja estrutura se baseia em unidades chamadas de **objetos** e suas interações. Um objeto é uma metáfora computacional que emula um objeto do cotidiano. Eles possuem atributos que os permitem armazenar dados e possuem métodos que ditam seu comportamento, porém diferentes dos agentes os objetos não são pró-ativos, e sim executam métodos usando os dados fornecidos quando lhes é solicitado. Assim temos uma outra possível maneira de descrever agentes: eles são objetos que tem controle sobre sua própria execução (ABAR et al., 2017).

A programação orientada a objetos é o meio mais utilizado para emular agentes em software. Para isso cria-se uma classe para gerar apenas um objeto emulando um ambiente. Além dos atributos

e métodos que que descrevam suas características, este objeto ambiente deve conter um vetor com objetos de outra classe que emula os agentes. Uma vez iniciados os objetos da classe dos agentes devem seguir em atividade de maneira autônoma até que decidam interromper seu ciclo operacional por conta própria. A figura 3.6 contém um esquema relacionando como os objetos devem estar estruturados no código.

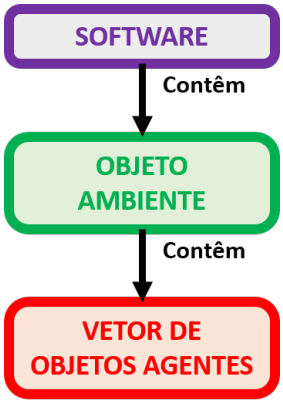


Fig. 3.6: Esquema com representação da estrutura de um código de um modelo baseado em agentes.

Existem muitas linguagens com suporte a programação orientada a objetos que podem ser utilizadas para criação de modelos baseados em agentes, como por exemplo C++ e Python. Porém existem também muitos softwares dedicados que já possuem toda essa estrutura pronta, assim como muitas ferramentas complementares. Esses softwares variam enormemente em complexidade e escalabilidade, indo desde o StarLogo, que é extremamente simples, porém possui baixa escalabilidade, servindo mais para propósitos educacionais que para modelagem cientifica, até o Swarm, que possui uma escalabilidade muito alta, servindo para simular com robustez uma ampla variedade de fenômenos biológicos, sociais e econômicos, sendo porém muito mais complicado de utilizar. (ABAR et al., 2017)

Um dos conceitos de programação orientada a objetos mais importantes para a construção de MBAs é o conceito de **encapsulamento**. Na seção de design do agente enfatizamos que eles só dispõem de duas maneiras de interagir com o ambiente onde estão inseridos: eles sentem o ambiente através de sensores e agem sobre ele através de atuadores. Cada agente deve ser o único a ter controle sobre seu ciclo operacional e estados internos uma vez que sua execução comece.

Agora vamos abordar um outro aspecto da implementação do modelo, que é como fazer as atualizações temporais dos estados dos agentes. Vamos supor que implementamos um modelo baseado em agentes no qual uma quantidade constante deles é atualizada sempre seguindo uma sequência pré-estabelecida (primeiro o agente 1, seguido do agente 2...). Como alguns sempre agem antes de outros, isso poderia causar uma assimetria de condições entre eles, pois agir primeiro pode ser uma vantajoso em algumas situações (assim como desvantajoso em outras). Isso torna necessário fazer as atualizações de uma maneira que a ordem de atualização não privilegie nenhum dos agentes.

Existem duas maneiras populares de fazer isso, através de atualizações **sincrônicas** e **assincrônicas**. Na atualização assincrônica os agentes são atualizados em cada passo de tempo seguindo uma sequência aleatória, onde o resultado das ações de um deles está imediatamente disponível para o próximo a ser atualizado. Já na atualização sincrônica os resultados das ações de cada agente vão

sendo armazenados para que eles sejam atualizados simultaneamente ao fim de um passo de tempo (aqui não há a necessidade de aleatorização, pois todos os agentes agem sob exatamente as mesmas condições). Da atualização sincrônica para a assincrônica há um atraso no acesso a informação dos resultados da atualização de outros agentes, assim as duas maneiras podem apresentar resultados diferentes, e por esse motivo é necessário avaliar cuidadosamente qual escolher a depender do sistema estudado e deixar claro qual abordagem foi utilizada. Caso a intenção seja emular tempo contínuo a atualização assíncrona é geralmente uma melhor opção. (CARON-LORMIER et al., 2008)

Por fim, MBAs são complicados de reportar quando comparados a modelos analíticos, que encontram na matemática uma linguagem padrão de descrição. Por esse motivo eles frequentemente são descritos verbalmente, sem o detalhamento de equações, regras e sequências utilizados, o que compromete a compreensão do modelo e a replicação dos resultados. Para servir como padrão um de comunicação de MBAs foi criado o protocolo ODD[1] (visão geral, conceitos de *desing* e detalhes) (GRIMM et al., 2006). A visão geral do nosso modelo se encontra na subseção 4.2, enquanto os conceitos de *design* e detalhes se encontram no apêndice A.

3.2 Redes Complexas

Além da modelagem baseada em agentes, uma outra ferramenta muito utilizada para analisar sistemas complexos através de uma uma metodologia *botton-up* é a abordagem das **redes complexas**.

Assim como o conceito de agentes a ideia de redes está muito presente em nosso cotidiano. Temos redes de amigos e familiares, utilizamos diariamente a internet, que é uma rede mundial de computadores, onde temos acesso a diversos tipos de redes sociais, dependemos do bom funcionamento diário de diversas redes, como da rede de transportes, de telefonia e de distribuição de energia e água. Além disso diversas outras redes também são estudadas por seu valor cientifico e tecnológico: a rede de componentes químicos que compõem uma célula (conectados por reações químicas), a rede de neurônios interligados por sinapses que compõem nosso sistema nervoso, redes ecológicas e diversas outras. (ALBERT; BARABÁSI, 2002)

Podemos definir rede como um conceito abstrato que "representa uma ampla variedade de estruturas nas quais as entidades de um sistema complexo são representadas pelos nós de rede, e as relações ou interações entre essas entidades são apreendidas por meio das arestas da rede"(ESTRADA; KNIGHT, 2015). Matematicamente elas são representadas através de estruturas matemáticas denominadas **grafos**, e são estudadas através de técnicas provenientes da teoria dos grafos e da física estatística.

Vamos agora definir matematicamente o conceito de grafo. Um grafo G é um par (V, E), o que significa que ele é uma combinação de nós e arestas. O conjunto V é chamado de **conjunto dos nós** de G, e seus elementos são os **nós** (também chamados de vértices) que se ligam através de arestas. E é o **conjunto de arestas** de G, cujos elementos são as **arestas**. Cada aresta liga dois nós não necessariamente distintos, e por isso nós as representamos como pares (v_i, v_j), onde v_i e v_j pertencem ao conjunto V.

O conjunto E das arestas recebe algumas nomenclaturas específicas para casos especias. O par (v, v) representa uma aresta que liga o nó v a ele mesmo. Se $(v, v) \in E$ para todo $v \in V$, ou seja, se todo o nó possui um laço, que é uma aresta que o ligue a si mesmo, dizemos que o conjunto E das

[1] Em inglês, *Overview, Design concepts and Details.*

arestas é **reflexivo**. A situação oposta também é um caso especial, se $(v, v) \notin E$ para todo $v \in V$, ou seja, se nenhum nó do grafo possui uma laço, dizemos que o conjunto E é **anti-reflexivo**. Por fim, se $(v_1, v_2) \in E \iff (v_2, v_1) \in E$, ou seja, uma aresta (v_1, v_2) está presente no grafo se e somente se a aresta simétrica correspondente a a ela (v_2, v_1) também estiver, dizemos que o conjunto E é **simétrico**. (ESTRADA; KNIGHT, 2015)

Em posse dos conceitos de anti-reflexividade e simetria é possível classificar os grafos nas seguintes categorias:

- Se E é não simétrico então G é um **grafo direcionado** (ou **digrafo**), pois existe a possibilidade de um nó v_1 estar conectado com outro nó v_2 sem que v_2 esteja conectado com v_1;
- Se E é simétrico então G é um **grafo não direcionado**, pois se um nó v_1 está conectado com outro nó v_2 isso implica necessariamente que v_2 também está conectado com v_1;
- Se E é simétrico, anti-reflexivo e não contém arestas duplicadas, então G é um **grafo simples**, o que significa que ele é não direcionado, não possui laços e cada nó só se conecta a outro por uma única aresta;

A figura 3.7 é uma representação gráfica de um grafo simples, onde os círculos numerados representam os nós do conjunto de vértices $V = \{1, 2, 3, 4, 5, 6\}$ e as linhas entre eles representam as ligações do conjunto de arestas $E = \{\{1, 2\}, \{1, 5\}, \{2, 3\}, \{2, 5\}, \{3, 4\}, \{4, 5\}, \{4, 6\}\}$. Não devemos confundir a representação do grafo com o grafo em si, pois existem muitas maneiras de representar o mesmo grafo, mudando as posições espaciais dos nós e mantendo as mesmas conexões.

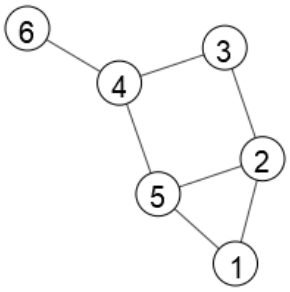


Fig. 3.7: Representação gráfica do grafo simples $G = (V, E)$, onde $V = \{1, 2, 3, 4, 5, 6\}$ e $E = \{\{1, 2\}, \{1, 5\}, \{2, 3\}, \{2, 5\}, \{3, 4\}, \{4, 5\}, \{4, 6\}\}$.

É possível definir uma maneira de representar grafos univocamente através de matrizes (ESTRADA; KNIGHT, 2015). Para um grafo simples $G = (V, E)$, onde $V = \{1, 2, 3, ..., n\}$ podemos definir:

$$a_{ij} = \begin{cases} 1 & (i, j) \in E \\ 0 & (i, j) \notin E \end{cases} \tag{3.1}$$

com $1 \leq i, j \leq n$. A matriz $A = (a_{ij})$ é chamada **matriz de adjacência**. Ela contém toda a informação disponível sobre o grafo, sendo redundante inclusive, uma vez que a informação sobre a

ligação entre o nó i e o nó j está tanto disponível no elemento a_{ij} quanto no a_{ji}. Em posse da matriz de adjacência é possível utilizar álgebra matricial para determinar algumas características dos grafos. Existem também outras maneiras de representar redes, como a matriz de incidência e a matriz de vizinhança (ANDRADE et al., 2008).

Na figura 3.7 podemos perceber que o nó 6 possui apenas uma conexão, os nós 1 e 3 possuem duas cada um e os nós 2, 4 e 5 possuem 3 cada um. O número de conexões de um nó i é chamado de **grau**, que vamos representar por d_i. A soma do grau de todos os vértices é o dobro do valor de arestas (m), pois ao somarmos os graus cada aresta é contabilizada duas vezes, uma para cada nó a qual está ligada. A partir do grau de todos os vértices podemos determinar duas propriedades estruturais clássicas, que são grandezas criadas para caracterizar uma rede. Uma delas é o valor do **grau médio** da rede, dado por:

$$\bar{d} = \frac{1}{n} \sum_{i \in V} d_i = \frac{2m}{n} \tag{3.2}$$

e a outra é a **distribuição de grau**, que é calculada com frequência para qualquer rede estudada, pois é uma maneira de se conseguir visualizar a estrutura de uma rede, podendo determinar o mecanismo de formação de redes reais ou fornecer pistas sobre ele.

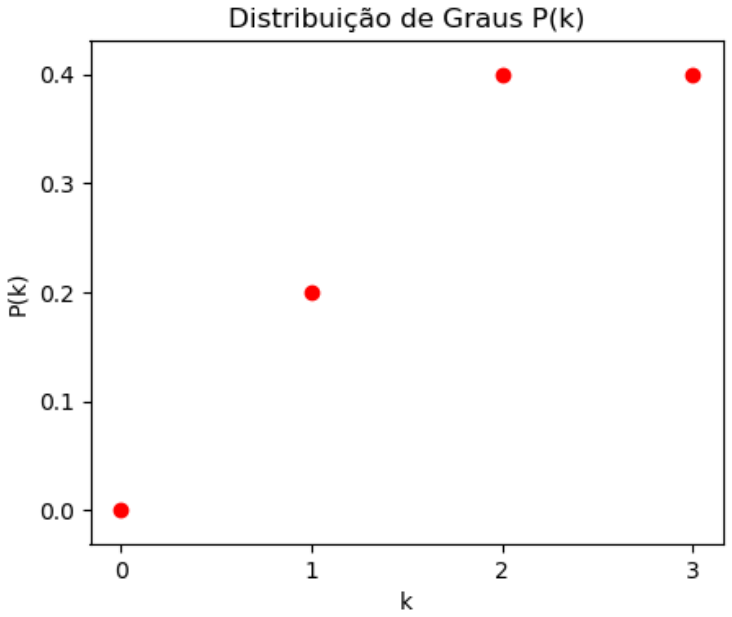


Fig. 3.8: Exemplo de distribuição de graus do grafo da figura 3.9

Nosso interesse, ao usar redes complexas, é conseguir determinar algumas propriedades estruturais das nossas redes ligadas aos fenômenos emergentes que buscamos estudar, que são a especiação e a formação de dialetos. Para isso vamos trabalhar com os conceitos de conectividade e modularidade.

3.2.1 Conectividade

Para entender o conceito de conectividade é preciso primeiro apresentar algumas definições. Uma **cadeia** em uma rede é uma série de arestas, representada por:

$$(u_1, w_1), (u_2, w_2), ..., (u_p, w_p) \tag{3.3}$$

onde $w_i = u_{i+1} (i = 1, 2, ..., p - 1)$, o que significa o segundo nó que qualquer aresta da cadeia é sempre igual ao primeiro nó da sua sucessora. Uma cadeia é **fechada** quando $u_1 = w_p$, ou seja, quando o primeiro nó da primeira aresta é igual ao ultimo nó da ultima aresta. Uma **trilha** é uma rede na qual todas as arestas (u_i, v_i) são distintas. Um **caminho** é uma trilha para qual todos os u_i são distintos. Um caminho fechado se chama **ciclo** (ou **circuito**). Um grafo que não possua nenhum ciclo é classificado como **acíclico**. O comprimento de uma cadeia, trilha ou caminho é dado pelo número de arestas que o compõe. Ciclos de comprimento 3 são chamados de triângulos. (ESTRADA; KNIGHT, 2015)

Uma rede está **conectada** se existe um caminho conectando quaisquer dois nós pertencentes a ela. Quando uma rede não está conectada ela pode ser dividida em **componentes** que estão individualmente conectados. Nas redes não direcionadas nunca há arestas indo de um componente a outro, conforme podemos ver na figura 3.9, porém para redes direcionadas podemos ter componentes que se ligam através de arestas que seguem em apenas um sentido. Quando não possui nenhuma aresta saindo dele, um componente é classificado como **fortemente conectado**.

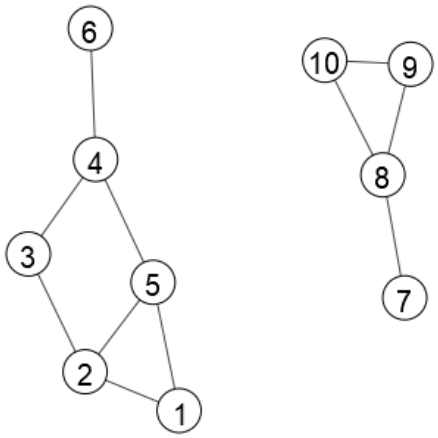


Fig. 3.9: Representação gráfica de grafo não direcionado e não conectado com 2 componentes.

O conceito de espécie que estamos usando para analisar o modelo desenvolvido nesse trabalho é o de espécie por reconhecimento, de Paterson e McEvey (1993). Com base nele vamos montar uma rede onde os nós são os pássaros machos e fêmeas, e as arestas serão adicionadas entre quaisquer deles que conseguem se reconhecer e acasalar, independente da distância. Na condição inicial da simulação a rede estará conectada, porém caso a rede se desconecte em dois ou mais componentes, teremos então dois grupos de pássaros não se reconhecem, ou seja, duas ou mais especies pelo conceito de espécie por reconhecimento. Assim, a conectividade da rede será usada para determinar a ocorrência ou não de especiação.

3.2.2 Comunidades e Modularidade

Para entender o conceito de modularidade primeiro precisamos compreender o conceito de comunidade, e para isso vamos apresentar mais algumas definições. Um grafo $G_s = (V_s, E_s)$ é um **subgrafo** de $G = (V, E)$ se $V_s \subseteq V$ e $E_s \subseteq V_s \otimes V_s \cap E$, ou seja, o conjunto de nós V_s do subgrafo G_s deve ser composto apenas por nós pertencentes ao grafo G e o subconjunto de arestas E_s não apenas deve ser composto de arestas pertencentes a G como essas arestas só podem estar ligadas a nós de V_s. Assim, dado um subgrafo $G_1 = (C, E_1)$ contido em um grafo $G = (V, E)$ com $n_C = |C|$ vértices, podemos definir os graus **interno** (k_i^{int}) e **externo** (k_i^{ext}) de cada um da seguinte maneira:

$$k_i^{int} = \sum_{j \in C} a_{ij} \quad , \quad k_i^{ext} = \sum_{j \in \bar{C}} a_{ij} \tag{3.4}$$

onde $A = a_{ij}$ é a matriz de adjacência de G e $\bar{C}$ é o complemento de C, composto por todos os nós que pertencem a V mas não pertencem a C, ou seja, nós de G não pertencentes ao subgrafo G_1. Usando os graus interno e externo de todos os nós podemos identificar quantas ligações conectam os nós dentro do subgrafo G_1 (que denominaremos m_C) e o limite de C, que é o número arestas que conectam os nós de C com os nós de $\bar{C}$ (que denominaremos $m_{C-\bar{C}}$):

$$m_C = \frac{1}{2} \sum_{i \in C} k_i^{int} \quad , \quad m_{C-\bar{C}} = \sum_{i \in \bar{C}} k_i^{ext} \tag{3.5}$$

Uma propriedade estrutural que podemos calcular para as redes é a **densidade** $\delta(G)$. Elá é dada pela razão entre o valor m de ligações da rede e o número de ligações possíveis, que pode ser escrita em termo do número de nós como $n(n-1)/2$:

$$\delta(G) = \frac{2m}{n(n-1)} \tag{3.6}$$

Podemos definir de maneira similar a **densidade interna** $\delta_{int}(G)$, que é a densidade de ligações internas de G_1:

$$\delta_{int}(C) = \frac{2m_C}{n_C(n_C - 1)} = \frac{\sum_{i \in C} k_i^{int}}{n_C(n_C - 1)} \tag{3.7}$$

onde o número de ligações é dado por m_C e só são contabilizadas as ligações possíveis entre os nós pertencentes a C. Outra grandeza que precisamos definir é a **densidade externa** $\delta_{ext}(G)$, que é a densidade de ligações entre C e $\bar{C}$:

$$\delta_{ext}(C) = \frac{m_{C-\bar{C}}}{n_C(n - n_C)} = \frac{\sum_{i \in C} k_i^{ext}}{n_C(n_C - 1)} \tag{3.8}$$

Com essas definições e a definição de conectividade que vimos na sessão 3.2.1, podemos então definir comunidade:

> Uma comunidade de uma rede é um conjunto de nós internamente conectados que possuem uma densidade interna significativamente maior que a densidade externa. (ESTRADA; KNIGHT, 2015)

O padrão de interconexão entre os nós da rede formando comunidades reproduz o comportamento esperado para dialetos diferentes em diferentes populações de Passeriformes. Pássaros que pertencem a mesma comunidade produzem cantos que possuem dialetos similares, e por isso possuem uma probabilidade maior de estarem interconectados. Então para detectar a formação de dialetos precisamos medir se as redes de reconhecimento do nosso sistema possuem alta modularidade.

Para medir a qualidade da estrutura de comunidades Newman e Girvan (2004) sugerem o conceito de **modularidade**. Seja $G = (V, E)$ um grafo com n nós e m arestas com uma matriz de adjacência $A = (a_{ij})$ e supondo que nós tenhamos dividido os nós em n_C comunidades $V_1, V_2, ..., V_{n_C}$. Definimos s_{ir} como igual a 1 se o nó i está na comunidade r e como igual a 0 caso não esteja. A modularidade é então definida como:

$$Q = \frac{1}{4m} \sum_{r=1}^{n_C} \sum_{i,j=1}^{n} \left(a_{ij} - \frac{k_i k_j}{2m} \right) s_{ir} s_{jr} \tag{3.9}$$

Podemos interpretar a modularidade como a soma sobre todas as comunidades da diferença entre a fração de arestas dentro da comunidade e a fração esperada para uma rede aleatória com os mesmos graus para cada nó:

$$Q = \sum_{k=1}^{n_C} \left[\frac{|E_k|}{m} - \frac{1}{4m^2} \left(\sum_{j \in V_k} k_j \right)^2 \right] \tag{3.10}$$

onde $|E_k|$ é o numero de arestas dentro da comunidade V_k. O valor da modularidade pode assumir valores entre -1 e 1 a depender da rede e dos conjuntos V_k escolhidos. Quando a densidade interna dos componentes é semelhante a densidade de uma rede aleatória, então encontramos o valor de Q próximo a 0, indicando que não há uma estrutura de comunidades (se as redes são aleatórias não se espera que existam grupos com uma maior probabilidade de se conectarem entre si). Quanto maior o valor de Q mais os conjuntos escolhidos representam com qualidade uma estrutura de comunidades. Valores negativos significam que os conjuntos escolhidos não representam as comunidades da rede, possuindo menos conexões entre seus nós que com o resto da rede.

Então temos um problema a mais, como escolher corretamente os conjuntos V_k de uma rede que representem de fato sua estrutura natural de comunidades? Para isso existem diversos algoritmos, e o que utilizamos foi o **método de Louvain para detecção de comunidades** (BLONDEL et al., 2008), um algoritmo de optimização de modularidade muito útil para trabalhar com grafos grandes, por ser muito rápido.

O método de Louvain necessita da definição de modularidade para redes ponderadas, dado por:

$$Q = \frac{1}{4m} \sum_{i,j=1} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_1, c_2) \tag{3.11}$$

onde A_{ij} é o peso da aresta que conecta os nós i e j, $k_i = \sum_j A_{ij}$ é a soma dos pesos de todas as arestas ligadas ao nó i, c_i é a comunidade a qual pertence o nó i, a função $\delta(u, v)$ é 1 quando $u = v$ e 0 quando $u \neq v$ e $m = \frac{1}{2} \sum_{ij} A_{ij}$. Essa expressão funciona como a de modularidade de grafos não ponderados no limite de todas as arestas possuírem o mesmo peso.

O método consiste em duas fases que são repetidos iterativamente até que não se obtenha mais nenhum ganho de modularidade:

1. Cada nó da rede começa sendo sua própria comunidade, assim inicialmente existem tantas comunidades quanto nós. Para cada nó i, é calculado como a modularidade varia caso este nó seja retirado de sua própria comunidade e seja incluso na de cada um de seus vizinhos. O nó então é incluído naquela comunidade no qual a variação modularidade foi máxima, mas apenas se ela for positiva, caso não exista variação positiva ele permanece em sua própria comunidade. Esse processo é aplicado repetidamente para todos os nós até que nenhuma variação positiva de modularidade seja mais possível, ou seja, nenhuma mudança individual consegue aumentar a modularidade, quando passamos para o próximo passo;

2. É construída uma segunda rede na qual os nós são as comunidades encontradas durante a primeira fase. O peso das novas conexões é dado pela soma de todas as arestas que conectam duas comunidades. Arestas dentro de cada comunidades são incluídas como loops. A rede gerada por essa segunda fase então é submetida novamente a primeira fase;

Capítulo 4

Resultados Originais

4.1 Introdução

Os Passeriformes são uma ordem muito diversa da classe das aves. Ela é composta de duas subordens, os **Suboscines**, com cerca de 1250 espécies catalogadas, que apresentam pouco efeito de aprendizagem no desenvolvimento do canto, e os **Oscines**, com cerca de 4650 espécies, cujo desenvolvimento do canto é influenciado por aprendizado socialmente mediado (FREEMAN; MONTGOMERY; SCHLUTER, 2017). Os Oscines correspondem sozinhos a cerca de 45% da diversidade na classe das aves, o que leva ao questionamento de qual o fator responsável por essa diversidade.

Os cantos dos Passeriformes são "vocalizações longas e complexas produzidas pelos machos na temporada de acasalamento"(CATCHPOLE; SLATER, 2008). Eles possuem duas funções principais, **atrair a fêmea para o acasalamento** e **defender o território**. O desenvolvimento dos cantos é muito custoso para os Passeriformes, e como um carácter sexual secundário presente majoritariamente nos machos, sua evolução é explicada como um caso de seleção sexual por escolha pela fêmea. Porém para a subordem dos Oscines há também uma componente cultural presente no aprendizado socialmente mediado que influencia no processo evolutivo.

Aprendizagem pode acelerar a especiação, pois comportamentos aprendidos são frutos de evolução genética assim como de evolução cultural, que por ser mais rápida pode aumentar as taxas de mudança (VERZIJDEN et al., 2012; CHEBIB; MARRIOTT, 2016; LACHLAN; SERVEDIO, 2004). Porém o contrário também pode acontecer, por exemplo, em cenários onde a seleção natural ou sexual impulsionam a divergência genética, o aprendizado pode mascarar esses efeitos e inibir a especiação (FREEMAN; MONTGOMERY; SCHLUTER, 2017; VERZIJDEN et al., 2012).

Para investigar o papel da aprendizagem na evolução dos Passeriformes nós construímos um modelo baseado em agentes. A necessidade de uso dessa ferramenta se justifica fenomenologicamente, pois os pássaros, que são os componentes do sistema, exibem aprendizado e adaptação. Além disso as propriedades emergentes que avaliamos, a formação de dialetos e a especiação, são dependentes das interações possíveis entre os agentes, que são heterogêneas e dinâmicas. Um modelo baseado em agentes permite a inserção dessas características de uma maneira muito natural (MACAL; NORTH, 2014; BONABEAU, 2002).

Em nosso modelo não nos baseamos nas características de nenhuma espécie de Passeriforme específica, e levamos em consideração principalmente diferenças gerais entre as duas subordens: a aprendizagem e diferenças de plasticidade e discriminação de cantos.

4.2 Modelo Computacional

As seguir, encontra-se a descrição da visão geral do modelo conforme proposto pelo protocolo ODD (GRIMM et al., 2005). Os conceitos de desing e os detalhes, também pertencentes ao protocolo, podem ser encontrados no apêndice A. O código usado na implementação pode ser encontrado no apêndice B.

4.2.1 Propósito

O propósito desse modelo baseado em agentes é avaliar as diferenças na dinâmica evolutiva de entre dois subgrupos da ordem dos Passeriformes, os Oscines e os Subocines. Consideramos três diferenças entre essas subordens: i) o aprendizado socialmente mediado do canto, presente nos Oscines e quase inexistente nos Suboscines, ii) a maior variabilidade de cantos disponíveis aos machos nos Oscines, e iii) a maior discriminação entre cantos pertencentes e não pertencentes a própria espécie por parte dos Suboscines (FREEMAN; MONTGOMERY; SCHLUTER, 2017). Para isso foi desenvolvido um modelo onde a aptidão não é calculada explicitamente e depende apenas da própria dinâmica social, em um ambiente homogêneo e desprovido de barreiras geográficas. Três aspectos da dinâmica social foram considerados: i) o processo de seleção de parceiro sexual pela fêmea através do canto, ii) o aprendizado do canto dos filhotes machos de Oscines, e iii) competição intraespecífica (que gera heterogeneidade na distribuição espacial dos agentes em nosso modelo).

4.2.2 Variáveis de estado e escalas

Variáveis de estado

Nesse modelo serão feitas simulações contendo apenas indivíduos de um dos dois subgrupos de Passeriformes. Em cada simulação os agentes representarão Passeriformes classificados por: *Aprendizado do Canto* (**Com Aprendizado** ou **Sem Aprendizado**, apenas um tipo por simulação), *Sexo* (**machos** ou **fêmeas**, definido no nascimento de cada agente), *Estágio da Vida* (**filhotes** ou **adultos**, a depender do *Tempo de Cristalização*) e *Relacionamento* (**Com Parceiro** ou **Sem Parceiro**). Há uma variável no modelo de mundo de cada agente para especificar cada um desses estados.

Cada agente possui um conjunto de variáveis de posicionamento espacial e movimentação. O posicionamento espacial é dado por uma variável *Posição*, representada por um par de coordenadas (X, Y) que assumem valores dentro do intervalo $[0; 1[\subset R$, o que significa dizer que o tamanho do mundo é normalizado no modelo de mundo do agente. O movimento é realizado pelo processo **Andar**, detalhado na seção A.4, e depende das variáveis *Passo dos Agentes* e *Ângulo de movimento* para a maioria dos agentes. As exceções são agentes no primeiro ano de vida, que para simular migração utilizam as variáveis *Passo dos Agentes*, *Proporção de Passo de Migração* e *Ângulo de Movimento de Migração*, e as situações nas quais os agentes ficam parados, que dependem das variáveis *Tempo de Ninho* e *Relacionamento*.

Como o modelo de mundo é normalizado, as interações entre agentes dependem de uma distância dada em termos de *Passos dos Agentes*. O *Raio de Audição* é a distância máxima na qual um agente consegue escutar o canto do outro, tanto para fins de aprendizado (descrito pelo processo **Aprender**) quanto para fins de cortejo (descrito pelo processo **Escolher Parceiro**). Todos os machos cantam

com mesma intensidade, e todos os agentes possuem a mesma capacidade de audição. Já o *Raio de Competição* é a distancia na qual os agentes competem entre si pelos recursos do ambiente. Como os recursos não aparecem explicitamente no modelo, a competição é simulada tornando mais provável que um agente morra caso haja um grande número de outros agentes em sua vizinhança, o que feito multiplicando o número de agentes dentro do *Raio de Competição* pela *Probabilidade de Morte*, conforme descrito no processo **Morrer**.

Os agentes contabilizam o tempo de duas formas. Em cada ano os agentes começam com um *Contador de Meses* zerado, que é incrementado em 1 todo mês até que se complete um ano. *Contador de Meses* é comparado nos filhotes com *Tempo de Ninho*, para determinar se eles devem se mover, e com *Tempo de Cristalização*, para determinar se eles devem aprender. Além disso os agentes possuem uma variável *Idade* que é incrementada em 1 todo ano. No ano 0 de vida, após o *Tempo de Ninho* os agentes se movem com um movimento que simula imigração. Quando *idade* $>=$ *Idade Máxima*, os agentes morrem, o que pode acontecer antes disso por competição intraespecífica, como descrito no processo **Morrer**.

Quando em mês de acasalamento, as fêmeas possuem igual probabilidade de acasalarem como qualquer macho em sua vizinhança reconhecido como pertencente a sua própria espécie, dada por *Probabilidade de Acasalamento*, conforme descrito no processo **Escolher Parceiro**. Caso haja acasalamento, o número de filhotes é aleatório, mas nunca superior ao *Máximo de Filhotes por Ninhada*. Cada filhote tem uma *Probabilidade de Mutação* sobre seus *Alelos de Canto*, que pode acarretar, com igual probabilidade, em um aumento ou decréscimo do valor de cada Alelo por uma quantidade igual ao *Tamanho de Mutação*. A reprodução é descrita no processo **Reproduzir**.

Além disso os agentes possuem algumas variáveis ligadas ao genótipo e ao fenótipo do canto, que serão descritas abaixo:

- *Alelos de Canto:* cada agente possui dois *Alelos de Canto* que correspondem a números em um espaço unidimensional contínuo de cantos. Esses alelos são utilizados para calcular o início do *Intervalo de Cantos Possíveis e Reconhecidos*, assim ele controla a herança genética tanto dos valores de canto possíveis para machos, quanto dos cantos reconhecidos por fêmeas e machos como pertencentes a própria espécie, o que apesar de não ser comprovado é uma suposição razoável (LACHLAN; SERVEDIO, 2004). Existem potencialmente infinitos alelos na população, o que é uma condição suficiente para que o genótipo e os fenótipos sejam capazes de assumir uma distribuição normal na população (RIDLEY, 2006). Não há relação de dominância entre quaisquer dois alelos, assim para determinar o inicio do *Intervalo de Cantos Possíveis e Reconhecidos* de um agente é feita uma média entre os valores de seus dois alelos durante sua construção. Os *Alelos de Canto* estão representados em verde na figura 4.1;

- *Intervalo de Cantos Possíveis e Reconhecidos:* Um intervalo contínuo no espaço unidimensional de cantos (representado em roxo na figura 4.1), que possui dois papeis: i) determinar quais cantos são possíveis para um macho e ii) determinar quais valores de cantos são reconhecidos por fêmeas e machos como pertencentes a própria espécie. No modelo do padrão auditivo para o desenvolvimento do canto, representado na figura 2.6, essa variável representaria o padrão bruto de canto que nasce com cada indivíduo. Nos machos, um valor desse intervalo é sorteado para ser o valor inicial do *Valor de Canto*. Esse valor inicial é o *Valor de Canto* definitivo caso o macho seja um Suboscine, ou passa pelo refinamento do processo de aprendizagem socialmente mediada caso seja um Oscine, através do processo **Aprender**, que é executado para

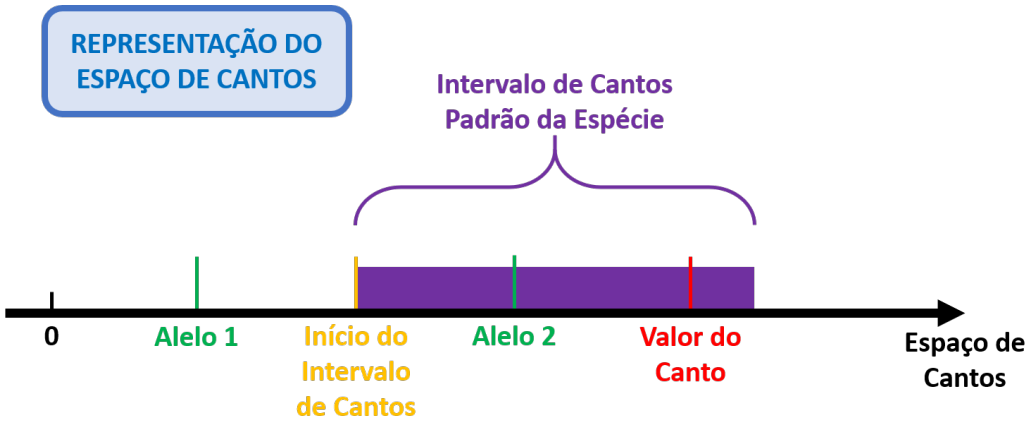


Fig. 4.1: Representação do espaço unidimensional de cantos e algumas variáveis de estado. Em verde temos os valores dos dois *Alelos de Canto*, em amarelo o inicio do *Intervalo de Cantos Possíveis e Reconhecidos*, que é calculado como uma média aritmética dos valores dos *Alelos de Canto*, em roxo temos o próprio *Intervalo de Cantos Possíveis e Reconhecidos* e em vermelho o *Valor de Canto*, que só é utilizado para agentes machos.

Oscines machos filhotes. Por determinar tanto quais cantos podem ser aprendidos como os que podem ser reconhecidos, ao estabelecermos intervalos de canto menores para os Suboscines virtuais garantimos que os Oscines terão uma maior variabilidade de cantos disponíveis por agente e uma menor discriminação com relação a cantos pertencentes a própria espécie, duas das três diferenças que implementamos entre as subordens. O *Intervalos de Canto Possíveis e Reconhecidos* está representado em roxo na figura 4.1, e o início dele, que é calculado como uma média entre os *Alelos de Canto*, está em amarelo;

- *Valor de Canto:* Apenas para Passeriformes machos, corresponde a um valor no espaço unidimensional de cantos que representa o canto do macho. Esse valor sempre se encontra dentro do *Intervalos de Canto Possíveis e Reconhecidos*, mesmo após passar pelo refino da aprendizagem socialmente mediada no caso dos Oscines, pois, como visto no modelo do padrão auditivo do desenvolvimento do canto descrito na seção 2.2.2, os Oscines não aprendem qualquer som que escutam, mas apenas aqueles filtrados pelo padrão bruto de cantos da espécie. O *Valor de Canto* está representado em vermelho na figura 4.1;

- *Tempo de Cristalização:* Apenas para os Oscines, é o tempo em meses no qual um Passeriforme virtual pode modificar seu canto através de aprendizagem socialmente mediada. Quando o *Contador de Meses* de um agente ultrapassa o se iguala ao *Tempo de Cristalização* em um agente filhote, a variável *Estágio da Vida* muda de **filhote** para **adulto**, e o agente não é mais capaz de modificar o seu *Valor de Canto*.

Escalas

- **Escala espacial:** Interações locais de curto alcance (distancia definida pelo *Raio de Audição* e pelo *Raio de Competição* dos agentes) em um mundo finito homogêneo e isotrópico com

condições de contorno periódicas de topologia toroidal e tamanho normalizado no modelo de mundo do agente, ou seja, as coordenadas *X* e *Y* que codificam a *Posição* dos agentes no mundo assumem valores dentro do intervalo $[0; 1[\subset R$;

- **Escala temporal:** Tempo discreto contabilizado em anos, onde cada ano é composto por 12 iterações assíncronas, com cada uma delas correspondendo a um mês real, sendo um ano composto por 11 meses regulares e um 12º mês de acasalamento;

4.2.3 Visão geral dos processos e sequência

O modelo simula a dinâmica social e de movimento espacial, que ocorre em uma escala de tempo ecológica, através de processos que atuam nas iterações correspondentes a um mês regular. Já a dinâmica evolutiva resulta das iterações correspondentes as temporadas de acasalamento, onde os processos responsáveis pela a dinâmica de nascimento e morte alteram ano após ano as frequências genotípicas na população através da seleção sexual e da deriva gênica.

Processos

- **Crescer** (*mês regular*)**:** os filhotes machos que atingem um *Tempo de Cristalização* (em meses) se tornam adultos e cristalizam seu canto;

- **Aprender** (*mês regular*) (*apenas Oscines*)**:** Oscines machos filhotes escutam os *Valores de Canto* dos machos adultos em sua vizinhança, identificam a moda, e aprendem substituindo seu *Valore de Canto* atual por um outro valor que é a média aritmética entre o valor atual e a moda;

- **Andar** (*mês regular*)**:** todos os agentes se movem com movimento browniano correlacionado com passo de tamanho variável (que é maior em média no primeiro ano de vida), com exceção dos seguintes casos: os Passeriformes virtuais que possuem um parceiro permanecem parados durante todo o ano, simulando uma situação de "ninho"; todos os filhotes permanecem parados até atingirem um *Tempo de Ninho*, contabilizado em meses;

- **Morrer** (*mês de acasalamento*)**:** os agentes morrem no mês de acasalamento em duas situações diferentes: qualquer agente morre caso atinja a *Idade Máxima*; todos os demais agentes na simulação, que não morreram em decorrência da idade avançada nesse mês de acasalamento, possuem uma probabilidade de morrer proporcional a quantidade de outros agentes vivos dentro do seu *Raio de Competição* (simulando os efeitos de uma competição intraespecífica);

- **Escolher parceiro** (*mês de acasalamento*)**:** as fêmeas adultas escutam o *Valor de Canto* dos machos adultos em sua vizinhança, determinam quais deles pertencem a mesma espécie que elas (verificando se o *Valor de Canto* de cada um deles se encontra dentro do seu *Intervalo de Cantos Possíveis e Reconhecidos*), e sorteiam aleatoriamente dentre estes com qual acasalar;

- **Reproduzir** (*mês de acasalamento*)**:** na época de acasalamento os agentes adultos que possuem parceiro se reproduzem e tem um número aleatório de filhotes que não ultrapassa um valor *Máximo de Filhotes por Ninhada*;

Sequência

Em cada iteração referente aos meses regulares, todos os processos são executados um por um por cada agente, com atualização assíncrona, ou seja, o primeiro agente executa todos os processos, o segundo faz o mesmo e assim por diante até que todos tenham agido. A ordem de ação dos agentes é aleatória e sorteada no início de cada mês.

Já no mês de acasalamento cada um dos três processos ocorre com todos os agentes, também com iterações assíncronas, antes que um outro processo comece a ser executado, ou seja, o primeiro processo é feito por cada agente seguindo uma ordem aleatória sorteada no inicio do processo, seguido pelo segundo processo e pelo terceiro da mesma maneira.

As sequências de processos referentes aos meses regulares e de acasalamento estão descritas abaixo, e na figura 4.2 temos um fluxograma mostrando como as sequências dos meses regulares e do mês de acasalamento se interligam para compor um ano de simulação:

- **Sequência dos meses regulares:** crescer ⇒ aprender ⇒ andar
- **Sequência do mês de acasalamento:** morrer ⇒ escolher parceiro ⇒ reproduzir

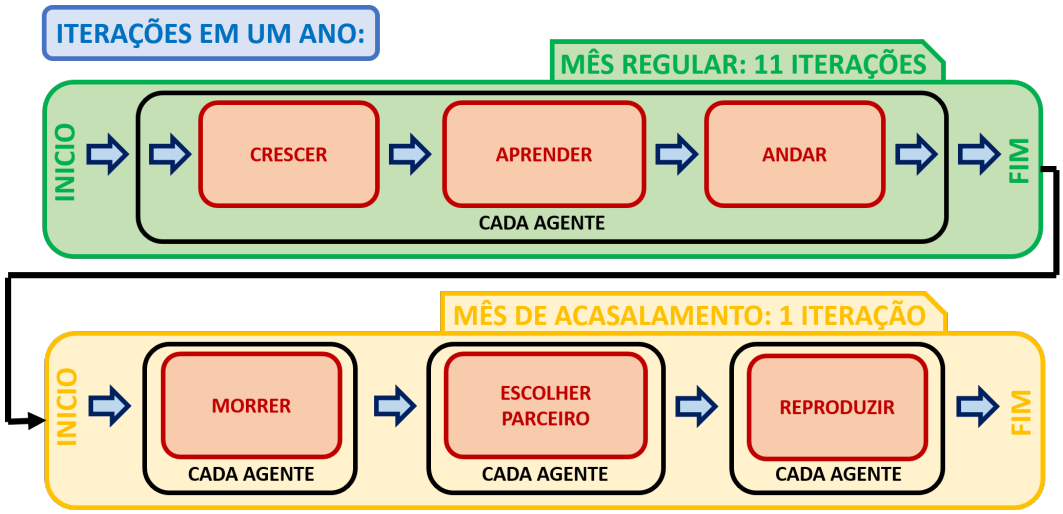


Fig. 4.2: Fluxograma representando como a composição de 12 iterações referentes a meses forma uma correspondente a um ano, sendo 11 iterações de meses regulares e uma de mês de acasalamento.

4.3 Implementação

O modelo foi implementado em C++ no Qt Creator (QT COMPANY LTD., 2019), utilizando o paradigma de programação orientada a objetos conforme descrito na subseção 3.1.3. Também foi desenvolvida uma interface simples através Qt Creator (QT COMPANY LTD., 2019) para facilitar a visualização do fenômeno e interpretação dos resultados (figura 4.3). Os dados gerados foram analisados utilizando o Python (PYTHON SOFTWARE FOUNDATION, 2018), em especial a biblioteca Igraph (THE IGRAPH CORE TEAM, 2014), usada para a construção e manipulação dos grafos.

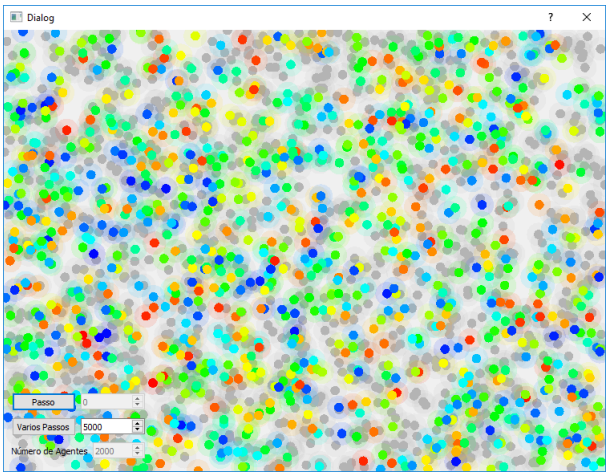


Fig. 4.3: Interface para visualização do modelo, desenvolvida no QT Creator (QT COMPANY LTD., 2019). Cada agente é representado por dois círculos, o menor de cor mais viva tem o tamanho dos *Raios de Audição* e o maior e quase transparente é do tamanho dos *Raios de Competição*. As fêmeas são cinzas e os machos estão coloridos com cores determinadas a partir do sistema HSV para refletir o *Valor de Canto*. O *Valor de Canto* mais baixo entre os machos vivos possui matiz 0 (vermelho) e o maior valor possui matiz 240 (azul). As demais matizes são calculadas de maneira proporcional.

Apesar do ambiente se tratar de um espaço 2D homogêneo e contínuo, o alto número de agentes fez com que fosse necessário utilizar uma estrutura de *patches* como descrito na subseção 3.1.3 para ambientes de topologia do tipo "sistema de informação geográfica". As interações do modelo são de curto alcance, assim é necessário medir a menor distância entre quaisquer dois agentes para determinar se eles podiam interagir ou não, o que se feito uma vez por iteração para cada combinação possível de dois agentes eleva muito o tempo de execução. Utilizando os *patches*, a distância é calculada apenas entre agentes dos *patches* dentro dos raios de interação, o que diminui consideravelmente o tempo de execução.

4.4 Cenários Simulados e Medidas

Foram executadas três séries de simulações para realizar diferentes estudos sobre sistema. Na primeira série, foram feitas 50 simulações em dois cenário, um com Oscines e outro com Suboscines, diferindo no *Aprendizado do Canto* e no *Tamanho do Intervalo de Cantos Possíveis e Reconhecidos*, conforme mostra a tabela 4.1. As medições foram realizadas desde a primeira iteração para captar o intervalo onde os *Alelos de Canto* se difundem pelo espaço de cantos a partir da sua distribuição inicial, dada pela Tabela 4.2. As simulações eram compostas de 20000 iterações (equivalentes a 20000 anos em nosso modelo). O objetivo dessa série era determinar as diferenças na velocidade e na intensidade da deriva gênica entre os dois cenários.

Na segunda série também foram realizadas 50 simulações para os mesmos dois cenários, um com Oscines e outro com Suboscines. A diferença é que nessa série foram realizadas 16000 interações sobre as quais não realizamos medidas, com o objetivo de desconsiderar os comportamentos transientes presentes no inicio da simulação referentes ao número de agentes, distribuição espacial e desvio

Parâmetro	Símbolo	Valor	
*Aprendizado do Canto**	A	Presente | Ausente	
Estagio de Vida	E_v	Adulto	
Tempo de Cristalização do Canto	T_c	6	meses
Tempo de Ninho	T_n	3	meses
Idade Máxima	I_m	8	anos
Máximo de Filhotes por Ninhada	F	7	
*Tamanho do Intervalo de Cantos Possíveis e Reconhecidos**	I_c	1 | 0.2	
Passo dos Agentes	P	0.0015	
Ângulo de Movimento	θ	140	graus
Proporção do Passo de Migração	P_f	2	
Ângulo de Movimento de Migração	θ_f	80	graus
Probabilidade de Acasalamento	P_a	0.435	
Probabilidade de Mutação	M_p	0.01	
Tamanho da Mutação	M_t	0.1	
Raio de Audição	R_a	6	
Raio de Competição	R_c	15	
Probabilidade de Morte	P_m	0.025	
Quantidade Inicial de Passeríformes	N_i	2000	

Tab. 4.1: Valores dos parâmetros de entrada utilizados. Todas são variáveis dos agentes, com exceção da *Quantidade Inicial de Passeriformes*, que é utilizada pelo ambiente no momento da sua construção. As variáveis marcadas com asterisco possuem dois valores pois foram adotados valores diferentes para cenários diferentes, e estão representadas na tabela na forma **Oscines | Suboscines**.

padrão da distribuição de alelos, captados na primeira série de simulações. As medições foram então efetuadas durante 20000 iterações, totalizando em simulações com 36000 iterações de um ano. O objetivo dessa segunda série era avaliar a modularidade das redes de reconhecimento para acasalamento entre agentes.

Na terceira série de simulações também não realizamos medidas sobre as 10000 iterações iniciais, a fim de desconsiderar o comportamento transiente presente no inicio das simulações. Foram realizadas 15 simulações para 4 cenário diferentes, todos com Oscines mas com diferentes tamanhos de mundo. As medições foram efetuadas por um período de 10000 iterações de um ano, totalizando 20000 iterações. O objetivo dessa terceira série era avaliar como o tamanho do mundo afeta a distribuição da modularidade das redes de reconhecimento para acasalamento. O motivo desse estudo ter sido efetuado apenas com Oscines é que para os parâmetros utilizados as populações de Suboscines virtuais se extinguiam com facilidade para tamanhos de mundo menores que os das duas primeiras séries de simulações.

A única diferença entre as simulações de um mesmo cenário, em uma mesma série, é a semente utilizada no gerador de números pseudo-aleatórios. Caso alguma das simulações em qualquer série apresentasse a extinção da população antes do período estipulado, esta era desprezada e uma outra era feita em seu lugar com uma nova semente (ocorreram apenas 3 casos com Suboscines nas duas primeiras séries, e 4 casos com Pscines para o menor tamanho de mundo na última série).

Os valores dos parâmetros utilizados estão registrados na tabela 4.1. Algumas variáveis internas dos agentes foram inicializadas com valores aleatórios, sendo esses valores fornecidos através de distribuições de probabilidades planas sobre os conjuntos descritos na tabela 4.2. Dentre essas variáveis internas temos a *Posição* (X, Y) e a *Idade*, que não representam diretamente características fenotípicas, e as variáveis que representa características que sofrem seleção sexual no modelo, o *Valor do Canto* e os dois *Alelos de Canto*, que são usados para calcular o *Inicio do Intervalo de Cantos Possíveis e Reconhecidos* (I_C) através de uma média aritmética $(I_C = (A_1 + A_2)/2))$. Já o sexo dos agentes na inicialização não foi determinado aleatoriamente, a primeira metade foi inicializada como macho, e a segunda metade como fêmea.

Variável do Agente	Símbolo	Intervalo de Sorteio
Posição (valor da coordenada X)	X	$[0; 1[\subset R$
Posição (valor da coordenada Y)	Y	$[0; 1[\subset R$
Idade	I	$[1, I_m - 1] \subset N$
Alelo 1	A_1	$[(2.5 - M_t/2); (2.5 + M_t/2)] \subset R$
Alelo 2	A_2	$[(2.5 - M_t/2); (2.5 + M_t/2)] \subset R$
Valor de Canto	V_c	$[I_n; I_n + I_c] \subset R$

Tab. 4.2: Intervalos sobre os quais ocorrem o sorteio dos valores iniciais de algumas variáveis internas dos agentes, seguindo distribuições planas.

Ao fim de cada iteração de um ano, armazenamos as seguintes informações para cada agente: *ID*, *Sexo*, *Posição* (valores normalizados de X e Y), *Inicio do Intervalo de Cantos Possíveis e Reconhecidos*, *Valor de Canto*, *Alelos de Canto* e *Idade*. Essas informações são armazenada em uma tabela diferente para cada ano, sinalizadas com o valor correspondente a iteração. Além disso, o número de agentes e a variabilidade na distribuição espacial de agentes (que chamaremos apenas de variabilidade espacial) são armazenados em cada iteração, sendo a variabilidade espacial calculada da seguinte maneira: o espaço é dividido regularmente em um número regiões quadradas R_s (49 em nossa análise) de mesmo lado L, e então a variabilidade espacial V_e é dada pelo calculo do desvio relativo do número de agentes N_R com relação a esses quadrados através da seguinte equação:

$$V_e = \frac{\sqrt{\left(\frac{1}{49}\sum_{R_s} N_R^2\right) - \left(\frac{1}{49}\sum_{R_s} N_R\right)^2}}{\left(\frac{1}{49}\sum_{R_s} N_R\right)} \tag{4.1}$$

Através dos dados armazenados montamos as redes de reconhecimento para acasalamento (PATERSON; MCEVEY, 1993), onde buscamos medir a formação de dialetos através do calculo da modularidade, e a especiação pela contagem do número de componentes. Além disso analisamos o efeito da deriva gênica sobre as populações avaliando a largura da distribuição dos alelos $A_{i,j}$, onde $i = 1, 2$ se refere a cada alelo e $j = 1, 2, ..., N$ se refere a cada agente, através do desvio padrão σ_A da distribuição dos alelos em cada intervalo de tempo, calculado através da seguinte equação:

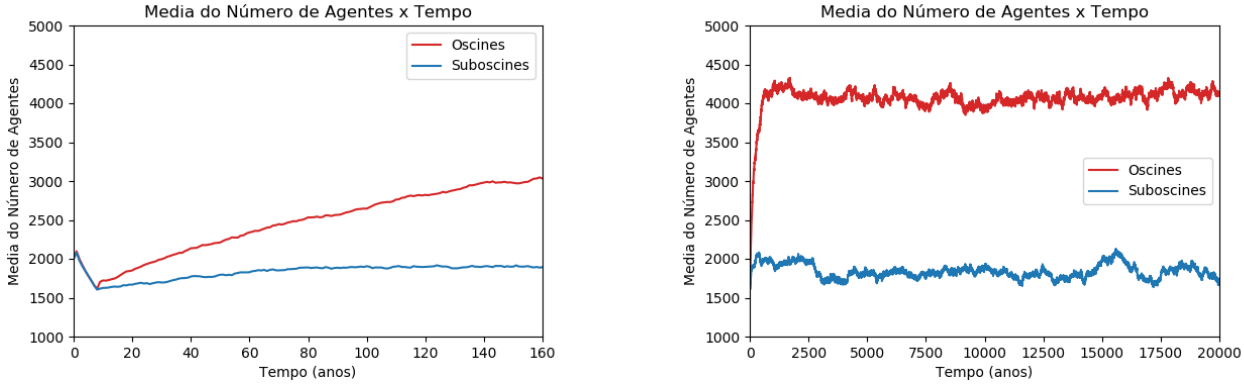
$$\sigma_A = \sqrt{\left(\frac{1}{2N}\sum_{i=1}^{2}\sum_{j=1}^{N} A_{i,j}^2\right) - \left(\frac{1}{2N}\sum_{i=1}^{2}\sum_{j=1}^{N} A_{i,j}\right)^2} \tag{4.2}$$

4.5 Resultados e Conclusões

4.5.1 Número de Indivíduos e Variabilidade Espacial

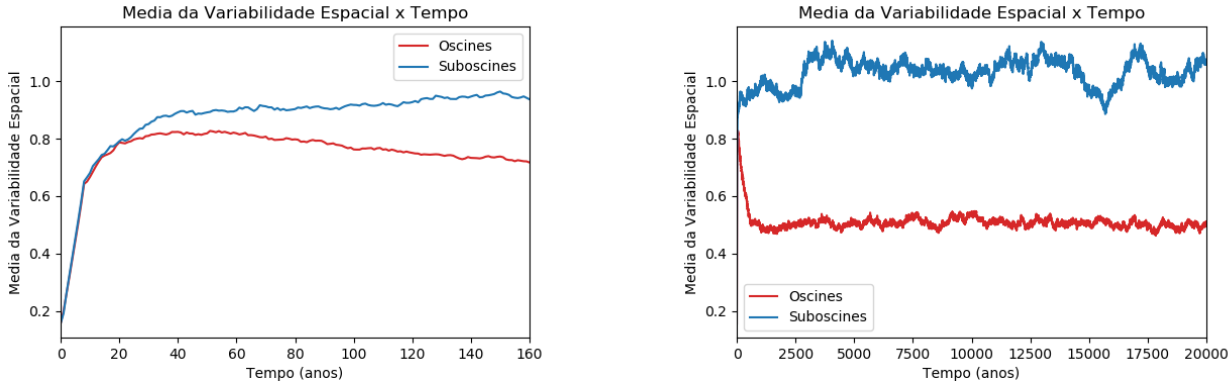
Com dados da primeira série de simulações construímos gráficos da média do número de agentes por tempo, figuras 4.4, e da média da variabilidade espacial por tempo, figuras 4.5. Individualmente as simulações apresentam grandes flutuações para essas duas grandezas, como podemos ver nos itens (a) e (b) da figura 4.6, porém o comportamento médio delas converge para uma situação de estabilidade em cerca de 1000 iterações, e nos apresenta algumas diferenças claras entre a dinâmica dos dois cenários.

Quando se estabiliza, a média do número de Oscines nas simulações é aproximadamente o dobro da média do número de Suboscines. Em nosso modelo, a mortalidade não depende diretamente do



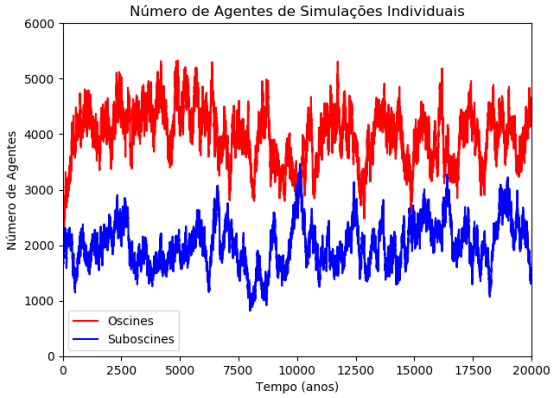
(a) Anos iniciais da simulação (até o ano 160). (b) Simulação completa.

Fig. 4.4: Média do número de agentes sobre 50 simulações $\langle N \rangle_{50sim}$ por tempo para Oscines e Suboscines. Foram utilizados dados gerados na primeira série de simulações.

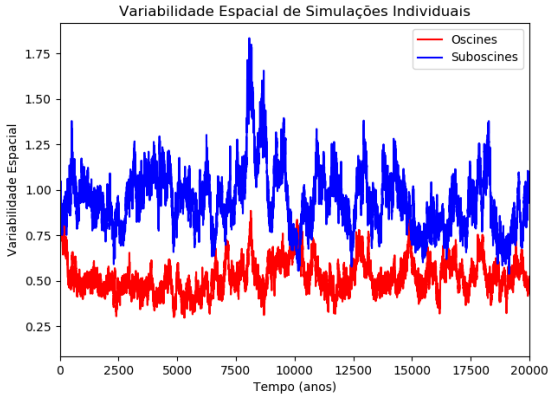


(a) Anos iniciais da simulação (até o ano 160). (b) Simulação completa.

Fig. 4.5: Média da variabilidade espacial sobre 50 simulações $\langle V_e \rangle_{50sim}$ por tempo, V_e calculada para cada instante de tempo em cada simulação através da equação 4.1, para Oscines e Suboscines. Foram utilizados dados gerados na primeira série de simulações.



(a) Número de agentes N pelo tempo (em anos).



(b) Número de agentes V_e pelo tempo (em anos).

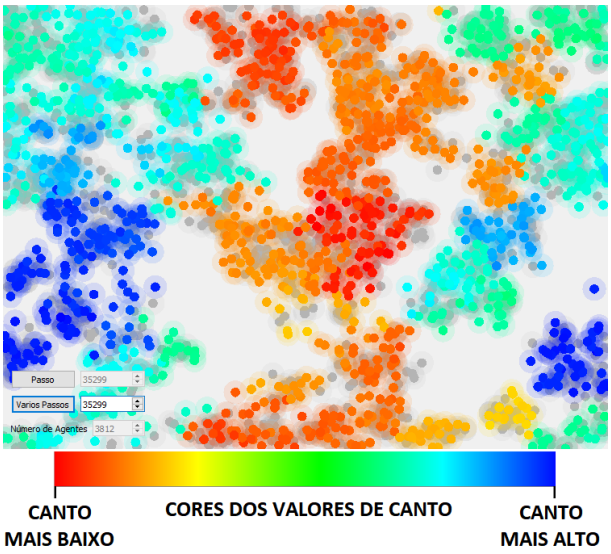
Fig. 4.6: Gráficos exibindo as flutuações no número de agentes N e na variabilidade espacial V_e (calculada através da equação 4.1) em simulações individuais da primeira série de simulações com Oscines e Suboscines.

Aprendizado do Canto e nem do *Tamanho do Intervalo de Cantos Possíveis e Reconhecidos*, que são as duas únicas diferenças entre os cenários na primeira série de simulações. Por outro lado, as probabilidades de acasalamento dependem, pois em nosso modelo é justamente o reconhecimento canto que torna o acasalamento possível. Ter um *Tamanho do Intervalo de Cantos Possíveis e Reconhecidos* maior facilita o acasalamento pois aumenta a probabilidade de reconhecimento, assim como o *Aprendizado do Canto* também facilita, pois geralmente na época que ele acontece os filhotes ainda estão próximos ao pai, que possui um canto que foi bem sucedido em encontrar uma parceira.

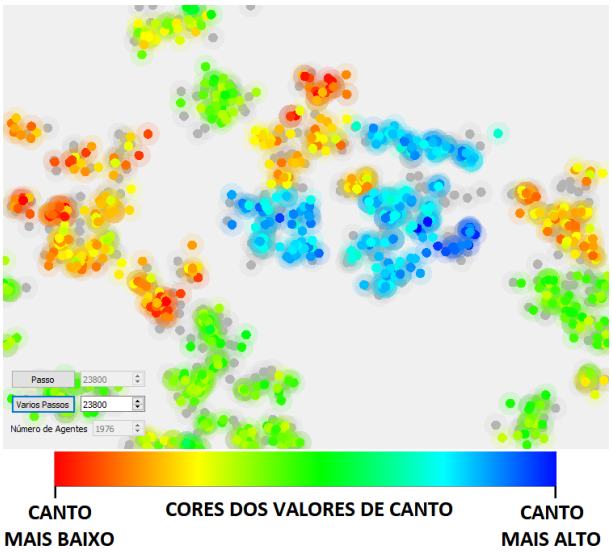
Já a média da variabilidade espacial, ao atingir um estado estacionário, é cerca do dobro no cenário com Suboscines em relação ao cenário com Oscines. Isso fica visível na diferença entre as distribuições espaciais do item (a) com Oscines e do item (b) com Suboscines nas figuras 4.7.

Também há um comportamento interessante nas primeiras iterações tanto na a média no número de agentes quanto na a média da variabilidade espacial, como podemos ver nos itens (a) das figuras 4.4 e 4.5. Nessas primeiras iterações ambos os cenários apresentam um comportamento praticamente idêntico, de uma queda na média de indivíduos e uma subida na média da variabilidade espacial. Isso acontece porque no início da simulação os agentes são distribuídos uniformemente, o que torna a densidade de Passeriformes muito baixa. Assim, como os agentes se encontram distantes uns dos outros, muitos morrem sem reproduzir, o que justifica a queda no número de indivíduos e a diferença de comportamento quase nula entre os dois cenários, pois as diferenças entre eles impactam apenas sobre o acasalamento, o mecanismo responsável pela mortalidade é idêntico em ambos. Além disso como a distribuição é inicialmente uniforme a variabilidade espacial é muito baixa, mas ao morrerem esses agentes deixam regiões do ambiente vazias, o que eleva a média da variabilidade espacial nos momentos iniciais para os dois cenários.

Esse regime dura até aproximadamente 8 iterações, que é a idade máxima dos agentes. Após 8 iterações, todos os Passeriformes presentes no inicio da simulação já morreram, e os agentes vivos são frutos dos casais formados nas iterações iniciais. Esses agentes já nascem próximos a outros, e assim tem uma maior chance de acasalar. Como a taxa de acasalamento aumenta, as diferenças



(a) Oscines (semente = 50, ano = 35299, nº de componentes = 1, nº de comunidades = 2, modularidade = 0.457).



(b) Suboscines (semente = 46, ano = 23800, nº de componentes = 2, nº de comunidades = 3, modularidade = 0.612).

Fig. 4.7: Distribuição espacial dos agentes em anos de alta modularidade de simulações da segunda série de simulações. O ano e a semente estão especificados, assim como algumas propriedades estruturais calculadas para a rede de reconhecimento para acasalamento. As fêmeas aparecem em cinza e machos estão coloridos conforme os valores de canto.

entre Oscines e Suboscines começa a aparecer. Para o cenário dos Suboscines, como o número de indivíduos não aumenta muito além do valor inicial, a variabilidade espacial sobe mais um pouco e se estabiliza. Já os Oscines passam por um período de crescimento populacional, até se estabilizarem por volta da 1000^{a} iteração, um processo que acontece de forma sincronizada com uma queda na variabilidade espacial.

As distrbuições espaciais da figura 4.7, assim como essa relação inversamente proporcional entre média do número de agentes e média na variabilidade espacial, indicam que em ambos os cenários os aglomerados de agentes tem densidades semelhantes, porém nas simulações com Oscines os aglomerados se espalham e ocupam espaços vazios, o que permite o aumento do número de agentes e a diminuição da variabilidade espacial mantendo a densidade constante. Já nas simulações com Suboscines, como não há um grande aumento populacional, os aglomerados permanecem com o mesmo tamanho, deixando espaços vazios, como podemos ver no item (b) da figura 4.7, o que mantém a variabilidade espacial mais alta.

4.5.2 Deriva Gênica

Para análise da deriva gênica foram usadas as medidas feitas na primeira série de simulações, tomadas desde o inicio das simulação, o que inclui regime não estacionário. Montamos o gráfico 4.8, que mostra a média do desvio padrão das distribuições de *Alelos de Canto* σ_A sobre 50 simulações pelo tempo, calculados através da equação 4.2, tanto no cenário com Oscines (em vermelho) quanto no com Suboscines (em azul).

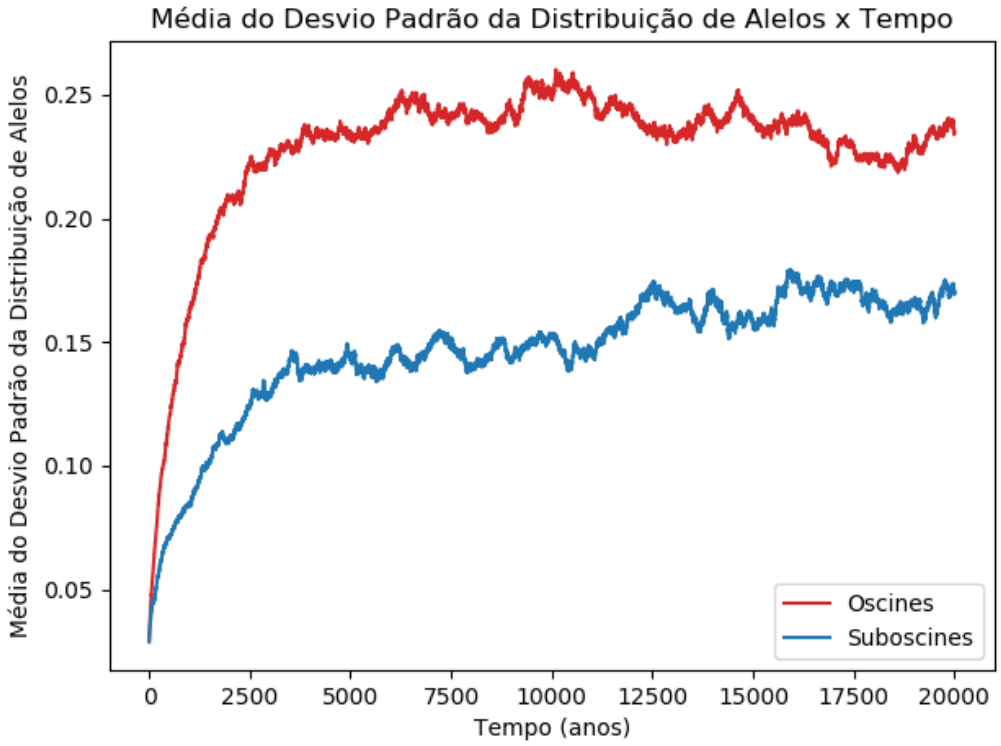


Fig. 4.8: Gráfico para a análise do comportamento da deriva gênica. Média calculada sobre o desvio padrão das distribuições de *Alelos de Canto* de 50 simulações $\langle \sigma_A \rangle_{50sim}$ por tempo, sendo σ_A calculada para cada instante de tempo em cada simulação através da equação 4.2, para Oscines e Suboscines. Dados provenientes da primeira série de simulações.

O desvio padrão mede a largura de uma distribuição. Em nossas simulações a média sobre o desvio padrão da distribuição de *Alelos de Canto* cresce mais rapidamente e se mantem maior no cenário **com Oscines** para todo intervalo de tempo, o que mostra que a distribuição se alarga mais rápido e se mantem mais larga.

Ambos os cenários apresentam as mesma *Probabilidade de Mutação* (M_p) e os mesmo *Tamanhos da mutação* (M_t), logo devem experimentar efeitos semelhantes de deriva. O que acontece aqui é que, com a difusão dos *Alelos de Canto*, a **seleção sexual**, que está atuando nesse modelo como seleção estabilizadora, tem um efeito muito mais contundente nos Suboscines que nos Oscines.

Os agentes com genótipos exóticos com relação a sua comunidade possuem em média menor aptidão, pois tendem a gerar cantos distantes da média, e assim são reconhecido por um número menor de fêmeas. Porém esse efeito é menor no caso dos Oscines. Quando uma mutação ou recombinação de alelos faz com que um agente Oscine macho tenha um *Intervalo de Cantos Possíveis e Reconhecidos* deslocado com relação a sua comunidade, mas que ainda possua alguma intersecção com ela, ele consegue aprender o canto com machos próximos, e assim se reproduzir e propagar seu genótipo. Já para agentes Suboscines com genótipos distantes da média, a redução da aptidão é mais expressiva. Como seu canto é sorteado aleatoriamente dentro do próprio *Intervalo de Cantos Possíveis e Reconhecidos*,

eles contam com uma maior probabilidade de não serem reconhecidos pela maioria das fêmeas, um efeito que é ainda amplificado pelo *Tamanho do Intervalo de Cantos Possíveis e Reconhecidos*, que é mais curto no cenário com Suboscines. Dessa maneira alelos exóticos tem uma maior dificuldade de difundir no espaço de cantos nas simulações com Suboscines.

Esse resultado está de acordo com o encontrado por Lachlan e Servedio (2004) em um modelo de genética populacional, de que aprendizado socialmente mediado permite uma maior divergência em Oscines por "mascarar"genótipos desvantajosos, assim como está de acordo com previsões teóricas que indicam que a aprendizagem em sistemas com seleção sexual estabilizadora pode favorecer a divergência por deriva gênica, o que pode acarretar em especiação (VERZIJDEN et al., 2012).

4.5.3 Formação de Dialetos

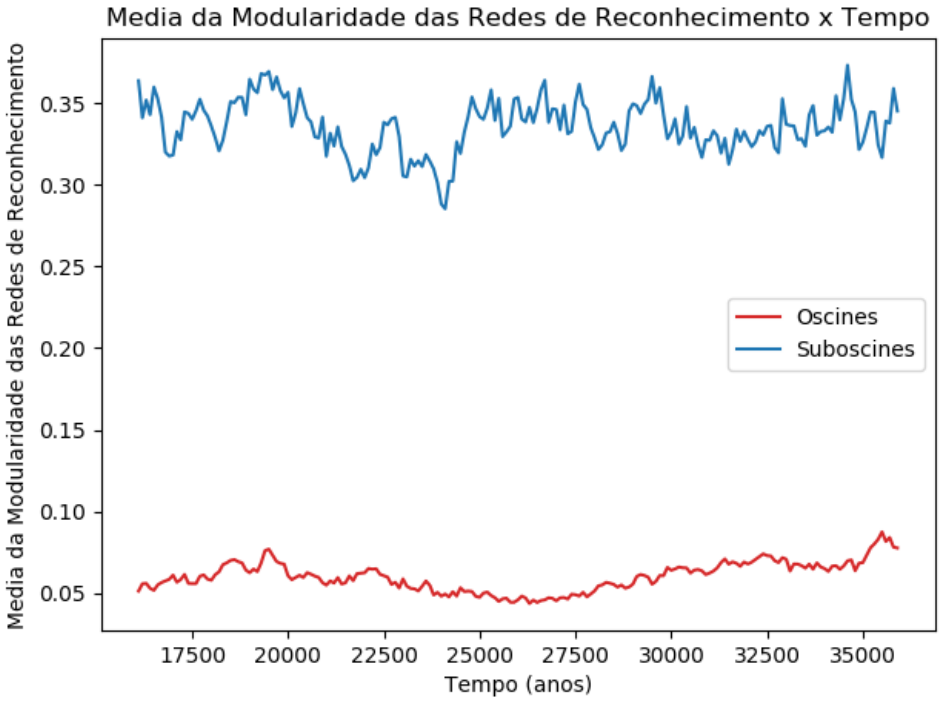


Fig. 4.9: Gráfico para a análise da formação de dialetos nas simulações. Média sobre a modularidade das redes de reconhecimento de 50 simulações $\langle Q_i \rangle_{50sim}$ por tempo, sendo $Q_i = \langle Q \rangle_{10t}$ uma média calculada sobre intervalos de 10 iterações seguidas em uma simulação, para Oscines e Suboscines. Dados provenientes da segunda série de simulações.

Para a analise da formação de dialetos foram usadas as medidas feitas na segunda leva de simulações, tomadas apenas a partir da 16.000ª geração, para tomar medidas em um regime estacionário. Foram geradas redes de reconhecimento em conjuntos de 10 gerações consecutivas intercaladas com intervalos de 90 gerações (por uma questão de tempo de execução da análise). Nestas redes os nós representavam Passeriformes machos e fêmeas, e as arestas ligavam fêmeas com machos cujo *Valor*

De Canto estava dentro de seu *Intervalo de Cantos Possíveis e Reconhecidos*, ou em outras palavras, as arestas designavam acasalamentos possíveis pelo sistema de reconhecimento para acasalamento (PATERSON; MCEVEY, 1993), independente da distância física entre os agentes. A modularidade foi calculada sobre essas redes usando as comunidades encontradas através do método de Louvain (BLONDEL et al., 2008). A média da modularidade para cada conjunto de 10 gerações era então armazenada para a construção do gráfico 4.9.

A média da modularidade nas simulações com Oscines se manteve baixa em todos os instante de tempo. Individualmente esse comportamento é esperado, pois a formação de comunidades depende da difusão dos *Alelos de Canto*, e não há nenhuma força evolutiva em nosso modelo além da deriva impulsionando essa difusão. Além disso, apesar do comportamento estável das médias, individualmente as simulações em ambos os cenários não possuem estabilidade com relação aos valores de modularidade pelo tempo, conforme podemos ver no gráfico 4.10. Esta instabilidade se dá porque não existe nenhuma barreiras além da distância física que impeça o fluxo gênico e cultural entre as comunidades. Assim mesmo havendo formação de dialetos em alguns momentos, eles podem convergir para um mesma região do espaço e se desfazer, causando quedas no valor de modularidade da rede de reconhecimento. Além disso é possível visualizar, através da interface gráfica, aglomerados de agentes se extinguindo devido a competição intraespecífica. Caso esses aglomerados correspondam as comunidades das redes de conhecimento, o que veremos na sessão 4.5.4, esse fenômeno também pode causar variações bruscas no valor de modularidade.

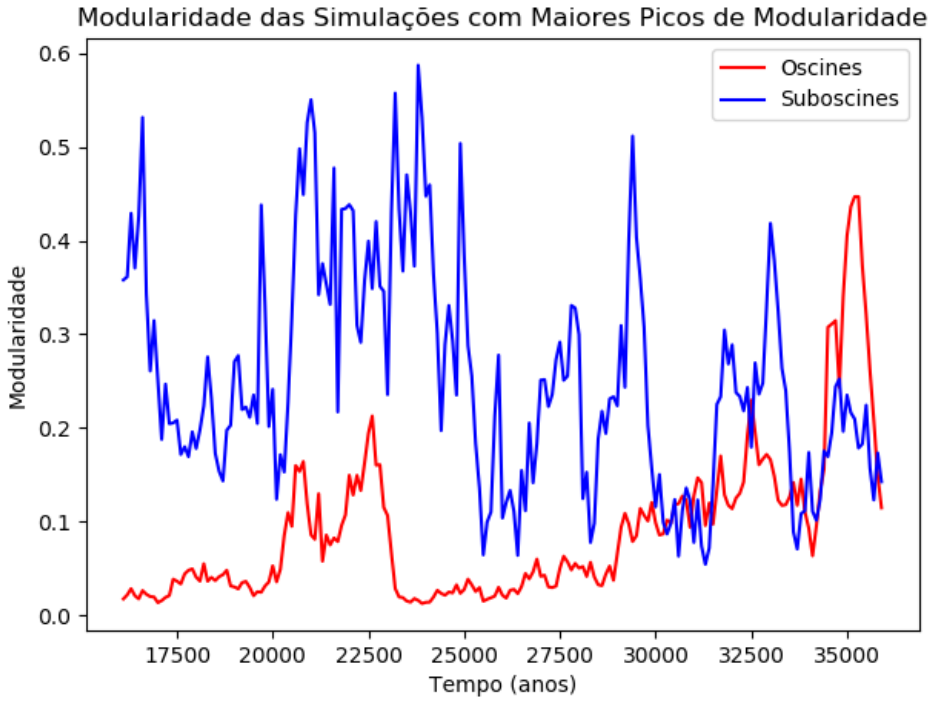


Fig. 4.10: Gráfico com a evolução temporal da modularidade nas simulações individuais que desenvolveram os picos mais altos de modularidade para Oscines e Suboscines. Dados provenientes da segunda série de simulações.

Apesar da média dos valores de modularidade ser baixa para todos os instantes de tempo, as simulações com Oscines também desenvolveram estruturas de comunidades em alguns instantes, como podemos ver na simulação representada na figura 4.10. Ao todo, 20% delas apresentaram modularidade superior a 0.3 em algum instante de tempo, conforme podemos ver na distribuição dos picos de modularidade das simulações com Oscines na figura 4.11.

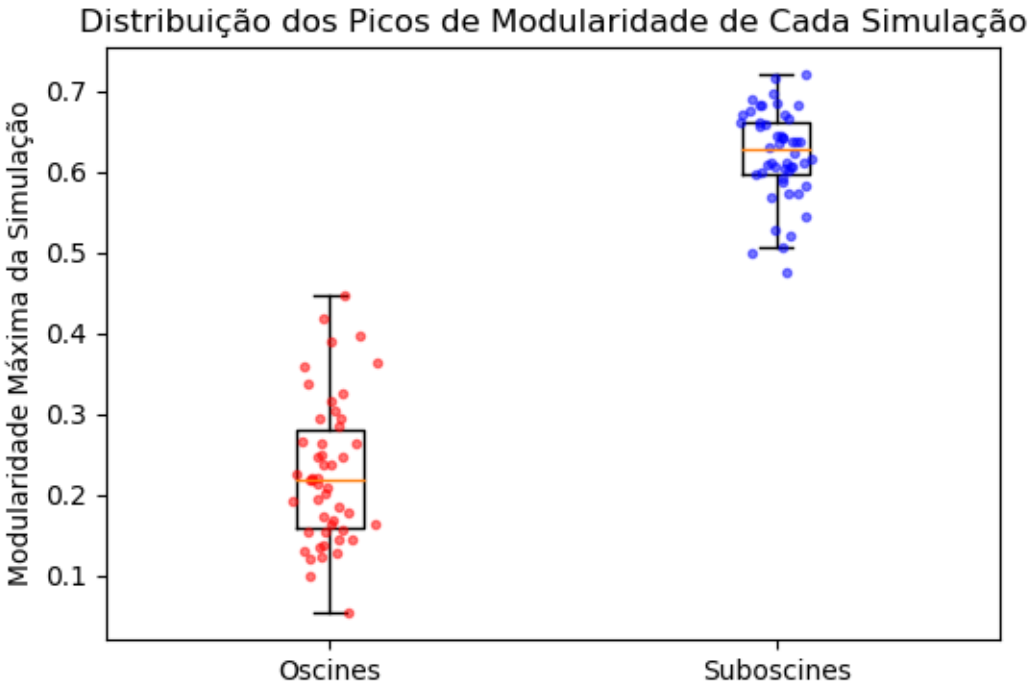


Fig. 4.11: Distribuição dos valores da modularidade dos picos de modularidade de cada simulação para Oscines e Suboscines. Dados provenientes da segunda série de simulações.

Já a média da modularidade das redes de reconhecimento de Suboscines possuem um valor maior que 0.3 para quase todos os instantes de tempo, o que indica que em média essas simulações geraram redes com uma forte estrutura de comunidades. Isso é surpreendente considerando que assim como para os Oscines não nenhum mecanismo que impulsione a difusão de *Alelos de Canto* além da deriva gênica, e eles estão sujeitos as mesmas condições que geram instabilidade sobre a modularidade. Além disso na simulação com Oscines os dialetos são produto tanto de uma evolução genética quanto de uma evolução cultural, sendo a segunda tipicamente mais rápida que que a primeira (VERZIJDEN et al., 2012). Porém 3 fatores são determinantes na formação da estrutura de comunidades nas simulações com Suboscines seja mais expressiva que nas com Oscines:

1. Como o espaço é homogêneo, não há nele seleção ecológica adaptando o canto a regiões diferentes, caso houvesse, isso levaria os agentes Oscines e Suboscines de regiões diferentes a desenvolverem dialetos próprios, e sendo evolução cultural em geral mais rápida que evolução biológica (CHEBIB; MARRIOTT, 2016), o cenário com Oscines poderia ter um desen-

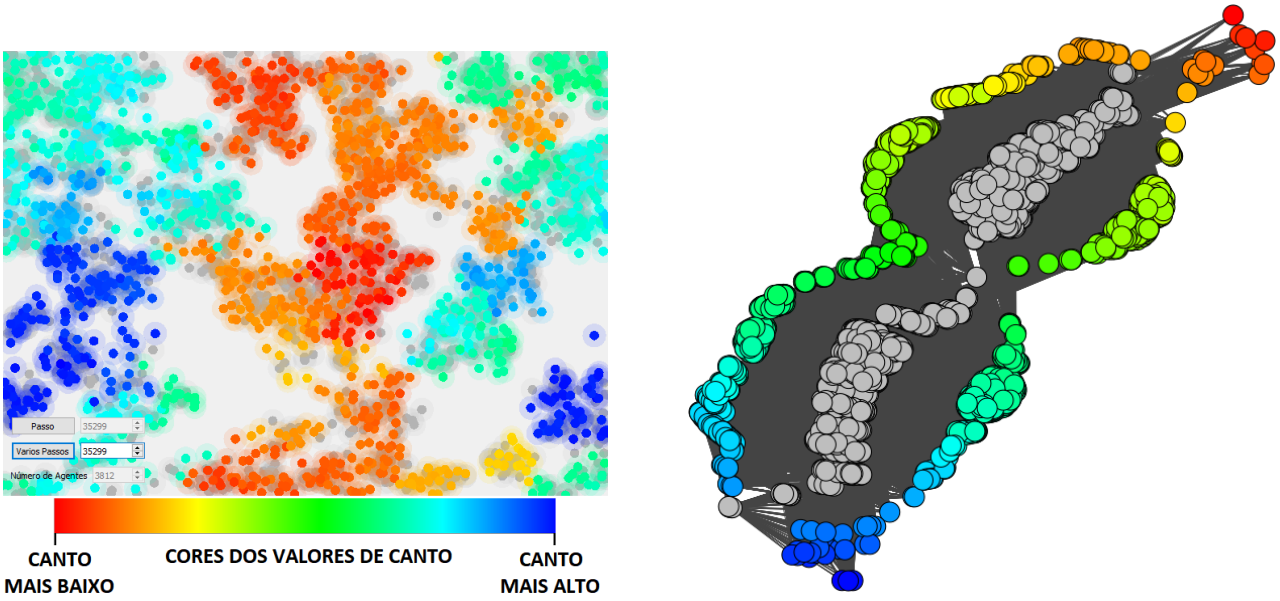
volvimento mais rápido de dialetos e consequentemente formar redes de reconhecimento com estrutura de comunidades mais rapidamente;

2. O aprendizado mascara genótipos desvantajosos, conforme visto na subseção 4.5.2, assim esses agentes que em um cenário sem aprendizagem se restringiriam a se conectar na rede de reconhecimento com uma comunidade menor de fêmeas, também com genótipos exóticos, acabam aprendendo um canto que lhes permitem se conectar com nós por toda a rede, diminuindo a modularidade;

3. A única barreira ao que impede o fluxo genético e cultural (esse último apenas para Oscines) é a distância, e ela é muito mais efetiva nas simulações com Suboscines, pois esses apresentam em média uma maior variabilidade espacial, conforme podemos ver no gráfico 4.5;

4.5.4 Especiação

Parte da proposta desse modelo é comparar o tempo de especiação dos dois cenários, usando como critério o número de componentes da rede de reconhecimento. Entretanto, apesar de fazer sentido, esse critério não se mostrou útil para nosso modelo, pois qualquer macho com um valor de canto exótico o suficiente para não acasalar com nenhuma fêmea já era considerado um componente a parte. Em todas as simulações há ao menos um momento onde o número de componentes é maior que 2, mas não necessariamente isso corresponde a uma especiação.



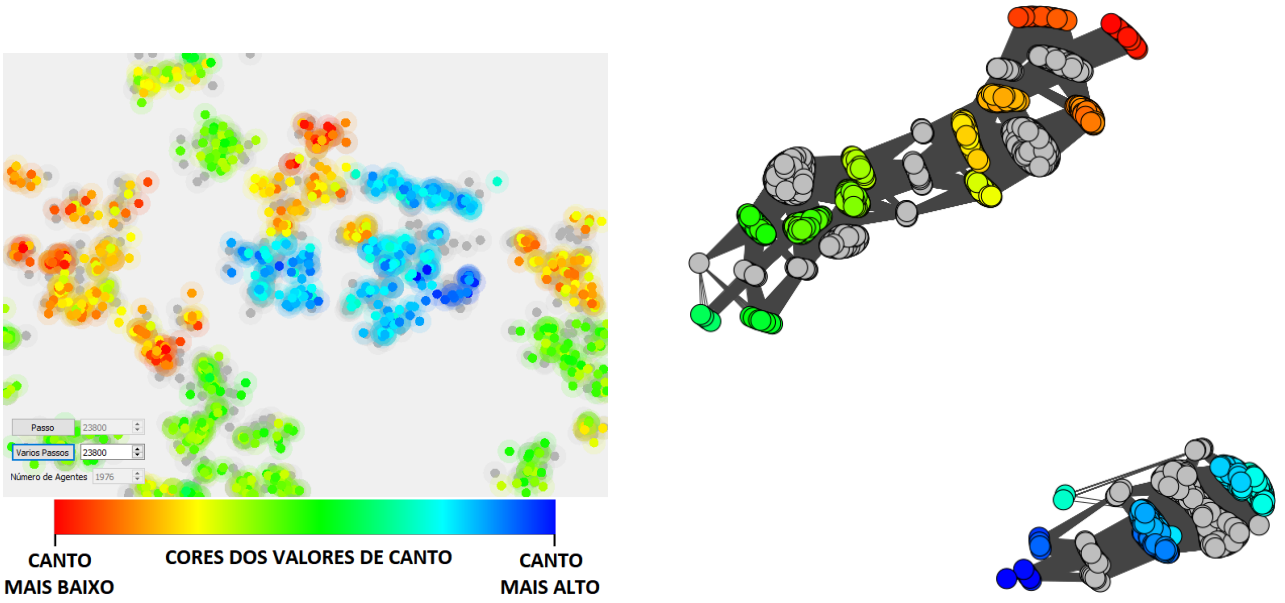
(a) Distribuição espacial dos agentes.

(b) Grafo da rede de reconhecimento para acasalamento.

Fig. 4.12: Caracterização de um pico de modularidade em uma simulação com Oscines, com fêmeas em cinza e machos coloridos conforme os valores de canto. (semente = 50, ano = 35299, nº de componentes = 1, nº de comunidades = 2, modularidade = 0.457)

No cenário com Oscines não encontramos nada parecido com uma especiação. A figura 4.12 mostra o momento de maior modularidade entre entre as simulações com Oscines e apesar dos dialetos serem claramente visíveis na distribuição espacial do item (a), a rede permanece conectada.

Já no caso dos Suboscines, ao investigarmos os picos de modularidade das simulações, encontramos algumas nas quais a rede não está conectada e as componentes possuem mais de um indivíduo, que poderiam representar uma especiação, como o pico de modularidade representado na figura 4.13. Ao todo encontramos redes desconectadas semelhantes nos picos de modularidade de 24% das simulações com Suboscines, o que juntamente com os resultados da seção 4.5.3 sobre formação de dialetos indicam que os Suboscines tem uma maior tendência a especiação no caso de um cenário homogêneo desprovido de barreiras geográficas.



(a) Distribuição espacial dos agentes

(b) Grafo da rede de reconhecimento para acasalamento.

Fig. 4.13: Caracterização de um pico de modularidade em uma simulação com Suboscines, com fêmeas em cinza e machos coloridos conforme os valores de canto. (semente = 46, ano = 23800, nº de componentes = 2, nº de comunidades = 3, modularidade = 0.612)

Analisando a distribuição espacial dos cantos em picos de modularidade das simulações de ambos os cenários, onde a formação de dialetos é mais clara, podemos notar que populações com dialetos diferentes podem até fazer fronteira, mas não se sobrepõem (exemplos no item (a) da figura 4.12 para Oscines e da figura 4.13 para Suboscines). Como agentes com cantos semelhantes acasalam com grupos semelhantes de fêmeas, provavelmente há alguma correspondência entre os aglomerados de agentes visíveis na distribuição espacial e as comunidades encontradas nas redes de reconhecimento para acasalamento. A correspondência não é completa, pois temos aglomerados diferentes que compartilham o mesmo dialeto, mas oferece um indicativo de que a distribuição espacial tem um papel importante na formação dos dialetos.

4.5.5 Efeito de Mundo Finito

Para testar a influência do tamanho do mundo na formação de dialetos utilizamos a terceira série de simulações, onde efetuamos medidas a partir da 10000ª geração em 4 cenários com tamanhos de mundo diferentes, todos com Oscines.

Como em nosso modelo todas as distâncias são expressas em termos de *Passos dos Agentes* (P), nós variamos o tamanho do mundo variando P, e expressamos o tamanho do mundo em termos do número de *Passo dos Agentes* necessários para dar uma volta completa no mundo na direção X ou na direção Y.

Armazenamos a média da modularidade de redes de reconhecimento construídas para 10 gerações consecutivas, intercaladas com intervalos de 90 gerações, assim como na subseção 4.5.3 sobre formação de dialetos. Com esses dados montamos o gráfico 4.14, com as distribuições de modularidade para cada tamanho de mundo diferente:

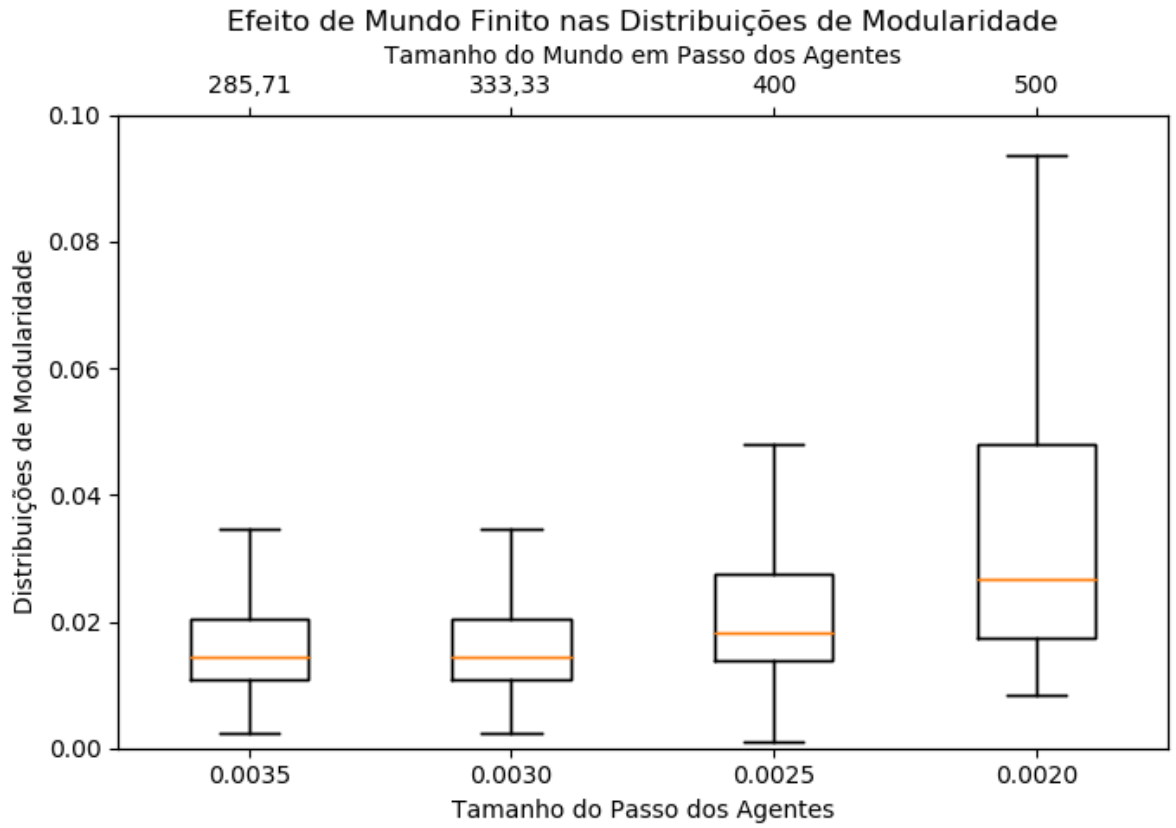


Fig. 4.14: Gráfico para ilustrar o efeito do mundo finito. Distribuições das modularidades para cenários de Oscines com diferentes tamanhos de mundo, expressos em termos do número de *Passo dos Agentes* necessários para dar uma volta completa no mundo na direção X ou Y. Dados provenientes da terceira série de simulações.

Nas distribuições com os dois maiores tamanhos de mundo, a distribuição de modularidades possui medias ligeiramente maiores e também contempla valores maiores de modularidade. Já os dois

menores tamanhos de mundo possuem comportamento muito semelhante, medias menores parecidas e com distribuições que contemplam apenas valores parecidos e menores de modularidade. Dada a aparente correspondência entre a distribuição espacial dos agentes e a formação de comunidades nas redes de reconhecimento para acasalamento, descrita na seção 4.5.4, o que pode estar acontecendo é que nos mundos menores a falta de espaço dificulta a formação de aglomerados e que eles se mantenham separados por muito tempo, o que dificulta a formação de comunidades.

Capítulo 5

Perspectivas Futuras

Apesar de reproduzir tanto resultados do modelo de genética populacional de Lachlan e Servedio (2004), quanto observações feitas por Freeman, Montgomery e Schluter (2017), o modelo ainda pode ser modificado para investigar outros cenários biologicamente interessantes, e melhorado para se tornar mais realista quanto a alguns aspectos relevantes para a dinâmica do sistema.

Algumas modificações que podem ser feitas são para analisar cenários com ambientes heterogêneos. O sistema de patches utilizado oferece uma base natural para essa modificação, pois na prática nosso ambiente homogêneo já é um sistema de informação geográfica com todos os patches iguais. Os cenários com ambientes heterogêneos que podem ser implementados são de dois tipos:

1. **Adição de gradientes espaciais:** adição de gradientes de grandezas que impactem sobre a dinâmica, como gradiente de ruido ou de recursos disponíveis. Um gradiente de ruido pode interferir na seleção da frequência ou da intensidade com que os passeriformes cantam, e esse impacto pode ser diferente na presença de aprendizado. Já um gradiente de recursos pode permitir trabalhar a questão da frequência do canto como um indicador de melhores territórios, conforme vimos na subseção 2.2.3.

2. **Adição de barreiras geográficas:** Adição de barreiras geográficas tornando alguns patches intransponíveis. Usando uma barreira para interromper o fluxo genético e cultural entre duas populações podemos avaliar melhor o caso de especiação alopátrica.

Dois aspectos do sistema podem ser melhorados para conferir maior realismo ao modelo. O primeiro se refere a representação do canto, cuja mudança se faz necessária para o estudo dos casos com gradientes espaciais. Em nosso modelo o canto e os alelos são representados apenas como números em um espaço unidimensional, mas esse número não mede qualidade ou representa caraterísticas físicas do canto, é usado apenas para o reconhecimento. A frequência, por exemplo, é uma característica física que certamente queremos representar, pois é fundamental no cenário com gradiente de ruído. Nosso modelo também não leva em conta quando e quanto um Passeriforme canta e o tamanho do seu repertório. Implementar a quantidade de canto é necessário para avaliar o canto como um indicador de qualidade de território e implementar o tamanho do repertório é relevante pois as fêmeas tendem a preferir machos com maiores repertórios (CATCHPOLE; SLATER, 2008).

O outro aspecto que pode ser melhorado é como as fêmeas se relacionam com o canto, que nesse trabalho se restringiu apenas ao seu reconhecimento ou não. Essa mudança se faz necessária

para que tenhamos uma reprodução mais precisa da seleção sexual e também nos permite realizar simulações onde as fêmeas tenham, dentre os cantos reconhecidos, preferências que sofram efeito de aprendizagem socialmente mediada.

Referências Bibliográficas

ABAR, S. et al. Agent based modelling and simulation tools: A review of the state-of-art software. *Computer Science Review*, Elsevier, v. 24, p. 13–33, 2017.

ALBERT, R.; BARABÁSI, A.-L. Statistical mechanics of complex networks. *Reviews of modern physics*, APS, v. 74, n. 1, p. 47, 2002.

ANDRADE, R. F. et al. Characterization of complex networks by higher order neighborhood properties. *The European Physical Journal B*, Springer, v. 61, n. 2, p. 247–256, 2008.

BENSCH, S.; HASSELQUIST, D. Evidence for active female choice in a polygynous warbler. *Animal Behaviour*, Elsevier, v. 44, p. 301–311, 1992.

BLONDEL, V. D. et al. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, IOP Publishing, v. 2008, n. 10, p. P10008, 2008.

BOCCARA, N. *Modeling complex systems*. [S.l.]: Springer Science & Business Media, 2010.

BONABEAU, E. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 99, n. suppl 3, p. 7280–7287, 2002.

BROOKS, R. A. Intelligence without representation. *Artificial intelligence*, Elsevier, v. 47, n. 1-3, p. 139–159, 1991.

CARON-LORMIER, G. et al. Asynchronous and synchronous updating in individual-based models. *Ecological Modelling*, Elsevier, v. 212, n. 3-4, p. 522–527, 2008.

CATCHPOLE, C.; SLATER, P. *Bird song: Biological themes and variations*. [S.l.]: Cambridge University Press, 2008.

CHEBIB, J.; MARRIOTT, C. Modeling the evolution of gene-culture divergence. In: MIT PRESS. *Proceedings of the Artificial Life Conference 2016 13*. [S.l.], 2016. p. 500–507.

CLAUSET, A.; NEWMAN, M. E.; MOORE, C. Finding community structure in very large networks. *Physical review E*, APS, v. 70, n. 6, p. 066111, 2004.

DARWIN, C. On the origin of species by means of natural selection. *Murray, London*, 1859.

DODD, D. M. Reproductive isolation as a consequence of adaptive divergence in drosophila pseudoobscura. *Evolution*, Wiley Online Library, v. 43, n. 6, p. 1308–1311, 1989.

ERIKSSON, D.; WALLIN, L. Male bird song attracts females - a field experiment. *Behavioral Ecology and Sociobiology*, Springer, v. 19, n. 4, p. 297–299, 1986.

ESTRADA, E.; KNIGHT, P. A. *A first course in network theory*. [S.l.]: Oxford University Press, USA, 2015.

FISHER, R. A. *The genetical theory of natural selection: a complete variorum edition*. [S.l.]: Oxford University Press, 1999.

FRANKLIN, S.; GRAESSER, A. Is it an agent, or just a program?: A taxonomy for autonomous agents. In: SPRINGER. *International Workshop on Agent Theories, Architectures, and Languages*. [S.l.], 1996. p. 21–35.

FREEMAN, B. G.; MONTGOMERY, G. A.; SCHLUTER, D. Evolution and plasticity: Divergence of song discrimination is faster in birds with innate song than in song learners in neotropical passerine birds. *Evolution*, Wiley Online Library, v. 71, n. 9, p. 2230–2242, 2017.

GRIMM, V. et al. A standard protocol for describing individual-based and agent-based models. *Ecological modelling*, Elsevier, v. 198, n. 1-2, p. 115–126, 2006.

GRIMM, V. et al. Pattern-oriented modeling of agent-based complex systems: lessons from ecology. *science*, American Association for the Advancement of Science, v. 310, n. 5750, p. 987–991, 2005.

GUDWIN, R. R. *IA009 - Introdução a teoria de Agentes*. 2010. Disponível em: <http://www.dca.fee.unicamp.br/~gudwin/courses/IA009/>. Acesso em: 01/05/2019.

HARRISON, R. G. *Diverse origins of biodiversity*. [S.l.]: Nature Publishing Group, 2001.

JANSSEN, M. A. *Introduction to Agent-Based Modeling*. [S.l.: s.n.], 2012.

KROODSMA, D. E. Songs of the alder flycatcher (empidonax alnorum) and willow flycatcher (empidonax traillii) are innate. *The Auk*, JSTOR, p. 13–24, 1984.

LACHLAN, R.; SERVEDIO, M. Song learning accelerates allopatric speciation. *Evolution*, Wiley Online Library, v. 58, n. 9, p. 2049–2063, 2004.

LYNCH, A. The population memetics of birdsong. *Ecology and evolution of acoustic communication in birds*, p. 181–197, 1996.

MACAL, C.; NORTH, M. Introductory tutorial: Agent-based modeling and simulation. In: IEEE. *Proceedings of the Winter Simulation Conference 2014*. [S.l.], 2014. p. 6–20.

MAYR, E. et al. Animal species and evolution. Belknap Press of Harvard University Press, 1963.

MEYER, D.; EL-HANI, C. N. *Evolução: o sentido da biologia*. [S.l.]: Unesp, 2005.

NELSON, D. A. A preference for own-subspecies' song guides vocal learning in a song bird. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 97, n. 24, p. 13348–13353, 2000.

NEWMAN, M. E. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 103, n. 23, p. 8577–8582, 2006.

NEWMAN, M. E.; GIRVAN, M. Finding and evaluating community structure in networks. *Physical review E*, APS, v. 69, n. 2, p. 026113, 2004.

NIXON, K. C.; WHEELER, Q. D. An amplification of the phylogenetic species concept. *Cladistics*, Wiley Online Library, v. 6, n. 3, p. 211–223, 1990.

NOTTEBOHM, F. The "critical period"for song learning. *Ibis*, Wiley Online Library, v. 111, n. 3, p. 386–387, 1969.

PATERSON, H. E.; MCEVEY, S. F. *Evolution and the recognition concept of species: collected writings*. [S.l.]: Johns Hopkins University Press Baltimore, 1993.

PYTHON SOFTWARE FOUNDATION. *Python 3.7.2*. 2018. Disponível em: <https://www.python.org/>. Acesso em: 08/07/2019.

QT COMPANY LTD. *Qt Creator 4.8.1, Based on Qt 5.12.0 (MSVC 2015, 32 bit)*. 2019. Disponível em: <https://www.qt.io/>. Acesso em: 08/07/2019.

RADESÄTER, T. et al. Song rate and pair formation in the willow warbler, phylloscopus trochilus. *Animal Behaviour*, Elsevier, v. 35, n. 6, p. 1645–1651, 1987.

RIDLEY, M. Evolução. 3ª edição. *Porto Alegre: Artmed*, 2006.

SALINAS, S. R. *Introdução a física estatística*. [S.l.]: Edusp, 2013.

SLATER, P. J. B. The study of communication. *Animal behaviour*, Blackwell Scientific Oxford, v. 2, p. 9–42, 1983.

THE IGRAPH CORE TEAM. *python-igraph 0.7.0*. 2014. Disponível em: <https://igraph.org/python/>. Acesso em: 08/07/2019.

THORPE, W. H. The learning of song patterns by birds, with especial reference to the song of the chaffinch fringilla coelebs. *Ibis*, Wiley Online Library, v. 100, n. 4, p. 535–570, 1958.

TURCHIN, P. Does population ecology have general laws? *Oikos*, Wiley Online Library, v. 94, n. 1, p. 17–26, 2001.

VERZIJDEN, M. N. et al. The impact of learning on sexual selection and speciation. *Trends in ecology & evolution*, Elsevier, v. 27, n. 9, p. 511–519, 2012.

WASSERMAN, F.; CIGLIANO, J. Song output and stimulation of the female in white-throated sparrows. *Behavioral Ecology and Sociobiology*, Springer, v. 29, n. 1, p. 55–59, 1991.

WHALING, C. S. et al. Acoustic and neural bases for innate recognition of song. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 94, n. 23, p. 12694–12698, 1997.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: Theory and practice. *The knowledge engineering review*, Cambridge University Press, v. 10, n. 2, p. 115–152, 1995.

ZAHAVI, A. Mate selection - a selection for a handicap. *Journal of theoretical Biology*, Elsevier, v. 53, n. 1, p. 205–214, 1975.

Apêndice A

Protocolo ODD: Conceitos de Design e Detalhes

A.1 Conceitos de Design

- **Emergência:** Esperamos encontrar duas propriedades emergentes: i) a formação de dialetos no canto dos Passeriformes, através do método de Louvain de detecção de comunidades (CLAUSET; NEWMAN; MOORE, 2004) em uma rede de acasalamentos possíveis por reconhecimento de canto. Checaríamos a intensidade desses dialetos através do calculo da modularidade da rede (NEWMAN, 2006); ii) a especiação, que mediríamos pela desconexão da rede de acasalamentos possíveis por reconhecimento de canto, seguindo o conceito de espécie por reconhecimento para acasalamento de Paterson e McEvey (1993).
- **Aprendizado:** Os filhotes machos de Oscines modificam o seu valor de canto através de aprendizado socialmente mediado desde que nascem até atingir o *Tempo de Cristalização* do canto.
- **Aptidão:** Não há nenhum calculo explícito de aptidão nesse modelo, porém ela está representada implicitamente. Os agentes machos que possuam um canto reconhecido por um maior número de fêmeas tem maiores chances de encontrar uma parceira e reproduzir, e da mesma forma uma fêmea que consiga identificar um maior número de machos como pertencente a sua espécie também tem maiores chances de encontrar um parceiro e reproduzir.
- **Percepção:** Todos os agentes podem escutar o *Valor de Canto* dos Passeriformes machos adultos que estiverem dentro do seu *Raio de Audição*.
- **Interação:** Os agentes sem parceiros podem formar casais para reproduzir no mês de acasalamento. Os agentes são monogâmicos. Os casais permanecem juntos o ano todo e se reproduzem nos meses de acasalamento, gerando um número aleatório de filhotes. Os filhotes permanecem parados juntos aos seus pais até que atinjam um *Tempo de Ninho*. Os agentes apresentam competição interespecífica representada por uma *Probabilidade de Morte* multiplicada pelo número de agentes dentro do *Raio de Competição*.
- **Estocasticidade:** Presente na inicialização do modelo nas seguintes variáveis internas dos agentes: *Posição* (X, Y), *Alelos de Canto*, *Valor de Canto* (sempre dentro do *Intervalo de*

Cantos Possíveis e Reconhecidos) e *Idade*. Durante a simulação, presente na escolha de parceiro das fêmeas (entre os machos reconhecidos), no movimento browniano correlacionado de passo variável, no número de pássaros nascidos de cada casal no mês de acasalamento, na *Probabilidade de Mutação* e no sentido da mutação (se leva ao aumento ou diminuição do valor de canto), na ordem na qual os agentes agem, no *Valor de Canto* inicial dos machos recém nacidos e na escolha dos *Alelos de Canto* transmitidos pela herança genética.

- **Observação:** Em cada iteração são armazenados o número de Passeriformes, a variabilidade na distribuição espacial dos agentes, calculada através da equação 4.1, e uma lista contendo as seguintes informações de cada agente: *ID*, *Sexo*, *Posição* (valores normalizados de X e Y), *Inicio do Intervalo de Cantos Possíveis e Reconhecidos*, *Valor de Canto*, *Alelos de Canto* e *Idade*.

Parâmetro	Símbolo	
Aprendizado do Canto	A	
Tempo de Cristalização do Canto	T_c	(em meses)
Tempo de Ninho	T_n	(em meses)
Idade Máxima	I_m	(em anos)
Máximo de Filhotes por Ninhada	F	
Tamanho do Intervalo de Possíveis e Reconhecidos	I_c	
Passo dos Agentes	P	
Ângulo de Movimento	θ	(em graus)
Proporção do Passo de Migração	P_f	
Ângulo de Movimento de Migração	θ_f	(em graus)
Probabilidade de Acasalamento	P_a	
Probabilidade de Mutação	M_p	
Tamanho da Mutação	M_t	
Raio de Audição	R_a	
Raio de Competição	R_c	
Probabilidade de Morte	P_m	
Quantidade Inicial de Passeríformes	N_i	

Tab. A.1: Lista dos parâmetros de entrada do modelo.

A.2 Inicialização

O ambiente é criado, e preenchido com agentes machos e fêmeas, em igual quantidade, dispostos de maneira aleatória no espaço seguindo uma distribuição uniforme. Cada agente é iniciado com valores para suas variáveis internas que dependem de algumas das informações de entrada, que descreveremos com os símbolos indicados pela tabela A.1. Todas as distribuições usadas na inicialização são uniformes e geradas pela mesma semente.

Cada agente, macho ou fêmea, é inicializado com uma *Idade* aleatória do intervalo de inteiros $[1; (Im - 1)]$. As coordenadas X e Y são sorteadas aleatoriamente, do intervalo dos racionais $[0; 1[\subset R$, o que significa dizer o tamanho do mundo é normalizado no modelo de mundo do agente. Os ângulos iniciais são sorteados do intervalo dos racionais $[0; 360[\subset R$, em graus. Cada um dos dois *Alelos de Canto* do agente é sorteado do intervalo dos racionais $[2.5 - Mt/2; 2.5 + Mt/2] \subset R$, que define uma distribuição uniforme e estreita centrada no valor arbitrário 2.5 (este valor é arbitrário e não interfere na simulação). É feita a média entre os *Alelos de Canto* para determinar o *Inicio do Intervalo de Cantos Possíveis e Reconhecidos*, e um *Valor de Canto* é sorteado aleatoriamente dentro desse intervalo (porém o *Valor de Canto* só é utilizado em agentes machos).

A.3 Entrada

As variáveis fornecidas como entrada da simulação estão representadas na tabela A.1.

A.4 Submodelos

- **Crescer** (*mês regular*)**:** três variáveis são importantes para esse submodelo: a variável booleana *Estágio da Vida*, que especifica a etapa da vida do agente (caso $true$ o agente é um adulto, caso $false$ o agente é um filhote); uma variável inteira chamada *Contador de Meses*, que nasce zerada e é incrementada em 1 a cada iteração de mês; e a variável *Tempo de Cristalização*, que armazena quantos meses o agente permanece como filhote. Em cada a iteração referente a um mês o agente compara *Contador de Meses* com o *Tempo de Cristalização*, e caso *Contador de Meses* $>=$ *Tempo de Cristalização*, então o agente atribui *Estágio da Vida* $= true$, se tornando adulto;

- **Aprender** (*mês regular*)**:** o aprendizado socialmente mediado é implementado alterando o *Valor de Canto* dos filhotes machos para uma média entre o *Valor de Canto* atual e a moda do *Valores de Canto* dos machos adultos da vizinhança:

$$Novo\,Valor\,de\,Canto = \frac{Valor\,de\,Canto\,Atual + Moda\,da\,Vizinhana}{2} \tag{A.1}$$

  Como os cantos assumem valores contínuos a moda é feita através da discretização do *Intervalo de Cantos Possíveis e Reconhecidos*. O intervalo é dividido em dez seguimentos de igual tamanho e aquele que contém o maior número de *Valores de Canto* dentro do *Raio de Audição* é selecionado, havendo um sorteio em caso de empate. A moda é dada então pela média dos *Valores de Canto* que se encontram dentro do seguimento selecionado;

- **Andar** (*mês regular*)**:** os agentes podem se mover ou ficar parados. Quando há movimento ele é um movimento browniano correlacionado com passo de tamanho variável. Os tamanhos do passos são calculados através da multiplicação de um *Passo Base* com um fator que é o módulo do valor sorteado por uma função que segue uma distribuição normal, a função *normal_distribution* da biblioteca *random* do C++, que segue a seguinte função densidade de probabilidade:

$$p\left(x|\mu,\sigma\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\left(x-\mu\right)^2}{2\sigma^2}\right) \tag{A.2}$$

No C++, a função *normal_distribution* necessita também de um gerador de números aleatórios criado pelo construtor *default_random_engine*, para o qual estabelecemos uma semente. Para que a esta seja aleatória para cada agente, usamos números aleatórios gerados pela função *rand*.

A decisão de se mover ou ficar parado depende do *Estágio de Vida* do agente (filhote ou adulto). Vamos descrever cada caso:

- **Filhote:** o tipo de movimento desempenhado pelos filhotes depende do *tempo de ninho*. Quando *contador de meses* $<=$ *tempo de ninho* o filhote permanece parado e quando *contador de meses* $>$ *tempo de ninho* o agente se move;
- **Adultos:** quando possuem parceiro, os adultos permanecem parados simulando um ninho até que um dos membros do casal morra. Quando não possuem parceiro os agentes se movem;

O tamanho do *Passo Base* e do ângulo utilizado depende da idade do agente:

- $Idade > 0$**:** O *Passo Base* é dado pelo parâmetro de entrada *Passo dos Agentes* e o ângulo é dado pelo parâmetro *Ângulo de Movimento*;
- $Idade = 0$**:** O *Passo Base* é o parâmetro *Passo dos Agentes* multiplicado pelo parâmetro *Proporção do Passo de Migração*, que é sempre maior que 1, ambos fornecidas como entrada. Além disso o ângulo utilizado é o *Ângulo de Movimento de Migração*, que é diferente do normal e escolhido para maximizar a difusão os agentes no espaço. Essas mudanças visam simular o efeito de migração no movimento de agentes jovens;

- **Morrer** (*mês de acasalamento*)**:** esse método ocorre em duas etapas, a de mortes por idade e a de mortes por competição, que acontecem nessa ordem. As etapas estão descritas abaixo:
  - **Mortes por Idade:** todos os agentes para os quais *Idade* $>=$ *Idade Maxima* são mortos, sendo *Idade Maxima* um parâmetro fornecido como entrada;
  - **Mortes por Competição:** analisamos um a um, em uma ordem aleatória, se cada agente morre ou permanece vivo. Em cada um deles é calculada uma probabilidade individual de morte pela multiplicação do parâmetro de entrada *Probabilidade de Morte* com o número de agentes vivos dentro do seu *Raio de Competição*. Em seguida é sorteado um valor do intervalo $[0;1] \subset R$, caso o valor sorteado seja menor que a probabilidade individual de morte o agente morre, e caso seja maior o agente vive;

- **Escolher parceiro** (*mês de acasalamento*)**:** executamos uma a uma, em ordem aleatória, a análise das fêmeas aos machos em seu *Raio de Audição*. Para cada uma delas, os machos vizinhos, que são apenas aqueles cujo *Valor do Canto* está dentro do *Intervalo de Cantos Possíveis e Reconhecidos* da fêmea, são dispostos em uma lista aleatória. Para cada macho da lista a fêmea sorteia um valor do intervalo $[0;1] \subset R$, caso esse valor seja menor que a *Probabilidade de*

Acasalamento a fêmea toma o macho como seu parceiro. Quando um macho da lista é selecionado, os demais que ainda não tiveram uma chance são descartados. Caso nenhum macho seja tomado como parceiro a fêmea permanece sem parceiros nesse mês de acasalamento;

- **Reproduzir** (*mês de acasalamento*)**:** cada casal de agentes tem um numero de filhotes aleatório sorteado com uma distribuição de probabilidade homogênea do intervalo $[0; F[\subset N$, onde F é o *Máximo de Filhotes por Ninhada*, conforme a tabela A.1. Cada um desses filhotes pode ser macho ou fêmea, com igual probabilidade. Os valores das coordenadas X e Y de nascimento dos filhotes são as mesmas da mãe. Os valores de ambos os *Alelos de Canto* são escolhidos aleatoriamente, sendo um deles do pai e o outro da mãe, de maneira a obedecer as proporções mendelianas. Todas as demais variáveis são inicializadas como as dos agentes do inicio da simulação, conforme a seção A.2 que trata da inicialização;

Apêndice B

Código

B.1 main.cpp

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#include "ambiente.h" // para criar o agente e o ambiente
#include "agente.h"

using namespace std;

int main(int argc, char *argv[])
{
//#### inicializando os parametros ####

// parametros ligados a execucao da simulacao

unsigned int ladoGrade = 175;

unsigned int ladoVariacao = 7; // ladoVariacao eh divisor de ladoGrade

unsigned int numeroDeGeracoes = 20000;

unsigned int tempoDeEquilibrio = 16000;

// parametros do modelo

bool oscines = true;
```

```
31
32 int subgenero = atoi(argv[1]);
33 if (subgenero == 0)
34 {
35 oscines = false;
36 }
37
38 int semente = atoi(argv[2]);
39
40 // demais parametros
41
42 unsigned int capacidadeDeSuporte = atoi(argv[3]);
43
44 unsigned int tempoDeCristalizacao = atoi(argv[4]);
45
46 unsigned int tempoDeNinho = atoi(argv[5]);
47
48 unsigned int idadeMaxima = atoi(argv[6]);
49
50 unsigned int maximoFilhos = atoi(argv[7]);
51
52 double tamanhoRangeCanto = atof(argv[8]);
53
54 double passoDosAgentes = atof(argv[9]);
55
56 double propMigracao = atof(argv[10]);
57
58 double probabilidadeDeAcasalamento = atof(argv[11]);
59
60 double probabilidadeDeMutacao = atof(argv[12]);
61
62 double passoDeMutacao = atof(argv[13]);
63
64 double audicao = atof(argv[14]);
65
66 double distanciaCompeticao = atof(argv[15]);
67
68 double probabilidadeDeMorte = atof(argv[16]);
69
70 srand(semente);  // aplicando a semente
71
72 // criando pasta
73
74 char argUm[10];
75 strcpy(argUm,argv[1]);
76 char argDois[10];
77 strcpy(argDois,argv[2]);
```

```
78  char argTres[10];
79  strcpy(argTres,argv[3]);
80
81  char diretorio[] = "C:\\resultados\\A=";
82
83  strcat(diretorio,argUm);
84  strcat(diretorio,"_Se=");
85  strcat(diretorio,argDois);
86
87  mkdir(diretorio);
88
89  //#### ARQUIVO DE PARAMETROS ####
90
91  ofstream parametros;
92  parametros.open("C:\\resultados\\A="+argv[1]+"_Se="+argv[2]+"\\
       parametros_A="+argv[1]+"_.txt");
93
94  parametros << subgenero << " " << numeroDeGeracoes << " " <<
       tempoDeEquilibrio << endl << " " << endl << endl;
95
96  parametros << "Simulacao do Modelo de especiacao de ";
97
98  // escrevemos oscine ou suboscines no cabecalho
99  if (oscines==true) {parametros << "Oscines";}
100 else {parametros << "Suboscines";}
101
102 parametros << endl << endl;
103
104 parametros << "semente=" << semente << endl;
105 parametros << "ladoGrade=" << ladoGrade << endl;
106 parametros << "numeroDeGeracoes=" << numeroDeGeracoes << endl;
107 parametros << "tempoDeEquilibrio=" << tempoDeEquilibrio << endl;
108
109 parametros << endl;
110
111 parametros << "Lista de Parametros Utilizados:" << endl << endl;
112
113 parametros << "capacidadeDeSuporte=" << capacidadeDeSuporte << endl;
114 parametros << "tempoDeCristalizacao=" << tempoDeCristalizacao << endl;
115 parametros << "tempoDeNinho=" << tempoDeNinho << endl;
116 parametros << "idadeMaxima=" << idadeMaxima << endl;
117 parametros << "maximoFilhos=" << maximoFilhos << endl;
118 parametros << "tamanhoDoRangeDeCanto=" << tamanhoRangeCanto << endl;
119 parametros << "passoDosAgentes=" << passoDosAgentes << endl;
120 parametros << "propMigracao=" << propMigracao << endl;
121 parametros << "probabilidadeDeAcasalamento=" <<
       probabilidadeDeAcasalamento << endl;
```

```
122 parametros << "probabilidadeDeMutacao=" << probabilidadeDeMutacao <<endl;
123 parametros << "passoDeMutacao=" << passoDeMutacao << endl;
124 parametros << "audicao=" << audicao << endl;
125 parametros << "distanciaCompeticao=" << distanciaCompeticao << endl;
126 parametros << "probabilidadeDeMorte=" << probabilidadeDeMorte << endl;
127
128 //#### ARQUIVOS DE RESULTADOS ####
129
130 ofstream numeroDeIndividuos;
131 numeroDeIndividuos.open("C:\\resultados\\A="+argv[1]+"_Se="+argv[2]+"\\
       numeroDeIndividuos_A="+argv[1]+"_.txt");
132
133 ofstream variacaoEspacial;
134 variacaoEspacial.open("C:\\resultados\\A="+argv[1]+"_Se="+argv[2]+"\\
       variacaoEspacial_A="+argv[1]+"_.txt");
135
136 //#### CONSTRUINDO O AMBIENTE ####
137
138 ambiente caatinga(oscines, ladoGrade, capacidadeDeSuporte,
       tempoDeCristalizacao, tempoDeNinho, idadeMaxima, maximoFilhos,
       tamanhoRangeCanto, passoDosAgentes, probabilidadeDeAcasalamento,
       probabilidadeDeMutacao, passoDeMutacao, propMigracao, audicao,
       distanciaCompeticao, probabilidadeDeMorte);
139
140 // armazenando o numero de passaros inicial
141 numeroDeIndividuos << caatinga.getNumeroPassaros() << endl;
142
143 // armazenando a variacao espacial inicial
144 variacaoEspacial << caatinga.variacaoEspacial(ladoVariacao) << endl;
145
146 //#### VARIAVEIS PARA CONTROLE DE POPULAÇÃO E TEMPO ####
147
148 bool zerou = false;
149 bool explodiu = false;
150
151 clock_t tempoInicial,tempoFinal;
152
153 tempoInicial=clock();  //iniciando o relogio
154
155 //#### COLOCANDO O SISTEMA EM EQUILIBRIO ####
156
157 for (unsigned int i=0; i<tempoDeEquilibrio; i++)
158 {
159 caatinga.rodaGeracaoPatches();  // faz uma iteracao de uma geracao
160
161 cout << i << ": " << caatinga.getNumeroPassaros() << endl; // na tela
162
```

```
163 // armazenando alguns dados
164 numeroDeIndividuos << caatinga.getNumeroPassaros() << endl;
165 variacaoEspacial << caatinga.variacaoEspacial(ladoVariacao) << endl;
166
167 // controle de populacao
168
169 if (caatinga.getNumeroPassaros() == 0)
170 {
171 zerou = true;
172 break;
173 }
174
175 if (caatinga.getNumeroPassaros() == 30000)
176 {
177 explodiu = true;
178 break;
179 }
180 }
181
182 tempoFinal=clock();  // tempo para atingir o equilibrio
183
184 // no arquivo de parametros
185 parametros << endl << "tempo até atingir o equilibrio: ";
186 parametros << (double)(tempoFinal-tempoInicial)/CLOCKS_PER_SEC;
187 parametros << endl << endl;
188
189 // na tela
190 cout << endl << "Equilibrio atingido em ";
191 cout << (double)(tempoFinal-tempoInicial)/CLOCKS_PER_SEC;
192 cout << " segundos!" << endl << endl;
193
194 //#### SIMULACAO ####
195
196 cout << "Começando a simulação: " << endl << endl;
197
198 for (unsigned int i=0; i<numeroDeGeracoes; i++)
199 {
200 caatinga.rodaGeracaoPatches();  // faz uma iteracao de uma geracao
201
202 cout << i << ": " << caatinga.getNumeroPassaros() << endl;  // na tela
203
204 // armazenando alguns dados
205 numeroDeIndividuos << caatinga.getNumeroPassaros() << endl;
206 variacaoEspacial << caatinga.variacaoEspacial(ladoVariacao) << endl;
207
208 // armazenando variaveis internas dos agentes
209
```

```
210 ofstream tabelaDePassaros;
211 tabelaDePassaros.open("C:\\resultados\\A="+argv[1]+"_Se="+argv[2]+"\\
       tabela_"+to_string(i)+"_A="+argv[1]+"_.txt");
212
213 for (int j = 0; j < caatinga.getNumeroPassaros(); j++)
214 {
215 agente *ponteiro = caatinga.getPonteiroAgente(j);
216 int sexo = 0;
217 if (ponteiro->getEhMacho() == true)
218 {sexo = 1;}
219
220 tabelaDePassaros << ponteiro->getId() << " ";
221 tabelaDePassaros << sexo << " ";
222 tabelaDePassaros << ponteiro->getLocal().getX() << " ";
223 tabelaDePassaros << ponteiro->getLocal().getY() << " ";
224 tabelaDePassaros << ponteiro->getInicioRangeCanto() << " ";
225 tabelaDePassaros << ponteiro->getValorCanto() << " ";
226 tabelaDePassaros << ponteiro->getGeneCantoI() << " ";
227 tabelaDePassaros << ponteiro->getGeneCantoII() << " ";
228 tabelaDePassaros << ponteiro->getIdade() << " ";
229 tabelaDePassaros << endl;
230 }
231
232 tabelaDePassaros.close();
233
234 // controle de populacao
235
236 if (caatinga.getNumeroPassaros() == 0)
237 {
238 zerou = true;
239 break;
240 }
241
242 if (caatinga.getNumeroPassaros() == 30000)
243 {
244 explodiu = true;
245 break;
246 }
247 }
248
249 tempoFinal=clock();  // tempo de simulacao
250
251 // na tela
252 cout << endl << "tempo de simulacao: ";
253 cout << (double)(tempoFinal-tempoInicial)/CLOCKS_PER_SEC;
254 cout << endl << endl;
255
```

```
256 // no arquivo de parametros
257 parametros << endl << "tempo de simulacao: ";
258 parametros << (double)(tempoFinal-tempoInicial)/CLOCKS_PER_SEC;
259 parametros << endl << endl;
260
261 if (zerou == true)
262 {parametros << "Zerou" << endl;
263
264 if (explodiu == true)
265 {parametros << "Explodiu" << endl;}
266
267 parametros.close();
268 numeroDeIndividuos.close();
269 variacaoEspacial.close();
270 }
```

B.2 agente.h

```
1  #ifndef AGENTE_H
2  #define AGENTE_H
3  #include <vector>           // para criar vetor de vizinhanca
4  #include <stdlib.h>         // para gerar numeros aleatoriamente
5  #include "posicao.h"        // objetos com as posicoes dos agentes
6  #include "movimentacao.h"   // objetos com as regras de movimentacao
7
8  using namespace std;
9
10 class agente
11 {
12 public:
13 // #### construtor para os agentes no inicio da simulacao
14
15 agente(int id,
16 bool macho,
17 bool oscine,
18 double tamRange,
19 double passo,
20 double propMigracao,
21 double probAcasalamento,
22 double probMutacao,
23 double passoMutacao,
24 unsigned int tempoDeCristalizacao,
25 unsigned int tempoDeNinho,
26 unsigned int idadeMaxima,
27 unsigned int indiceMaximo,
28 double audicao);
```

```
29
30 // #### construtor para agentes nascidos pela reprodução sexuada
31
32 agente(int id,
33 bool macho,
34 bool oscine,
35 double tamRange,
36 double passo,
37 double propMigracao,
38 double probAcasalamento,
39 double probMutacao,
40 double passoMutacao,
41 unsigned int tempoDeCristalizacao,
42 unsigned int tempoDeNinho,
43 unsigned int indiceMaximo,
44 double audicao,
45 double geneMaeI,
46 double geneMaeII,
47 double genePaiI,
48 double genePaiII,
49 posicao localMae);
50
51 /*#### metodos chamados pelo ambiente ####*/
52
53 void rodaModelo();                    // roda o modelo de mundo do agente
54 void atuacao();                       // responsavel pela atuacao do agente
55 void fimDeAcasalamento();             // finaliza o relacionamento
56 void fimDeAno();                      // modificacoes de fim do ano
57
58 //#### metodos que lidam com a vizinhança de um agente (no ambiente)
59
60 void limparVizinhanca();
61 void adicionarVizinhanca(agente *vizinho);
62
63 /*#### metodos get end set ####*/
64
65 posicao getLocal() const;
66 double getRaioVizinhanca() const;
67 bool getEhMacho() const;
68 double getGeneCantoI() const;
69 double getGeneCantoII() const;
70 double getValorCanto() const;
71 vector <agente *> getVizinhanca() const;
72 bool getAdulto() const;
73 bool getFlertando() const;
74 void setFlertando(bool value);
75 int getIdade() const;
```

```
76  bool getEmRelacionamento() const;
77  movimentacao getMovimento() const;
78  double getInicioRangeCanto() const;
79  double getRangeCanto() const;
80  bool getMorto() const;
81  void setMorto(bool value);
82  int getId() const;
83  int getIndiceX() const;
84  int getIndiceY() const;
85  void setVizinhanca(const vector<agente *> &value);
86  bool getMudaPatch() const;
87  int getVizinhancaSize() const;
88  int getIndiceXVelho() const;
89  int getIndiceYVelho() const;
90  int getParceiro() const;
91
92  private:
93  int id;                         // identificador do agente
94
95  /*#### variaveis de estado ####*/
96
97  bool oscine;          // determina se temos um oscine ou um suboscine
98  bool ehMacho;         // determina se temos um agente macho ou femea
99  bool adulto;                    // para diferenciar adultos e filhotes
100 bool flertando;                 // passaro em epoca de acasalamento
101 bool emRelacionamento;          // passaro encontrou um parceiro
102 bool morto;                     // determina se esta morto
103 int contadorDeMeses;           // conta os meses do ano
104 int tempoDeCristalizacao;      // meses para que um filhote macho cresca
105 int tempoDeNinho;              // meses que um agente fica no ninho
106 int idade;                     // tempo de vida em anos
107
108 /*#### variaveis espaciais e de interacao ####*/
109
110 posicao local;                 // objeto com as posicoes do agente
111 movimentacao movimento;        // objeto com as regras de movimentacao
112 double propMigracao;           // proporcao dos passos de um filhote
113 vector <agente *> vizinhanca;  // vetor de ponteiros com vizinhos
114 int parceiro;                  // id do parceiro
115 double raioVizinhanca;         // tamanho da vizinhanca de um agente
116 double audicao;                // para determinar o raioVizinhanca
117
118 /*#### variaveis relacionadas aos cantos ####*/
119
120 double geneCantoI;           // alelo de canto numero 1
121 double geneCantoII;          // alelo de canto numero 2
122 double inicioRangeCanto;     // media aritmetica entre os genes 1 e 2
```

```
123 double rangeCanto;           // tamanho da faixa de canto reconhecível
124 double valorCanto;           // valor de canto do macho
125 double passoMutacao;             // tamanho do passo de mutação dado
126 double probabilidadeMutacao;     // probabilidade de ocorrer mutação
127 double probabilidadeAcasalamento;// probabilidade de acasalamento
128
129 // #### variaveis relacionadas aos patches
130 // cada agente possui armazenadas seus indices atualizados
131
132 int indiceXVelho;      // indices dos patches no inicio da atualizacao
133 int indiceYVelho;
134
135 int indiceX;           // novos indices dos patches do agente
136 int indiceY;
137
138 int indiceMaximo;      // maior valor que um indice pode assumir
139
140 void atualizaIndice(); // metodo que atualiza os valores dos indicies
141 bool mudaPatch;        // true caso haja mudanca de indice
142
143 /*#### alguns métodos auxiliares privados ####*/
144
145 void aprenderCanto();       // agente aprende o canto
146 void condicoesDeContorno(); // aplica as condicoes de contorno ao agente
147 void namoraComigo(int id);  // torna um passaro macho 'parceiro'
148 void selecionarParceiro();  // identifica o parceiro
149
150 //#### diferentes modelos de mundo para agentes diferentes
151
152 void modeloOscineMacho();
153 void modeloOscineFemea();
154 void modeloSuboscineMacho();
155 void modeloSuboscineFemea();
156 };
157
158 #endif // AGENTE_H
```

B.3 agente.cpp

```
1 #include "agente.h"
2 #include <vector>
3 #include <algorithm>
4
5 #define PI 3.1416
6 #define angulo 140
7 #define anguloMigracao 80
```

```
8
9  agente::agente(int id,
10 bool macho,
11 bool oscine,
12 double tamRange,
13 double passo,
14 double propMigracao,
15 double probAcasalamento,
16 double probMutacao,
17 double passoMutacao,
18 unsigned int tempoDeCristalizacao,
19 unsigned int tempoDeNinho,
20 unsigned int idadeMaxima,
21 unsigned int indiceMaximo,
22 double audicao)
23 {
24 // cria agentes dispostos aleatoriamente no inicio da simulacao
25
26 this->id = id;
27
28 this->oscine = oscine;
29 this->morto = false;
30 this->idade = rand()%(idadeMaxima-1) +1;  // a idade inicial nunca eh 0
31 this->contadorDeMeses = 0;
32 this->tempoDeCristalizacao = tempoDeCristalizacao;
33 this->tempoDeNinho = tempoDeNinho;
34
35 this->adulto = true;
36 this->ehMacho = macho;          // assinalando o sexo do agente
37 this->flertando = false;        // definindo os passaros como solteiros
38 this->emRelacionamento = false; // não esta em relacionamento
39 this->probabilidadeAcasalamento = probAcasalamento;
40
41 this->local.setX((double)rand()/RAND_MAX);  // variaveis de "local"
42 this->local.setY((double)rand()/RAND_MAX);  // da classe "posicao"
43
44 this->condicoesDeContorno();  // para evitar X e Y igual a 1
45
46 this->movimento.setPasso(passo); // variaveis da classe "movimentacao"
47 this->movimento.setPassoMigracao(passo * propMigracao);
48 this->movimento.setDirecao((double)rand()/RAND_MAX*360);
49 this->movimento.setAnguloDeVisao(angulo);
50 this->movimento.setAnguloDeMigracao(anguloMigracao);
51
52 this->rangeCanto = tamRange;
53
54 this->probabilidadeMutacao = probMutacao;
```

```
55 this->passoMutacao = passoMutacao;
56
57 this->geneCantoI = (2.5 - this->passoMutacao/2) + this->passoMutacao * ((
      double)rand()/RAND_MAX);
58 this->geneCantoII = (2.5 - this->passoMutacao/2) + this->passoMutacao *
      ((double)rand()/RAND_MAX);
59 this->inicioRangeCanto = (geneCantoI + geneCantoII)/2;
60 this->valorCanto = (this->inicioRangeCanto) + (this->rangeCanto) * ((
      double)rand()/RAND_MAX);
61
62 // inicializando as demais variaveis
63
64 this->audicao = audicao;
65 this->raioVizinhanca = this->audicao*this->movimento.getPasso();
66 this->limparVizinhanca();
67 this->parceiro = 0;
68
69 // inicializando variaveis referentes aos indices do agente
70
71 this->indiceMaximo = indiceMaximo;
72 this->indiceX=this->local.getX() * this->indiceMaximo;
73 this->indiceY=this->local.getY() * this->indiceMaximo;
74 }
75
76 agente::agente(int id,
77 bool macho,
78 bool oscine,
79 double tamRange,
80 double passo,
81 double propMigracao,
82 double probAcasalamento,
83 double probMutacao,
84 double passoMutacao,
85 unsigned int tempoDeCristalizacao,
86 unsigned int tempoDeNinho,
87 unsigned int indiceMaximo,
88 double audicao,
89 double geneMaeI,
90 double geneMaeII,
91 double genePaiI,
92 double genePaiII,
93 posicao localMae)
94 {
95 // cria agentes que surgirao por reproducao sexuada
96
97 this->id = id;
98
```

```
99  this->oscine = oscine;
100 this->morto = false;
101 this->idade = 0;
102 this->contadorDeMeses = 0;
103 this->tempoDeCristalizacao = tempoDeCristalizacao;
104 this->tempoDeNinho = tempoDeNinho;
105
106 this->ehMacho = macho;    // assinalando o sexo do agente
107
108 // apenas temos filhotes oscines machos
109 if (this->oscine == true)
110 {
111 if (this->ehMacho == true) {this->adulto = false;}
112 else {this->adulto = true;}
113 }
114 else {this->adulto = true;}
115
116 this->flertando = false;        // definindo os passaros como solteiros
117 this->emRelacionamento = false; // nao esta em relacionamento
118 this->probabilidadeAcasalamento = probAcasalamento;
119
120 this->local = localMae;  // "local" iniciada com o posicionamento da mae
121
122 this->condicoesDeContorno();
123
124 this->movimento.setPasso(passo);  // variaveis da classe "movimentacao"
125 this->movimento.setPassoMigracao(passo * propMigracao);
126 this->movimento.setDirecao((double)rand()/RAND_MAX*360);
127 this->movimento.setAnguloDeVisao(angulo);
128 this->movimento.setAnguloDeMigracao(anguloMigracao);
129
130 this->rangeCanto = tamRange;
131
132 this->passoMutacao = passoMutacao;
133 this->probabilidadeMutacao = probMutacao;
134
135 // vamos decidir os alelos I e II do filhote
136 // primeiro para o aleloI (gene do pai):
137
138 double dado = (double)rand()/RAND_MAX;
139
140 if (dado < 0.5){this->geneCantoI = genePaiI;}
141 else{this->geneCantoI = genePaiII;}
142
143 // chance de mutação em relacao ao gene escolhido
144
145 dado = (double)rand()/RAND_MAX;    // chance de mutação genetica
```

```
146
147 if (dado < this->probabilidadeMutacao){
148 //mutacao pode ser 'positiva' ou 'negativa'
149
150 dado = (double)rand()/RAND_MAX;
151
152 if (dado < 0.5){this->geneCantoI += this->passoMutacao;}// passo positivo
153 else{this->geneCantoI -= this->passoMutacao;}// passo negativo
154 }
155
156 // agora para o alelo II (gene da mae):
157
158 dado = (double)rand()/RAND_MAX;
159
160 if (dado < 0.5){this->geneCantoII = geneMaeI;}
161 else{this->geneCantoII = geneMaeII;}
162
163 // chance de mutacao em relacao ao gene escolhido
164
165 dado = (double)rand()/RAND_MAX;    // chance de mutação genetica
166
167 if (dado < this->probabilidadeMutacao)
168 {
169 //mutacao que pode ser 'positiva' ou 'negativa'
170
171 dado = (double)rand()/RAND_MAX;
172
173 if(dado < 0.5){this->geneCantoII += this->passoMutacao;}// passo positivo
174 else{this->geneCantoII -= this->passoMutacao;}// passo negativo
175 }
176
177 // usamos os genes para determinar o inicio do range
178
179 this->inicioRangeCanto = (geneCantoI + geneCantoII)/2;
180
181 // colocamos um valor aleatorio inicial no canto dos machos;
182
183 this->valorCanto = (this->inicioRangeCanto) + (this->rangeCanto) * ((
      double)rand()/RAND_MAX);
184
185 // incicializando as demais variaveis
186
187 this->audicao = audicao;
188 this->raioVizinhanca = this->audicao * this->movimento.getPasso();
189 this->limparVizinhanca();
190 this->parceiro = 0;
191
```

```
192 // inicializando variaveis referentes aos indices do agente
193
194 this->indiceMaximo = indiceMaximo;
195 this->indiceX = this->local.getX() * this->indiceMaximo;
196 this->indiceY = this->local.getY() * this->indiceMaximo;
197 }
198
199 void agente::rodaModelo()
200 {
201 this->contadorDeMeses++;
202
203 if (this->oscine == true){
204 if (this->ehMacho == true){this->modeloOscineMacho();}
205 else{this->modeloOscineFemea();}
206 }
207 else{
208 if (this->ehMacho == true){this->modeloSuboscineMacho();}
209 else{this->modeloSuboscineFemea();}
210 }
211 }
212
213 void agente::modeloOscineMacho()
214 {
215 // modelo de mundo dos oscines machos
216
217 // primeiramente determinamos o aprendizado
218
219 if (this->adulto == false)    // caso seja filhote
220 {
221 this->aprenderCanto();    // temos aprendizado do canto
222
223 if (this->contadorDeMeses >= this->tempoDeCristalizacao)
224 {this->adulto = true;}  // se virou adulto seta adulto como true
225 }
226
227 // determinamos o movimento do oscine
228
229 if (this->flertando == false) // mes regular
230 {
231 if (this->idade == 0)             // primeiro ano de vida
232 {
233 if (this->contadorDeMeses > this->tempoDeNinho)  // apos sair do ninho
234 {this->movimento.movimentoMigracao();}
235 else                                             // enquanto no ninho
236 {this->movimento.parado();}
237 }
238 else                              // demais anos
```

```
239 {
240 if (this->emRelacionamento == true)  // caso tenha parceiro
241 {this->movimento.parado();}
242 else                                 // caso nao tenha parceiro
243 {this->movimento.movimento();}
244 }
245 }
246 else                           // mes de acasalamento
247 {this->movimento.parado();}    // quem escolhe eh a femea
248 }
249
250 void agente::modeloOscineFemea()
251 {
252 // modelo de mundo dos oscines femeas
253
254 if (this->flertando == false)  // mes regular
255 {
256 if (this->idade == 0)    // primeiro ano de vida
257 {
258 if (this->contadorDeMeses > this->tempoDeNinho)   // apos sair do ninho
259 {this->movimento.movimentoMigracao();}
260 else                                               // enquanto no ninho
261 {this->movimento.parado();}
262 }
263 else                      // demais anos
264 {
265 if (this->emRelacionamento == true)     // caso tenha parceiro
266 {this->movimento.parado();}
267 else                                    // caso nao tenha parceiro
268 {this->movimento.movimento();}
269 }
270 }
271 else    // mes de acasalamento
272 {
273 if (this->emRelacionamento == false)  // caso nao tenha um parceiro
274 {
275 this->movimento.parado();             // parada
276 this->selecionarParceiro();           // a femea seleciona o parceiro
277 }
278 else                                  // caso tenha um parceiro
279 {this->movimento.parado();}           // apenas fica parada
280 }
281 }
282
283 void agente::modeloSuboscineMacho()
284 {
285 // modelo de mundo dos suboscines machos
```

```
286
287 if (this->flertando == false) // mes regular
288 {
289 if (this->idade == 0)    // primeiro ano de vida
290 {
291 if (this->contadorDeMeses > this->tempoDeNinho) // apos sair do ninho
292 {this->movimento.movimentoMigracao();}
293 else                                          // enquanto no ninho
294 {this->movimento.parado();}
295 }
296 else                    // demais anos
297 {
298 if (this->emRelacionamento == true)    // caso tenha parceiro
299 {this->movimento.parado();}
300 else                                   // caso nao tenha parceiro
301 {this->movimento.movimento();}
302 }
303 }
304 else                        // mes de acasalamento
305 {this->movimento.parado();   // quem escolhe eh a femea}
306 }
307
308 void agente::modeloSuboscineFemea()
309 {
310 // modelo de mundo dos suboscines femeas
311
312 if (this->flertando == false) // mes regular
313 {
314 if (this->idade == 0)    // primeiro ano de vida
315 {
316 if (this->contadorDeMeses > this->tempoDeNinho)  // apos sair do ninho
317 {this->movimento.movimentoMigracao();}
318 else                                           // enquanto no ninho
319 {this->movimento.parado();}
320 }
321 else                    // demais anos
322 {
323 if (this->emRelacionamento == true)    // caso tenha parceiro
324 {this->movimento.parado();}
325 else                                   // caso nao tenha parceiro
326 {this->movimento.movimento();}
327 }
328 }
329 else                        // mes de acasalamento
330 {
331 if (this->emRelacionamento == false)  // caso nao tenha um parceiro
332 {
```

```
333 this->movimento.parado();                // parada
334 this->selecionarParceiro();              // a femea seleciona o parceiro
335 }
336 else                                     // caso tenha um parceiro
337 {this->movimento.parado();}              // apenas fica parada
338 }
339 }
340
341 void agente::atuacao()
342 {
343 // primeiramente nos deslocamos a quantidade definida no 'rodaModelo'
344
345 this->local.acumulaX(this->movimento.getDx());
346 this->local.acumulaY(this->movimento.getDy());
347
348 // depois aplicamos as condições de contorno e atualizamos os indices
349
350 this->condicoesDeContorno();
351 this->atualizaIndice();
352 }
353
354 void agente::aprenderCanto()
355 {
356 // filhote oscine macho modifica seu canto atraves de aprendizado
357
358 // primeiros checamos se algum vizinho eh reconhecivel
359
360 if (this->vizinhanca.size() > 0)
361 {
362 int vizinhancaReal = 0;
363
364 for (unsigned int i = 0; i < this->vizinhanca.size(); i++)
365 {
366 // varremos o vetor de vizinhos procurando reconhecimento
367 double canto = this->vizinhanca[i]->getValorCanto();
368
369 if ((canto > this->inicioRangeCanto) && (canto < this->inicioRangeCanto+
       this->rangeCanto))
370 {vizinhancaReal++;}  // vizinho reconhecido
371 }
372
373 if (vizinhancaReal > 0)  // se ao menos um vizinho eh reconhecido
374 {
375 double cantoTotal = 0;          // para somar os cantos
376 int numeroDeCantos = 0;         // numero de cantos armazenados
377
378 // discretizamos o espaco de cantos pra eleger uma moda
```

```
379 vector <vector <double>> faixasDeCanto;
380 unsigned int numeroDeFaixas = 10;
381 double tamanhoDeFaixa = (double) this->rangeCanto/numeroDeFaixas;
382
383 vector <double> listaDeCantos;
384
385 for (unsigned int i = 0; i < numeroDeFaixas; i++)
386 {faixasDeCanto.push_back(listaDeCantos);}
387
388 for (unsigned int i = 0; i < this->vizinhanca.size(); i++)
389 {
390 // varremos a vizinhanca colocando cantos reconhecidos em faixas
391 double canto = this->vizinhanca[i]->getValorCanto();
392
393 if ((canto > this->inicioRangeCanto) && (canto < this->inicioRangeCanto+
       this->rangeCanto))
394 {
395 double cantoNormalizado = canto - this->inicioRangeCanto;
396
397 int faixa = cantoNormalizado/tamanhoDeFaixa;
398
399 faixasDeCanto[faixa].push_back(canto);
400 }
401 }
402
403 // identificamos uma das faixas com maior numero de cantos
404
405 int maiorFaixa = 0;
406
407 for (unsigned int j = 0; j < numeroDeFaixas; j++)
408 {
409 if (faixasDeCanto[j].size() >= faixasDeCanto[maiorFaixa].size())
410 {maiorFaixa = j;}
411 }
412
413 // sepearamos todas as faixas que empatam como maiores
414
415 vector <int> maioresFaixas;
416
417 for (unsigned int j = 0; j < numeroDeFaixas; j++)
418 {
419 if (faixasDeCanto[j].size() == faixasDeCanto[maiorFaixa].size())
420 {maioresFaixas.push_back(j);}
421 }
422
423 // sorteamos uma faixa entre as maiores
424 int maiorFaixaFinal = rand() % maioresFaixas.size();
```

```
425
426 // calculamos a media dos cantos na faixa com mais cantos
427
428 for (unsigned int i = 0; i < faixasDeCanto[maioresFaixas[maiorFaixaFinal
    ]].size(); i++)
429 {
430 cantoTotal += faixasDeCanto[maioresFaixas[maiorFaixaFinal]][i];
431 numeroDeCantos++;
432 }
433
434 double cantoMedio = (cantoTotal/numeroDeCantos);    // calculamos a media
435
436 this->valorCanto = (valorCanto + cantoMedio)/2;    // modificamos o canto
437 }
438 }
439 }
440
441 void agente::limparVizinhanca()
442 {
443 // limpa o vetor de ponteiros da vizinhanca de um agente
444
445 this->vizinhanca.clear();
446 }
447
448 void agente::adicionarVizinhanca(agente *vizinho)
449 {
450 // adiciona o ponteiro de um vizinho ao vetor da vizinhanca
451
452 this->vizinhanca.push_back(vizinho);
453 }
454
455 void agente::atualizaIndice()
456 {
457 // os agentes armazenam indices e se mudaram de patch na ultima iteracao
458
459 this->mudaPatch=false;
460
461 this->indiceXVelho=this->indiceX;   // salvamos o indice
462 this->indiceYVelho=this->indiceY;
463
464 // calculamos novos indices;
465
466 this->indiceX=this->local.getX()*this->indiceMaximo;
467 this->indiceY=this->local.getY()*this->indiceMaximo;
468
469 // testamos se os novos indices sao iguais aos antigos
470
```

```
471 if((this->indiceXVelho!=this->indiceX) || (this->indiceYVelho!=this->
       indiceY))
472 {this->mudaPatch=true;} // caso sim, armazenamos que houve mudanca
473 }
474
475 void agente::condicoesDeContorno()
476 {
477 // aplica as condicoes de contorno
478
479 if(this->local.getX()>=1) this->local.acumulaX(-1);
480 if(this->local.getX()<0)  this->local.acumulaX(1);
481 if(this->local.getY()>=1) this->local.acumulaY(-1);
482 if(this->local.getY()<0)  this->local.acumulaY(1);
483 }
484
485 void agente::namoraComigo(int id)
486 {
487 // eh chamado para o macho quando a femea quer estabelecer um
       relacionamento sério com ele
488
489 this->emRelacionamento = true;
490 this->parceiro = id;
491 }
492
493 void agente::selecionarParceiro()
494 {
495 // usado pelas femeas para selecionar um macho
496
497 if (this->emRelacionamento==false)   // se ela nao esta em relacionamento
498 {
499 // criamos um vetor para sortear qual macho sera avaliado primeiro
500
501 vector <unsigned int> sorteio;
502 for (unsigned int i=0; i<this->vizinhanca.size(); i++)
503 {sorteio.push_back(i);}
504 shuffle(sorteio.begin(),sorteio.end(),std::default_random_engine(1));
505
506 // uma vez sorteado a femea ira avaliar cada um separadamente
507
508 for (unsigned int j=0; j<this->vizinhanca.size(); j++)
509 {
510 if (this->vizinhanca[sorteio[j]]->getEmRelacionamento()==false && this->
       vizinhanca[sorteio[j]]->getEhMacho()==true) // apenas machos solteiros
511 {
512 double cantoAuxiliar=this->vizinhanca[sorteio[j]]->getValorCanto();
513
514 if ((cantoAuxiliar >= this->inicioRangeCanto) && (cantoAuxiliar <= this->
```

```
   inicioRangeCanto + this->rangeCanto)) // apenas com canto reconhecivel
{
double dado = (double)rand()/RAND_MAX;

if (dado < this->probabilidadeAcasalamento) // avaliamos a probabilidade
{
// caso ela resolva acasalar

this->emRelacionamento = true;

// colocamos a ID do parceiro na variavel parceiro
this->parceiro = this->vizinhanca[sorteio[j]]->getId();
this->vizinhanca[sorteio[j]]->namoraComigo(this->id);
// coloca o parceiro em relacionamento com a femea
break;
}}}}}}

void agente::fimDeAno()
{
\\ prepara os agentes para uma nova iteracao

this->flertando = false;
this->contadorDeMeses = 0;

if (this->adulto == true)
{this->idade++;}
}

void agente::fimDeAcasalamento()
{
this->emRelacionamento = false;
this->parceiro = 0;
}

// #### metodos get e set ####

posicao agente::getLocal() const
{return local;}

double agente::getRaioVizinhanca() const
{return raioVizinhanca;}

bool agente::getEhMacho() const
{return ehMacho;}

double agente::getGeneCantoI() const
{return geneCantoI;}
```

```
561
562 double agente::getValorCanto() const
563 {return valorCanto;}
564
565 vector<agente *> agente::getVizinhanca() const
566 {return vizinhanca;}
567
568 bool agente::getAdulto() const
569 {return adulto;}
570
571 void agente::setFlertando(bool value)
572 {flertando = value;}
573
574 int agente::getIdade() const
575 {return idade;}
576
577 bool agente::getEmRelacionamento() const
578 {return emRelacionamento;}
579
580 movimentacao agente::getMovimento() const
581 {return movimento;}
582
583 double agente::getInicioRangeCanto() const
584 {return inicioRangeCanto;}
585
586 double agente::getRangeCanto() const
587 {return rangeCanto;}
588
589 bool agente::getMorto() const
590 {return morto;}
591
592 void agente::setMorto(bool value)
593 {morto = value;}
594
595 int agente::getId() const
596 {return id;}
597
598 int agente::getIndiceX() const
599 {return indiceX;}
600
601 int agente::getIndiceY() const
602 {return indiceY;}
603
604 bool agente::getMudaPatch() const
605 {return mudaPatch;}
606
607 int agente::getVizinhancaSize() const
```

```
608 {return this->vizinhanca.size();
609 }
610 int agente::getIndiceXVelho() const
611 {return indiceXVelho;}
612
613 int agente::getIndiceYVelho() const
614 {return indiceYVelho;}
615
616 int agente::getParceiro() const
617 {return parceiro;}
618
619 void agente::setVizinhanca(const vector<agente *> &value)
620 {vizinhanca = value;}
621
622 double agente::getGeneCantoII() const
623 {return geneCantoII;}
624
625 bool agente::getFlertando() const
626 {return flertando;}
```

B.4 ambiente.h

```
1  #ifndef AMBIENTE_H
2  #define AMBIENTE_H
3  #include "agente.h"       // classe do agente
4  #include <vector>         // para criar vetor de agentes
5  #include <map>            // usado para criar patches
6  #include <utility>
7
8  using namespace std;
9
10 // #### ambiente onde estarao alocados os agentes ####
11
12 class ambiente
13 {
14 public:
15 // #### contrutor do ambiente ####
16
17 ambiente(bool oscines,
18 unsigned int ladoGrade,
19 unsigned int capacidadeDeSuporte,
20 unsigned int tempoDeCristalizacao,
21 unsigned int tempoDeNinho,
22 unsigned int idadeMaxima,
23 unsigned int maximoFilhos,
24 double tamRange,
```

```
25 double passoDosAgentes,
26 double probAcasalamento,
27 double probMutacao,
28 double passoMutacao,
29 double propMigracao,
30 double audicao,
31 double distanciaCompeticao,
32 double probabilidadeDeMorte);
33
34 // metodos publicos
35
36 void rodaGeracaoPatches();          // faz uma iteracao de um ano
37
38 double variacaoEspacial(int lado); // calcula a variabilidade espacial
39
40 // metodos get e set (ambiente2.cpp)
41
42 posicao getLocal(int i);
43 agente *getPonteiroAgente(int i);
44 int getNumeroPassaros();
45 double getTamanhoDoMundo() const;
46 bool getEhMacho(int i);
47 bool getRelacionamentoSerio(int i);
48 bool getAdulto(int i);
49 double getInicioRangeCanto(int i);
50 int getContadorDeGeracoes() const;
51 int getPassos() const;
52 int getTamanhoVizinhanca(int i);
53 vector<int> getVetorDeRanges() const;
54 vector<int> getVetorDeAlelos() const;
55 vector<int> getVetorDeCantos() const;
56 int getIndiceMaximo() const;
57
58 private:
59 // #### propriedades do ambiente ####
60
61 vector <agente> bando;  // vetor de objetos onde são alocados os agentes
62
63 unsigned int contadorDeAgentes=1;
64 // ao criarmos um agente usaremos esse valor como id e somamos +1
65
66 vector <unsigned int> passaro;        // vetor usado para fazer sorteios
67 double tamanhoDoMundo;                // tamanho do mundo (nao usado)
68 unsigned int capacidadeDeSuporte;     // numero inicial de passaros
69 unsigned int contadorDeGeracoes;      // contador de gerações
70 bool epocaDeAcasalamento;             // indica epoca de acasalamento
71 unsigned int mes;                     // mes do ano da simulacao
```

```
72                                      // mes = 12 -> mes de acasalamento
73
74  // #### metodos privados usados no rodaGeracaoPatches (ambiente.cpp) ####
75
76  void rodaAgentes();               // atualizacao assincrona de agentes
77  void atuaPercepcaoPatch(int i);   // atualiza aa vizinhanca do agente i
78  void rodaAcasalamento();          // passaros com parceiro acasalam
79  void acabandoAno();               // acaba a temporada de acasalamento
80  void rodaMortes();                // mata por idade e por competicao
81
82  void mortesPorCompeticao();       // mortes por competicao
83  void mortesPorIdade();            // mortes por idade
84
85  // #### metodos auxiliares (ambiente2.cpp) ####
86
87  double calcularDistancia(int i, int j); //calculos de distancia
88  double distanciaToroide(agente* a1, agente* a2);
89  double distanciaCentro(int i);
90
91  agente *encontraAgente(int id); // retorna o ponteiro pelo id do agente
92
93  // #### variaveis para armazenar as propriedades dos agentes ####
94
95  bool oscines;                 // true = passaros com aprendizado
96  unsigned int tempoDeCristalizacao;  // meses ate a cristalizacao do canto
97  unsigned int tempoDeNinho;  // meses que o filhote fica no ninho
98  unsigned int idadeMaxima;   // idade maxima dos agentes
99  unsigned int maximoFilhos;  // numero maximo de filhos por agente
100 double passoDosAgentes;     // tamanho normalizado do passo do agente
101 double propMigracao;        // proporcao de aumento dos passos do filhote
102 double probAcasalamento;    // probabilidade da acasalamento
103 double probMutacao;         // probabilidade de ocorrer mutacao
104 double passoMutacao;         // tamanho da mutacao
105 double tamRange;             // tamanho do range do canto
106 double audicao;              // audicao dos passaros
107 double distanciaCompeticao;  // usado para calcular o raio da morte
108
109 double raioDeVizinhanca;     // audicao vezes o passo
110 double raioDaMorte;          // raio de competicao vezes o passo
111
112 double probabilidadeDeMorte; // usado para morte por competicao
113
114 // #### metodos e variaveis que dos patches (ambientespace.cpp) ####
115
116 map <pair<int,int>,map<int, agente*>> grid; // mapa com patches e agentes
117
118 int indiceMaximo;          // maior valor de indice x ou y
```

```
119 int distanciaIndice;        // determinado pelo raio de vizinhanca
120 int distanciaIndiceMorte;  // determinado pelo raio de competicao
121
122 void fill_grid (int L);    // preenche o grid com todos os ponteiros
123
124 void add (agente* ag, int X, int Y);    // adiciona um agente ao grid
125 void remove_from_cell (agente* ag);     // remove um agente do grid
126
127 pair<int, int> get_search_cell(int x_in, int y_in);
128 // determina qual a celula vizinha em um mundo toroidal
129
130 // o metodo abaixo retorna a vizinhanca de um agente usando os patches
131
132 vector<agente *> Range_query(agente *ag1, double range, double raioViz);
133
134 int tamVizinhancaPatch (agente *ag1, double range, double raioViz);
135 };
136
137 #endif // AMBIENTE_H
```

B.5 ambiente.cpp

```
1  #include "ambiente.h"
2  #include <math.h>
3  #include <algorithm>
4  #include <map>
5
6  // #### construtor, iterador e os principais metodos ####
7
8  ambiente::ambiente(bool oscines,
9  unsigned int ladoGrade,
10 unsigned int capacidadeDeSuporte,
11 unsigned int tempoDeCristalizacao,
12 unsigned int tempoDeNinho,
13 unsigned int idadeMaxima,
14 unsigned int maximoFilhos,
15 double tamRange,
16 double passoDosAgentes,
17 double probAcasalamento,
18 double probMutacao,
19 double passoMutacao,
20 double propMigracao,
21 double audicao,
22 double distanciaCompeticao,
23 double probabilidadeDeMorte)
24 {
```

```
25 // armazenando as caracteristicas dos agentes passadas ao construtor
26
27 this->passoDosAgentes = passoDosAgentes;
28 this->propMigracao = propMigracao;
29 this->probAcasalamento = probAcasalamento;
30 this->probMutacao = probMutacao;
31 this->passoMutacao = passoMutacao;
32 this->tamRange = tamRange;
33 this->tempoDeCristalizacao = tempoDeCristalizacao;
34 this->tempoDeNinho = tempoDeNinho;
35 this->oscines = oscines;
36 this->audicao = audicao;
37 this->distanciaCompeticao = distanciaCompeticao;
38
39 this->raioDeVizinhanca = this->audicao * this->passoDosAgentes;
40 this->raioDaMorte = this->distanciaCompeticao * this->passoDosAgentes;
41
42 this->probabilidadeDeMorte = probabilidadeDeMorte;
43
44 // #### inicializamos as variaveis do ambiente ####
45
46 this->tamanhoDoMundo = 1000;
47 this->capacidadeDeSuporte = capacidadeDeSuporte;
48 // capacidadeDeSuporte na verdade indica a quantidade inicial
49 this->contadorDeGeracoes = 0;
50 this->epocaDeAcasalamento = false;
51 this->idadeMaxima = idadeMaxima;
52 this->maximoFilhos = maximoFilhos;
53 this->mes = 1;
54
55 // reservamos memoria para o vetor do bando
56
57 this->bando.reserve(500000);
58
59 // inicializamos os indices dos patches
60
61 this->indiceMaximo = ladoGrade;
62 this->distanciaIndice = this->indiceMaximo * this->raioDeVizinhanca;
63
64 double corrigindo = (double)this->indiceMaximo * this->raioDeVizinhanca;
65 if ((corrigindo - (double)this->distanciaIndice) > 0)
66 {this->distanciaIndice++;}
67
68 this->distanciaIndiceMorte = this->indiceMaximo * this->raioDaMorte;
69
70 corrigindo = (double)this->indiceMaximo * this->raioDaMorte;
71 if ((corrigindo - (double)this->distanciaIndiceMorte) > 0)
```

```
72 {this->distanciaIndiceMorte++;}
73
74 // #### criando os agentes ####
75
76 this->bando.clear();
77
78 for (unsigned int i = 0; i < (this->capacidadeDeSuporte/2); i++)
79 {
80 agente piupiu(this->contadorDeAgentes, true, this->oscines, this->
      tamRange, this->passoDosAgentes, this->propMigracao, this->
      probAcasalamento, this->probMutacao, this->passoMutacao, this->
      tempoDeCristalizacao, this->tempoDeNinho, this->idadeMaxima, this->
      indiceMaximo, this->audicao);
81 this->bando.push_back(piupiu);
82
83 this->contadorDeAgentes++;
84 }
85
86 for (unsigned int i = 0; i < (this->capacidadeDeSuporte/2); i++)
87 {
88 agente piupiu(this->contadorDeAgentes, false, this->oscines, this->
      tamRange, this->passoDosAgentes, this->propMigracao, this->
      probAcasalamento, this->probMutacao, this->passoMutacao, this->
      tempoDeCristalizacao, this->tempoDeNinho, this->idadeMaxima, this->
      indiceMaximo, this->audicao);
89 this->bando.push_back(piupiu);
90
91 this->contadorDeAgentes++;
92 }
93
94 this->passaro.clear();   // limpamos o vetor de sorteio
95
96 for (unsigned int i = 0; i < this->bando.size(); i++)
97 {this->passaro.push_back(i);} // preenchendo o vetor de sorteio
98
99 this->fill_grid(this->indiceMaximo);
100 }
101
102 void ambiente::rodaGeracaoPatches()
103 {
104 for(this->mes = 1; this->mes <= 12; this->mes++)
105 {
106 // cada geracao corresponde a 1 ano
107 // o 12 mes do ano corresponde a epoca de acasalamento
108
109 if (this->mes == 12) this->epocaDeAcasalamento = true;
110
```

```
111 if (epocaDeAcasalamento == false)
112 {this->rodaAgentes();}
113 else
114 {
115 this->rodaMortes();
116 this->rodaAcasalamento();
117
118 this->epocaDeAcasalamento = false;
119 this->contadorDeGeracoes++;
120 }
121 }
122 this->acabandoAno();
123 }
124
125 void ambiente::rodaAgentes()
126 {
127 // embaralhamos o vetor de sorteio
128
129 shuffle(passaro.begin(),passaro.end(),std::default_random_engine(1));
130
131 // cada agente eh atualizado seguindoa ordem
132
133 for (unsigned int j = 0; j < this->bando.size(); j++)
134 {
135 this->atuaPercepcaoPatch(this->passaro[j]);       // percepcao
136 this->bando[this->passaro[j]].rodaModelo();       // modelo de mundo
137 this->bando[this->passaro[j]].atuacao();          // atuacao
138
139 // caso o agente tenha mudado de patch atualizamos sua celula no grid
140
141 if (this->bando[passaro[j]].getMudaPatch() == true)
142 {
143 int novoX = this->bando[passaro[j]].getIndiceX();
144 int novoY = this->bando[passaro[j]].getIndiceY();
145
146 this->remove_from_cell(&bando[passaro[j]]);
147 this->add(&bando[passaro[j]],novoX,novoY);
148 }}}
149
150 void ambiente::atuaPercepcaoPatch(int i)
151 {
152 this->bando[i].limparVizinhanca();    // limpamos a vizinhança
153
154 if ((this->epocaDeAcasalamento == true) && (this->bando[i].getAdulto() ==
        true))
155 {this->bando[i].setFlertando(true);}  // em busca de acasalamento
156
```

```
157 if (((this->bando[i].getEhMacho() == true) && (this->bando[i].getAdulto()
        == false)) || ((this->bando[i].getEhMacho() == false) && (this->bando
       [i].getFlertando() == true) && (this->bando[i].getEmRelacionamento()
       == false)))
158 {
159 // ouvir cantos so eh relevante para machos filhotes e femeas solteiras
160 this->bando[i].setVizinhanca(this->Range_query(&bando[i], this->
       distanciaIndice, this->raioDeVizinhanca));
161 }
162 }
163
164 void ambiente::rodaAcasalamento()
165 {
166 // primeiramente fazemos com que sejam escolhidos os parceiros
167
168 this->rodaAgentes(); // atualiza percepcao e escolhe parceiros
169
170 // cada femea em um relacionamento gera filhotes
171
172 unsigned int tamanhoAtual = this->bando.size();
173
174 for (unsigned int i = 0; i < tamanhoAtual; i++)       // para cada agente
175 {
176 if (this->bando[i].getEhMacho() == false)             // femeas
177 {
178 if(this->bando[i].getEmRelacionamento() == true)   // em relacionamento
179 {
180 // usamos o construtor de agentes que criamos para reproducao sexuada
181
182 agente *parceiro = this->encontraAgente(this->bando[i].getParceiro());
183 // pegamos os valores necessarios para criacao do filhote
184
185 double geneMaeI  = this->bando[i].getGeneCantoI();
186 double geneMaeII = this->bando[i].getGeneCantoII();
187 double genePaiI  = parceiro->getGeneCantoI();
188 double genePaiII = parceiro->getGeneCantoII();
189 posicao localMae = this->bando[i].getLocal();
190
191 // vamos criar aleatoriamente macho e femeas
192
193 int filhos = rand()%this->maximoFilhos; // determinamos numero de filhos
194
195 double dado = 0;
196 bool ehMacho = false;
197
198 for (int j = 0; j < filhos; j++) // para cada filho
199 {
```

```
200 dado = (double)rand()/RAND_MAX;
201
202 if (dado < 0.5)      // determinando se macho ou femea
203 {ehMacho = true;}
204 else
205 {ehMacho = false;}
206
207 agente piupiu(this->contadorDeAgentes, ehMacho, this->oscines, this->
       tamRange, this->passoDosAgentes, this->propMigracao, this->
       probAcasalamento, this->probMutacao, this->passoMutacao, this->
       tempoDeCristalizacao, this->tempoDeNinho, this->indiceMaximo, this->
       audicao, geneMaeI, geneMaeII, genePaiI, genePaiII, localMae);
208
209 this->bando.push_back(piupiu);           // acrescentando ao vetor
210
211 this->contadorDeAgentes++;
212 }}}}
213
214 // preenchemos novamente o grid pois foram adicionados agentes
215
216 this->fill_grid(this->indiceMaximo);
217
218 // criamos novamente um vetor de sorteio pois foram adicionados agentes
219
220 this->passaro.clear();   // limpamos o vetor de sorteio
221
222 for (unsigned int i = 0; i < this->bando.size(); i++)
223 {this->passaro.push_back(i);}
224 }
225
226 void ambiente::rodaMortes()
227 {
228 // mortesPorIdade e mortesPorCompeticao assinalam as mortes
229
230 this->mortesPorIdade();
231 this->mortesPorCompeticao();
232
233 // colocamos entao todos os passaros mortos no fim do vetor
234
235 sort(this->bando.begin(),this->bando.end(), [](const agente& lhs, const
       agente& rhs){return lhs.getMorto() < rhs.getMorto();});
236
237 // agora precisamos deleta-los
238
239 unsigned int primeiroMorto = 0;  // vamos determinar o primeiro morto
240
241 for (unsigned int i = 0; i < this->bando.size(); i++)
```

```
{
if (this->bando[i].getMorto() == false)
{primeiroMorto++;}
else{break;}
}

// encerramos os relacionamentos de passaros vivos com conjes mortos

for (unsigned int j = 0; j < primeiroMorto; j++)
{
if (this->bando[j].getEmRelacionamento() == true)
{
agente *parceiro = this->encontraAgente(this->bando[j].getParceiro());

if (parceiro->getMorto() == true)
{this->bando[j].fimDeAcasalamento();}
}
}

// apagando os agentes mortos a partir do primeiro

this->bando.erase(this->bando.begin() + primeiroMorto,this->bando.end());

this->fill_grid(this->indiceMaximo);  // repovoamos o grid

// vamos sortear novamente o vetor de sorteio

this->passaro.clear();   // limpamos o vetor de sorteio

for (unsigned int m = 0; m < this->bando.size(); m++)
{this->passaro.push_back(m);}
}
```

B.6 ambiente2.cpp

```
#include "ambiente.h"
#include <math.h>
#include <algorithm>

// #### METODOS AUXILIARES ####

void ambiente::acabandoAno()
{
for (unsigned int i = 0; i < this->bando.size(); i++)
{this->bando[i].fimDeAno();}   // preparando pra proxima iteracao
}
```

```
12
13 void ambiente::mortesPorCompeticao()
14 {
15 // assinala as mortes que ocorrerao por competicao
16
17 double dado;
18 int tamVizinhanca=0;
19 double probabilidadeMorte;
20
21 // a morte sera assincrona, por isso embaralhamos o vetor de sorteio
22
23 shuffle(passaro.begin(),passaro.end(),std::default_random_engine(1));
24
25 // avaliamos se cada agente morre na sequencia
26
27 for (unsigned int j = 0; j < this->bando.size(); j++)
28 {
29 if (this->bando[passaro[j]].getMorto() == false)  // caso vivo
30 {
31 // checamos o numero de passaros vivos na vizinhanca e calculamos
      probabilidadeMorte
32
33 tamVizinhanca = this->tamVizinhancaPatch(&bando[passaro[j]],this->
      distanciaIndiceMorte,this->raioDaMorte);
34
35 probabilidadeMorte = this->probabilidadeDeMorte * tamVizinhanca;
36
37 dado = (double)rand()/RAND_MAX;
38
39 if(dado < probabilidadeMorte)  // avaliamos se o agente morre
40 {
41 this->bando[passaro[j]].setMorto(true);
42 }}}}
43
44 void ambiente::mortesPorIdade()
45 {
46 // assinala as mortes que ocorrerao por idade
47
48 for (unsigned int i = 0; i < this->bando.size(); i++)
49 {
50 if (this->bando[i].getIdade() >= idadeMaxima)
51 {
52 this->bando[i].setMorto(true);
53 }}}
54
55 double ambiente::calcularDistancia(int i, int j)
56 {
```

```
57 // calcularDistancia e distanciaToroide fazem a mesma coisa
58 // mas com parametros diferentes
59
60 double distancia=0;      // distância entre dois agentes
61
62 posicao localI,localJ;   // objetos "local" de ambos os agentes
63
64 localI=this->bando[i].getLocal();
65 localJ=this->bando[j].getLocal();
66
67 double x1=localI.getX();
68 double x2=localJ.getX();
69 double y1=localI.getY();
70 double y2=localJ.getY();
71
72 distancia=sqrt(min(abs(x1-x2),1-abs(x1-x2))*min(abs(x1-x2),1-abs(x1-x2))+
73 min(abs(y1-y2),1-abs(y1-y2))*min(abs(y1-y2),1-abs(y1-y2)));
74
75 return distancia;
76 }
77
78 double ambiente::distanciaToroide(agente *a1, agente *a2)
79 {
80 // calcularDistancia e distanciaToroide fazem a mesma coisa
81 // mas com parametros diferentes
82
83 posicao local1=a1->getLocal();
84 posicao local2=a2->getLocal();
85
86 double x1=local1.getX();
87 double x2=local2.getX();
88 double y1=local1.getY();
89 double y2=local2.getY();
90
91 double distancia=sqrt(min(abs(x1-x2),1-abs(x1-x2))*min(abs(x1-x2),1-abs(
      x1-x2))+
92 min(abs(y1-y2),1-abs(y1-y2))*min(abs(y1-y2),1-abs(y1-y2)));
93
94 return distancia;
95 }
96
97 double ambiente::distanciaCentro(int i)
98 {
99 // calcula a distancia entre um agente e o ponto (0.5,0.5)
100
101 double distancia=0;      // distância entre dois agentes
102
```

```
103 posicao localI,localJ;   // objetos "local" de ambos os agentes
104
105 localJ.setX(0.5);
106 localJ.setY(0.5);
107
108 localI=this->bando[i].getLocal();
109
110 double deltaX,deltaY; // váriaveis auxíliares para o calculo da distância
111
112 deltaX=localI.getX()-localJ.getX();
113 deltaY=localI.getY()-localJ.getY();
114
115 distancia=sqrt(deltaX*deltaX+deltaY*deltaY);
116
117 return distancia;
118 }
119
120 double ambiente::variacaoEspacial(int lado)
121 {
122 // retorna a variabilidade espacial
123
124 // lado eh o numero de caixas em um dos lados do mundo
125 // lado deve ser divisor de indiceMaximo
126
127 int tamanhoDosIntervalos = this->indiceMaximo/lado;
128
129 vector <double> numeroDeIndividuos;
130 numeroDeIndividuos.reserve(lado*lado);
131
132 for (int i = 0; i < lado; i++) // indices das caixas
133 {
134 for(int j = 0; j < lado; j++)
135 {
136 // nessa caixa:
137 int contador = 0;
138
139 for (int m = 0; m < tamanhoDosIntervalos; m++) // indices dos patches
140 {
141 for (int n = 0; n < tamanhoDosIntervalos; n++)
142 {
143 // nesse patch
144
145 pair<int,int> celula = make_pair(m + i*tamanhoDosIntervalos,n + j*
       tamanhoDosIntervalos);
146
147 map<int,agente*> listaDeAgentes = grid.at(celula);
148
```

```
149 contador += listaDeAgentes.size();
150 // somamos o numero de agentes no cantador da caixa
151 }}
152 // armazenamos o numero de agentes na caixa
153 numeroDeIndividuos.push_back((double)contador);
154 }}
155
156 // com vetor numeroDeIndividuos calculamos a media entre as caixas
157 double acumulador = 0;
158
159 for (int i = 0; i < numeroDeIndividuos.size(); i++)
160 {acumulador += numeroDeIndividuos[i];}
161
162 double media = (double)(acumulador/numeroDeIndividuos.size());
163
164 // com media e vetor numeroDeIndividuos calculamos o desvio padrao
165
166 acumulador = 0;
167
168 for (int i = 0; i < numeroDeIndividuos.size(); i++)
169 {acumulador += ((numeroDeIndividuos[i]-media)*(numeroDeIndividuos[i]-
       media));}
170
171 double dispersao = (double)(acumulador/numeroDeIndividuos.size());
172
173 double desvioPadrao = sqrt(dispersao);
174
175 // a variabilidade eh dada pelo desvio relativo
176
177 double desvioRelativo = desvioPadrao/media;
178
179 return desvioRelativo;
180 }
181
182 agente *ambiente::encontraAgente(int id)
183 {
184 // encontra o ponteiro de um agente pela sua id
185
186 agente *bird=nullptr;
187
188 for (unsigned int i = 0; i < this->bando.size(); i++)
189 {
190 if (this->bando[i].getId() == id)
191 {
192 bird = &this->bando[i];
193 break;
194 }}
```

```
195 return bird;
196 }
197
198 // #### METODOS GET E SET ####
199
200 posicao ambiente::getLocal(int i)
201 {
202 posicao local=this->bando[i].getLocal();
203 local.setX(local.getX()*this->tamanhoDoMundo);
204 local.setY(local.getY()*this->tamanhoDoMundo);
205 return local;
206 }
207
208 agente *ambiente::getPonteiroAgente(int i)
209 {
210 agente *ponteiro = &this->bando[i];
211 return ponteiro;
212 }
213
214 int ambiente::getNumeroPassaros()
215 {return this->bando.size();}
216
217 double ambiente::getTamanhoDoMundo() const
218 {return tamanhoDoMundo;}
219
220 bool ambiente::getEhMacho(int i)
221 {return this->bando[i].getEhMacho();}
222
223 bool ambiente::getRelacionamentoSerio(int i)
224 {return this->bando[i].getEmRelacionamento();}
225
226 bool ambiente::getAdulto(int i)
227 {return this->bando[i].getAdulto();}
228
229 double ambiente::getInicioRangeCanto(int i)
230 {return this->bando[i].getInicioRangeCanto();}
231
232 int ambiente::getContadorDeGeracoes() const
233 {return contadorDeGeracoes;}
234
235 vector<int> ambiente::getVetorDeRanges() const
236 {return vetorDeRanges;}
237
238 vector<int> ambiente::getVetorDeAlelos() const
239 {return vetorDeAlelos;}
240
241 vector<int> ambiente::getVetorDeCantos() const
```

```
242 {return vetorDeCantos;}
243
244 int ambiente::getIndiceMaximo() const
245 {return indiceMaximo;}
246
247 int ambiente::getPassos() const
248 {return passos;}
249
250 int ambiente::getTamanhoVizinhanca(int i)
251 {return this->bando[i].getVizinhancaSize();}
```

B.7 ambientespace.cpp

```
1  #include "ambiente.h"
2  #include <math.h>
3  #include <algorithm>
4
5  // #### metodos relacionados aos patches ####
6
7  void ambiente::fill_grid(int L)
8  {
9  // metodo coloca todos os agentes na celula correta do grid
10
11 this->grid.clear();  // limpando o grid
12
13 for (int x = 0; x<L; x++)  // criando novamente os elementos do grid
14 {
15 for (int y = 0; y<L; y++)
16 {
17 pair<int,int>coord = make_pair(x,y);       // criando chaves (x,y)
18 map<int, agente*> list;                    // criando lista de ponteiros
19
20 // criando a celula
21 pair<pair<int,int>,map<int,agente*>> cell = make_pair(coord,list);
22
23 this->grid.insert(cell);                   // inserimos a celula no mapa
24 }}
25
26 for (unsigned int i=0; i<this->bando.size(); i++)  // povoando o grid
27 {
28 add(this->getPonteiroAgente(i),
29 this->bando[i].getIndiceX(),this->bando[i].getIndiceY());
30 }
31 }
32
33 vector<agente *> ambiente::Range_query(agente *ag1, double range, double
```

```
   raioViz)
{
// retorna a vizinhanca de um determinado agente

// nomeia essa celula como center_cell
pair<int,int> center_cell=make_pair(ag1->getIndiceX(),ag1->getIndiceY());

vector <agente*> neighbors; //cria vetor de vizinhos, que vamos preencher

for (int x = (center_cell.first - range); x <= (center_cell.first + range
   ); x++)  // para cada x possivel
{
for (int y = (center_cell.second - range); y <= (center_cell.second +
   range); y++)  // para cada y
{
// verificamos as condicoes de contorno
pair<int,int> search_cell = get_search_cell(x,y);

// criamos um mapa com os agentes da celula
map<int, agente*> members = grid.at(search_cell);

for (auto it : members)  // para cada agente na celula
{
agente* ag2 = it.second;          // chamamos ele de ag2

if (ag1->getId()==ag2->getId()) continue; // nao sendo o proprio agente

if (ag2->getEhMacho() == true && ag2->getAdulto() == true)
{
// caso seja macho adulto checamos a distancia

double d = this->distanciaToroide(ag1,ag2);
if (d <= raioViz) {neighbors.push_back(ag2);}             // se estiver
   dentro do range acrescentamos a vizinhanca
}}}}
return neighbors;
}

int ambiente::tamVizinhancaPatch(agente *ag1, double range, double
   raioViz)
{
// retorna o tamanho da vizinhanca de um determinado agente

// nomeia essa celula como center_cell
pair<int,int> center_cell=make_pair(ag1->getIndiceX(),ag1->getIndiceY());

int contadorVizinhos = 0; // para contabilizar os vizinhos
```

```
76
77 for (int x = (center_cell.first - range); x <= (center_cell.first + range
      ); x++)  // para cada x possivel
78 {
79 for (int y = (center_cell.second - range); y <= (center_cell.second +
      range); y++)  // para cada y
80 {
81 // verificamos as condicoes de contorno
82 pair<int,int> search_cell = get_search_cell(x,y);
83
84 // criamos um mapa com os agentes da celula
85 map<int,agente*> members = grid.at(search_cell);
86
87 for (auto it : members)  // para cada agente na celula
88 {
89 agente* ag2 = it.second;          // chamamos ele de ag2
90
91 if (ag1->getId() == ag2->getId()) continue; // nao sendo o proprio agente
92
93 if (ag2->getMorto() == false)  // caso vivo
94 {
95 double d = this->distanciaToroide(ag1,ag2);    // checamos a distancia
96 if (d <= raioViz) {contadorVizinhos++;} // se for menor contabilizamos
97 }}}}
98 return contadorVizinhos;
99 }
100
101 void ambiente::add(agente *ag, int X, int Y)
102 {
103 // adiciona o agente a celula especificada
104
105 pair <int, int> xy = make_pair(X,Y);     // cria um par para a coordenada
106 pair <int,agente*> a = make_pair(ag->getId(),ag); // cria par id,ponteiro
107 this->grid.at(xy).insert(a);             // acrescenta o agente ao grid
108 }
109
110 void ambiente::remove_from_cell(agente *ag)
111 {
112 // remove o agente da celula
113
114 // descobrindo a celula do agente
115 pair<int,int> xy=make_pair(ag->getIndiceXVelho(),ag->getIndiceYVelho());
116 int id = ag->getId();          // descobrindo a id do agente
117 this->grid.at(xy).erase(id);  // removendo agente
118 }
119
120 pair<int, int> ambiente::get_search_cell(int x_in, int y_in)
```

```
{
// diz qual a celula na qual iremos pesquisar
// necessario para especificar as condicoes de contorno

int x_out = x_in;  // x e y de saida inicializados iguais aos de entrada
int y_out = y_in;

// para celulas nos limites do mundo temos condicoes de contorno (toro)

if(x_in < 0)                    {x_out = this->indiceMaximo - abs(x_in);}
if(x_in >= this->indiceMaximo) {x_out = x_in - this->indiceMaximo;}

if(y_in < 0)                    {y_out = this->indiceMaximo - abs(y_in);}
if(y_in >= this->indiceMaximo) {y_out = y_in - this->indiceMaximo;}

pair<int,int> result_cell = make_pair(x_out,y_out);

return result_cell;  // retornamos a celula correta
}
```

B.8 movimentacao.h

```
#ifndef MOVIMENTACAO_H
#define MOVIMENTACAO_H

#include <random> // para implementar um gerador de numeros aleatorios
#include <math.h>

using namespace std;

// encapsula as regras de movimentação do agente
// calcula dx e dy para somar a sua atual posicao

class movimentacao
{
public:
movimentacao();
void movimento();          // calcula o novo dx e o novo dy
void movimentoMigracao(); // calcula o novo dx e o novo dy para filhotes
void parado();             // impoe dx=0 e dy=0

double getDx() const;
double getDy() const;

// metodos para setar os valores dos parametros de movimento

```

```
25 void setDirecao(double value);
26 void setAnguloDeVisao(double value);
27
28 void setPasso(double value);
29 double getPasso() const;
30
31 double getPassoMigracao() const;
32 void setPassoMigracao(double value);
33
34 double getAnguloDeMigracao() const;
35 void setAnguloDeMigracao(double value);
36
37 private:
38 double dx,dy;              // são somadas ao posicionamento do agente
39
40 // movimento browniano correlacionado de passo variavel
41
42 double direcao;            // a direção para qual o passo deve ser dado
43 double anguloDeVisao;      // o angulo de visao do agente
44 double anguloDeMigracao;   // angulo de visao para machos jovens
45 double passo;              // o tamanho do passo
46 double passoMigracao;      // o tamanho do passo de migracao
47
48 // para distribuicao gaussiana no tamanho dos passos
49
50 default_random_engine gerador;
51 };
52
53 #endif // MOVIMENTACAO_H
```

B.9 movimentacao.cpp

```
1 #include "movimentacao.h"
2 #include <math.h>
3 #include <stdlib.h>
4 #include <random>
5
6 #define PI 3.1416    // o valor de PI eh usado no metodo "movimento"
7
8 movimentacao::movimentacao()
9 {
10 // inicializando todas as variaveis
11 // no construtor de cada agente ele deve indicar os valores de cada uma
12
13 this->anguloDeVisao=0;
14 this->anguloDeMigracao=0;
```

```
15 this->direcao=0;
16 this->passo=0;
17 this->dx=0;
18 this->dy=0;
19
20 gerador.seed(rand());
21 }
22
23 void movimentacao::movimento()
24 {
25 // calcula um novo dx e um novo dy
26 // movimento browniano correlacionado de passo variavel
27
28 normal_distribution <double> fatorPasso(0,1);
29
30 double variacaoDeDirecao = (double)rand()/RAND_MAX;
31 this->direcao += (variacaoDeDirecao - 0.5) * this->anguloDeVisao;
32
33 double fator = fabs(fatorPasso(this->gerador));
34
35 this->dx = this->passo * fator * cos(this->direcao * PI/180);
36 this->dy = this->passo * fator * sin(this->direcao * PI/180);
37 }
38
39 void movimentacao::movimentoMigracao()
40 {
41 // calcula um novo dx e um novo dy
42 // movimento browniano correlacionado de passo variavel
43 // passos maiores e angulo de visao menor
44
45 normal_distribution <double> fatorPasso(0,1);
46
47 double variacaoDeDirecao = (double)rand()/RAND_MAX;
48 this->direcao += (variacaoDeDirecao - 0.5) * this->anguloDeMigracao;
49
50 double fator = fabs(fatorPasso(this->gerador));
51
52 this->dx = this->passoMigracao * fator * cos(this->direcao * PI/180);
53 this->dy = this->passoMigracao * fator * sin(this->direcao * PI/180);
54 }
55
56 void movimentacao::parado()
57 {
58 this->dx=0;    // seta o dx e o dy como zero
59 this->dy=0;
60 }
61
```

```
62 double movimentacao::getDx() const
63 {return dx;}
64
65 double movimentacao::getDy() const
66 {return dy;}
67
68 void movimentacao::setDirecao(double value)
69 {direcao = value;}
70
71 void movimentacao::setAnguloDeVisao(double value)
72 {anguloDeVisao = value;}
73
74 void movimentacao::setPasso(double value)
75 {passo = value;}
76
77 double movimentacao::getPasso() const
78 {return passo;}
79
80 double movimentacao::getPassoMigracao() const
81 {return passoMigracao;}
82
83 void movimentacao::setPassoMigracao(double value)
84 {passoMigracao = value;}
85
86 double movimentacao::getAnguloDeMigracao() const
87 {return anguloDeMigracao;}
88
89 void movimentacao::setAnguloDeMigracao(double value)
90 {anguloDeMigracao = value;}
```

B.10 posicao.h

```
1  #ifndef POSICAO_H
2  #define POSICAO_H
3
4  // armazena a posicao do agente
5
6  class posicao
7  {
8  public:
9  posicao();
10
11 void acumulaX (double dx);  // adiciona os valores de dx e dy a x e y
12 void acumulaY (double dy);
13
14 double getX() const;
```

```
15 double getY() const;
16
17 void setX(double value);
18 void setY(double value);
19
20 private:
21 double x,y;     // coordenadas x e y do posicionamento do agente
22 };
23
24 #endif // POSICAO_H
```

B.11 posicao.cpp

```
1  #include "posicao.h"
2
3  posicao::posicao()
4  {
5
6  }
7
8  void posicao::acumulaX(double dx)
9  {this->x+=dx;}
10
11 void posicao::acumulaY(double dy)
12 {this->y+=dy;}
13
14 double posicao::getX() const
15 {return x;}
16
17 double posicao::getY() const
18 {return y;}
19
20 void posicao::setX(double value)
21 {x = value;}
22
23 void posicao::setY(double value)
24 {y = value;}
```