ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Lucas Moreno de Araujo

DEFINIÇÃO DE SEQUÊNCIAS DE ESTUDO COM BASE NO *ANT SYSTEM* E EM INFORMAÇÕES PRESENTES EM OBJETOS DE APRENDIZAGEM

Palmas
2012

ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Lucas Moreno de Araujo

DEFINIÇÃO DE SEQUÊNCIAS DE ESTUDO COM BASE NO *ANT SYSTEM* E EM INFORMAÇÕES PRESENTES EM OBJETOS DE APRENDIZAGEM

Trabalho apresentado como requisito parcial das disciplinas de Trabalho de Conclusão de Curso I e II (TCC I e TCC II) do curso de Sistemas de Informação, orientado pelo Professor Mestre Fabiano Fagundes.

Palmas
2012

ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Lucas Moreno de Araujo

DEFINIÇÃO DE SEQUÊNCIAS DE ESTUDO COM BASE NO *ANT SYSTEM* E EM INFORMAÇÕES PRESENTES EM OBJETOS DE APRENDIZAGEM

Trabalho apresentado como requisito parcial das disciplinas de Trabalho de Conclusão de Curso I e II (TCC I e TCC II) do curso de Sistemas de Informação, orientado pelo Professor Mestre Fabiano Fagundes.

Aprovada em de 2012.

BANCA EXAMINADORA

_____

Prof. M. Sc. Fabiano Fagundes

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Fernando Luiz de Oliveira

Centro Universitário Luterano de Palmas

_____

Prof. M. Sc. Parcilene Fernandes Brito

Centro Universitário Luterano de Palmas

Palmas
2012

SUMÁRIO

1. INTRODUÇÃO 1
1.1. Objetivos 3
1.1.1. Objetivo Geral 3
1.1.2. Objetivos Específicos 3
1.2. Justificativa 4
1.3. Problema 5
1.4. Hipóteses 6
2. REFERENCIAL TEÓRICO 7
2.1. Informática na educação 7
2.2. Objetos de Aprendizagem 9
2.2.1. Learning Object Metadata 11
2.2.1.1. Geral 12
2.2.1.2. Ciclo de Vida 15
2.2.1.3. Meta-metadados 16
2.2.1.4. Técnico 17
2.2.1.5. Educacional 19
2.2.1.6. Direito 22
2.2.1.7. Relacionamento 23
2.2.1.8. Anotações 24
2.2.1.9. Classificação 24
2.3. Representação de usuários 26
2.3.1. Modelo de usuário em um contexto educacional 29
2.3.1.1. Modelo do estudante 30
2.3.1.2. Módulo de diagnóstico 31
2.4. Ant system 34
2.4.1. Inspiração biológica 34

2.4.2. Abstração das formigas para a computação 37
2.4.2.1. Os elementos fundamentais 38
2.4.2.2. Estrutura do algoritmo 40
3. MATERIAIS E MÉTODOS 44
3.1. Materiais 44
3.2. Metodologia 44
4. RESULTADOS E DISCUSSÃO 47
4.1. Diagrama de classes 47
4.2. Classes do algoritmo AntStudy 48
4.2.1. Classe Grafo 48
4.2.2. Classe Aresta 55
4.2.3. Classe Vertice 57
4.2.4. Classe ObjetoAprendizagem 57
4.2.5. Classe Relacionamento 58
4.2.6. Classe Usuario 58
4.2.7. Classe AntSystem 62
4.3. Exemplo de funcionamento do algoritmo 69
5. CONSIDERAÇÕES FINAIS 76
6. REFERÊNCIAS BIBLIOGRÁFICAS 78

RESUMO

O objetivo geral deste trabalho é propiciar a definição de sequências de estudo a partir de um conjunto de objetos de aprendizagem descritos por metadados, levando-se em consideração as características do usuário que está utilizando tais recursos. As principais tecnologias envolvidas nesse contexto são: LOM – *Learning Object Metadata*, fornecendo a estrutura de metadados para a descrição do domínio e o *Ant System*, algoritmo da Inteligência Artificial baseado no comportamento das formigas para a definição de caminhos. Após o estudo dessas tecnologias, foi desenvolvido um algoritmo, que recebeu o nome de AntStudy. Após o desenvolvimento, foi criado um exemplo contendo usuários e objetos de aprendizagem fictícios, para mostrar o funcionamento do algoritmo. Nesse exemplo, constatou-se que a quantidade de feromônio decrescia rapidamente, portanto, seria necessário balancear o incremento ou evaporação do feromônio no grafo, para possibilitar a utilização do algoritmo em um ambiente real. Os resultados indicaram que é possível utilizar a técnica do *Ant System* para o estabelecimento de sequências de estudo.

PALAVRAS CHAVE: LOM, Ant System, sequências de estudo.

LISTA DE FIGURAS

Figura 1: Categorias do LOM 11
Figura 2: Metadados da categoria Geral 12
Figura 3: Metadados da categoria Ciclo de Vida 15
Figura 4: Metadados da categoria Meta-metadados 16
Figura 5: Metadados da categoria Técnico 18
Figura 6: Metadados da categoria Educacional 20
Figura 7: Metadados da categoria Direito 22
Figura 8: Metadados da categoria Relacionamento 23
Figura 9: Metadados da categoria Anotações 24
Figura 10: Metadados da categoria Classificação 25
Figura 11: Exemplos de estereótipos no curso de Sistemas de Informação 28
Figura 12: Procedimentos do módulo de diagnóstico 32
Figura 13: Pontes iguais (DORIGO e STÜZLE, 2004, p. 3, adaptado) 35
Figura 14: Pontes distintas (DORIGO e STÜZLE, 2004, p. 3, adaptado) 36
Figura 15: Estrutura para armazenar a quantidade de feromônio 39
Figura 16: Pseudocódigo do algoritmo Ant System 40
Figura 17: Diagrama de classes do AntStudy 48
Figura 18: Exemplo de matriz de adjacência e respectivo grafo 50
Figura 19: Método criarRelExplicito() 51
Figura 20: Métodos criarRelImplicito() 52
Figura 21: Método criarVertices() 52
Figura 22: Método criarArestas() 53

Figura 23: Método acrescentarUsuario() 54

Figura 24: Exemplo de visualização no AntStudy 55

Figura 25: Métodos que utilizam o atributo Feromonio 56

Figura 26: *HashMap* que caracteriza um objeto de aprendizagem 59

Figura 27: Estruturas para exemplo do cosseno 61

Figura 28: Exemplo do cálculo do co-seno 61

Figura 29: Método inicializarFeromonioTodasArestas() 63

Figura 30: Método filtrarVerticesVisitados() 64

Figura 31: filtrarVerticesComPreRequisitosNaoVisitados() 65

Figura 32: Exemplo do cálculo da probabilidade 66

Figura 33: Exemplo das faixas de valores na escolha do vértice 66

Figura 34: Método escolherVertice() 67

Figura 35: Exemplo de *backtracking* 68

Figura 36: Disposição inicial dos materiais utilizados na demonstração 71

Figura 37: Matriz de adjacências e *JFrame* do teste 72

Figura 38: Sequências de estudo obtidas como resultado do teste 73

Figura 39: Gráfico das variações de feromônio 74

LISTA DE TABELAS

Tabela 1: Atributos da classe Grafo 49

Tabela 2: Atributos da classe ObjetoAprendizagem.....57

Tabela 3: Atributos da classe Usuario.....58

Tabela 4: Atributos da classe AntSystem.....62

Tabela 5: Exemplo do cálculo de probabilidade.....66

Tabela 6: Características dos estudantes utilizados na demonstração.....70

Tabela 7: Características dos objetos de aprendizagem.....70

Tabela 8: Notas utilizadas na demonstração.....73

Tabela 9: Arestas da demonstração.....74

LISTA DE EQUAÇÕES

Equação 1: Probabilidade de uma formiga visitar um nó 41

Equação 2: Evaporação do feromônio 42

Equação 3: Incremento das taxas de feromônio 42

Equação 4: Cálculo do co-seno (SILVA, Edeilson et. al., 2011, p. 93, adaptada).....60

Equação 5: Cálculo da probabilidade de acesso a um objeto de aprendizagem 65

Equação 6: Incremento de feromônio no AntStudy.....69

1. INTRODUÇÃO

Com o crescente avanço na utilização das Tecnologias de Informação e Comunicação, é possível ter acesso a uma extensa gama de informações em curtos intervalos de tempo. Uma das principais ferramentas para se veicular essa informação é a internet. É possível notar que essas tecnologias tornam-se cada vez mais presentes em diversas áreas, como por exemplo, a educação. Nessa área, em particular, tal crescimento é evidenciado e discutido em vários trabalhos científicos, como o de Dowbor (2001, *online*):

> A mudança é hoje uma questão de sobrevivência, e a contestação não virá de “autoridades”, e sim do crescente e insustentável “saco cheio” dos alunos, que diariamente comparam os excelentes filmes e reportagens científicos que surgem na televisão e nos jornais, com as mofadas apostilas e repetitivas lições da escola.

Sabe-se que os principais problemas encontrados nesse contexto advêm justamente da facilidade de acesso à informação. Entre eles, é possível citar a dificuldade para avaliar a veracidade dos conteúdos encontrados e ainda julgar a relevância da informação adquirida para o contexto em que será aplicada.

No momento de fazer um trabalho escolar, considerando um exemplo atual, os acadêmicos utilizam inicialmente um site de busca e, a partir daí, possuem acesso a um determinado conjunto de *websites*, cujos conteúdos apresentam alguma relação com o termo indicado no momento da pesquisa. Entre esses *websites*, podem existir conteúdos que tenham nível básico, sendo destinados a uma visão inicial acerca do assunto ou ainda nível avançado, contendo detalhes rebuscados e de difícil entendimento. Além de ser difícil julgar quais informações estão corretas, ainda há outra dificuldade: como fazer para selecionar as informações, de forma que seja possível obter uma visão inicial acerca do assunto para posteriormente aprofundar nos conceitos e construir o conhecimento desejável de uma forma coerente?

Apesar de inicialmente ser identificado em cenários abrangentes, como o exemplo exposto anteriormente, percebe-se que esse problema também está presente em contextos mais restritos. Um exemplo que ilustra essa situação é a utilização de repositórios de arquivos em universidades, que disponibilizam conteúdos de diversas disciplinas para os acadêmicos. Nesse caso, tanto o problema de avaliação da veracidade como o julgamento da relevância dos conteúdos são resolvidos, já que geralmente as postagens passaram pelo crivo dos professores da instituição (muitas vezes, eles são os responsáveis pelo material compartilhado). Mesmo assim, ainda se tem o problema da sequência de estudos.

Este trabalho, portanto, pretende apresentar uma proposta para a solução deste problema de sequenciamento de estudos no caso de um ambiente virtual de aprendizado. Para tal, outras contribuições irão surgir no decorrer do processo, tanto para uma descrição mais detalhada dos recursos de aprendizado disponibilizados aos estudantes como também formas de representar o perfil acadêmico desses usuários. O sequenciamento será feito a partir de uma técnica da área de Inteligência Artificial denominada *Ant System*, baseada no comportamento das formigas.

O trabalho está estruturado da seguinte forma: apresentação de mais detalhes acerca do projeto, como objetivos, justificativa, hipóteses e problema; revisão de literatura, contendo os principais conceitos envolvidos no trabalho; materiais e métodos, contemplando tanto a parte de construção do projeto como o planejamento de sua execução e por fim, as referências bibliográficas que foram pesquisadas durante os estudos.

1.1. Objetivos

Nessa seção, são apresentados o objetivo geral e os objetivos específicos deste trabalho.

1.1.1. Objetivo Geral

O objetivo geral deste trabalho é propiciar, através da técnica *Ant System*, a definição de sequências de estudo a partir de um conjunto de objetos de aprendizagem descritos por metadados, levando-se em consideração as características do usuário que está utilizando tais recursos.

1.1.2. Objetivos Específicos

- Propor um conjunto de metadados para descrever objetos de aprendizagem, a partir de uma adaptação no LOM proposto pela IEEE;
- Propor uma representação de usuário para ser utilizado em um contexto educacional;
- Desenvolver um algoritmo baseado no *Ant System*, que irá trabalhar tanto com os metadados que caracterizam os objetos de aprendizagem como com dados do usuário para a definição de sequências de estudo;
- Criar um cenário de teste para observar o funcionamento do algoritmo desenvolvido.

1.2. Justificativa

Sabe-se que a utilização de técnicas de computação já proporcionou diversas melhorias à área de educação, como a disponibilização de novas formas de acesso a materiais para estudo e incremento da interatividade no processo de ensino-aprendizagem. Apesar dessas vantagens, ainda é possível encontrar alguns problemas nesse contexto. Sendo assim, uma das principais motivações para a realização do trabalho é a possibilidade de resolver o problema do sequenciamento de estudos em ambientes virtuais de aprendizado, proporcionando possíveis melhorias na qualidade da educação.

É importante ressaltar ainda que o *Ant System* e outras técnicas inspiradas na natureza são pouco explorados por parte de pesquisadores brasileiros. Portanto, podem surgir várias oportunidades de contribuições acadêmicas e publicações científicas. Esses estudos podem fomentar a exploração de novas aplicações dos algoritmos inspirados na natureza a outros problemas existentes.

Além disso, esse trabalho possibilita o desenvolvimento de pesquisas em várias áreas da computação, como a inteligência artificial, a modelagem de usuários e a construção de metadados para a posterior aplicação dos conceitos aprendidos em uma solução. Sendo assim, considera-se o projeto como uma oportunidade para o aperfeiçoamento teórico e técnico acerca de parte dos conhecimentos adquiridos durante o curso de Sistemas de Informação.

1.3. Problema

É possível estabelecer sequências de estudo a partir de um algoritmo baseado na técnica *Ant System*?

1.4. Hipóteses

Este trabalho foi construído a partir das seguintes hipóteses:

- é possível representar um usuário em um ambiente virtual de aprendizado. Essa representação deve contemplar o conhecimento do usuário acerca dos elementos do contexto;
- há a possibilidade de relacionar as características presentes em um objeto de aprendizagem com as informações provenientes do usuário. Tal relação deve permitir que seja averiguado se o recurso é adequado ao usuário em um determinado momento;
- pode-se aplicar a metodologia proposta no *Ant System* a problemas de outras naturezas que não pertencem à área de otimização e buscas pelo menor caminho.

2. REFERENCIAL TEÓRICO

Esta seção apresenta alguns conceitos advindos da introdução da informática na educação e posteriormente alguns fundamentos de técnicas da computação, como o LOM, representação de usuários e o Ant System.

2.1. Informática na educação

A partir da utilização de computadores na educação, é possível criar diversas aplicações voltadas para o aprendizado, como simulações, jogos educativos e sistemas de instrução. Tais *softwares* podem ser aplicados tanto no contexto escolar, como também em treinamentos empresariais, cursos de idiomas, entre outros. Porém, deve-se ressaltar que essas tecnologias devem ser desenvolvidas a partir de uma série de conceitos teóricos inertes na área da educação, como as metodologias de ensino e aprendizado, influência do grau de interatividade entre o aprendiz e o conteúdo e presença da figura de um professor ou instrutor para viabilizar o ensino.

Essa fundamentação é importante para que os sistemas possam atender às necessidades do usuário no que tange ao processo de ensino-aprendizagem de uma maneira mais eficaz. Segundo Barbosa (2008, p. 46), existem três teorias de aprendizado que fornecem subsídios para a construção de sistemas para esta área:

- abordagem behavorista, que comporta as modalidades de ensino iniciadas na década de 60, cuja denominação mais conhecida é CAI - *Computer Assisted Instruction* (Instrução Assistida por Computador). Nessa abordagem, o ensino não contempla as características individuais de um estudante; a proposta é de apresentar problemas a um conjunto de pessoas e registrar o desempenho conquistado;
- abordagem cognitivista, agregando funcionalidades “inteligentes” ao cenário anterior, em que já é possível armazenar características de um usuário e estabelecer formas de estudo individualizadas;
- abordagem construtivista, na qual é adicionada à abordagem anterior a interatividade e a gestão de conhecimento durante os estudos para que, de

fato, ocorra a construção do aprendizado, e não simplesmente a absorção de um conjunto de conceitos.

A partir da abordagem cognitivista, parte dos estudos realizados objetiva melhorar a adaptação da interface de um sistema de informação de acordo ao perfil de um usuário e ainda proporcionar sequências de estudo que sejam compatíveis com este perfil. Brusilovsky (1992, p. 501) afirma que esse processo de sequenciamento de estudos pode ser feito de várias formas. Alguns sistemas utilizam um formato mais simples, baseado somente na apresentação de um conteúdo (através de apresentações, por exemplo) seguida de um conjunto de atividades, que são aplicadas ao usuário de acordo ao nível de complexidade. Por fim, o usuário é avaliado e esses resultados são armazenados no módulo de usuário, indicando se ele tem ou não condições de avançar para outra área de estudos.

Outras estratégias de sequenciamento são mais complexas: inicialmente analisam o conteúdo de cada material ou atividade para selecionar quais estão de acordo ao modelo de usuário e a partir dessa seleção inicial, os conteúdos são apresentados aos usuários de acordo ao seu nível de aprendizado. Se for detectado que um usuário já domina um determinado assunto e conseguiu incrementar o nível de algumas habilidades (como capacidade de realizar cálculos matemáticos envolvendo equações de segundo grau), o sistema irá apresentar conteúdos que o possibilitem desenvolver outras habilidades e aprender novos conceitos.

Independente da estratégia adotada no momento da definição das sequências, foi possível perceber a existência de dois elementos: conjunto de informações que caracterizam um recurso de aprendizagem e o usuário. No primeiro elemento, percebe-se a necessidade de representar atributos como assuntos contemplados, complexidade, relacionamentos com outros materiais, entre outros. Em relação ao usuário, nota-se que é preciso criar um perfil para agregar vários tipos de informação, como as habilidades intelectuais e comportamentais, deficiências de aprendizado, desempenho em atividades respondidas no sistema, entre outros.

A principal dificuldade existente para a representação das informações de um usuário é a limitação imposta pela interface existente entre ser humano e a máquina. Um professor, ao caracterizar um estudante, certamente iria contemplar tanto recursos provenientes de seus trabalhos e provas como também de ações

refletidas através da sua linguagem, gestos e expressões. Porém, um computador ainda não é capaz de apresentar uma eficácia considerável no que tange à interpretação de tais comportamentos.

Para caracterizar os materiais didáticos, foi utilizado o conceito de objeto de aprendizagem, tema que será apresentado na próxima seção deste trabalho.

2.2. Objetos de Aprendizagem

O reconhecimento da importância da informação fez com que o ser humano buscasse cada vez mais novas tecnologias, tanto para armazenamento como para a veiculação de tais informações. O cenário atual reflete os resultados dessa busca: é possível conhecer fatos que ocorrem em locais geograficamente distantes em questão de alguns segundos. Uma das ferramentas que protagoniza esse cenário é a internet, que revolucionou várias áreas, como as comunicações, os negócios e também a educação. Nesta última área em especial, recursos denominados **Objetos de Aprendizagem**, do inglês *Learning Objects*, surgem como uma proposta para organizar informações relacionadas a materiais educacionais com o intuito de melhorar o ensino (WILEY, 2000, p. 1).

Audino e Nascimento (2010, p. 130) afirmam que, como os objetos de aprendizagem constituem uma proposta muito nova de introdução de melhorias no sistema educacional brasileiro, os estudiosos ainda não entraram um em consenso universal a respeito de sua definição. Uma das definições bastante aceita nesse contexto foi proposta pelo IEEE - *Institute of Electrical and Electronics Engineers* - que definiu objetos de aprendizagem como qualquer entidade digital ou não-digital que pode ser utilizada para educação, ensino ou treinamento (IEEE, 2002, p. 5). Esse trabalho contemplará os objetos de aprendizagem em formato digital.

“Um objeto virtual de aprendizagem é um recurso digital reutilizável que auxilie na aprendizagem de algum conceito e, ao mesmo tempo, estimule o desenvolvimento de capacidades pessoais, como, por exemplo, imaginação e criatividade” (SPINELLI, 2007, p. 7). Para isso, os materiais podem apresentar desde conteúdos puramente teóricos até mesmo atividades para validação do aprendizado.

Como exemplos desses objetos, é possível encontrar atividades contendo perguntas objetivas ou subjetivas, textos informativos, vídeos com propósito

acadêmico, entre outros. Além disso, um objeto de aprendizagem pode conter outros objetos em seu interior, como por exemplo o conjunto de materiais instrutivos utilizados num curso profissionalizante.

A evolução das TICs - Tecnologias da Informação e Comunicação - permite que tais objetos sejam facilmente veiculados, porém, ainda encontram-se várias dificuldades no que tange a utilizar esse recurso da melhor maneira possível. Isso pode ser notado principalmente em duas áreas (VILAR, 2006, p. 40-41):

- determinação da relevância do conteúdo do recurso;
- estabelecimento de relações entre vários objetos de aprendizado.

Um exemplo disso pode ser visualizado na seguinte situação: um usuário acessa o repositório de materiais didáticos de sua universidade e busca por “estruturas de dados”. O portal retorna como resposta a este acadêmico vários livros em formato digital, apresentações de slides e links, os quais possuem os termos da pesquisa em seu título. Nesse cenário, o usuário possui várias escolhas para iniciar seus estudos e terá que determinar, quais recursos irá utilizar para construir o conhecimento almejado. Porém, o acadêmico terá que visualizar o conteúdo de cada material para averiguar se ele está adequado às suas necessidades e conhecimentos prévios. Os materiais subsequentes serão escolhidos por este acadêmico utilizando-se essa metodologia. Em alguns casos ainda, a escolha realizada pelo usuário será simplesmente randômica e irá priorizar os primeiros resultados retornados.

Para sanar tais problemas e tentar propiciar um aprendizado de qualidade, muitas vezes é preciso recorrer a recursos auxiliares, como atribuição de palavras-chave e textos descritivos. Nesses dois casos, o autor deve fazer inicialmente uma análise de quais serão os possíveis usuários de seu conteúdo para que as informações sejam coerentes.

Pode-se notar, portanto, que os dados acerca de um objeto de aprendizagem apresentam suma importância na solução desses problemas. Nesse contexto, os dados sobre um objeto de aprendizagem podem ser chamados de metadados (dados sobre dados), visto que os recursos são constituídos por dados e informações. Já existem várias iniciativas para a padronização de uma estrutura de metadados que descreva os objetos de aprendizagem de uma maneira eficaz. Uma delas é da IEEE que propôs um padrão de metadados denominado *Learning Object*

Metadata, reconhecido pela sigla LOM. Esse padrão será descrito em maior nível de detalhes na próxima seção.

2.2.1. Learning Object Metadata

LOM - *Learning Object Metadata* - consiste em uma estrutura de metadados referentes a um objeto de aprendizagem, que tem a finalidade de descrever características importantes sobre ele (IEEE, 2002, p. 5). Segundo o órgão responsável pela especificação, o LOM tem o objetivo de facilitar a busca e aquisição dos objetos de aprendizagem, tanto por parte de instrutores de ensino, como também por sistemas programados para tal fim. Esse modelo não informa como os metadados devem ser consumidos por sistemas automatizados que trabalham com os objetos de aprendizado.

Ao todo, os metadados são agrupados em nove categorias principais de acordo com seu significado, conforme apresentado na Figura 1.

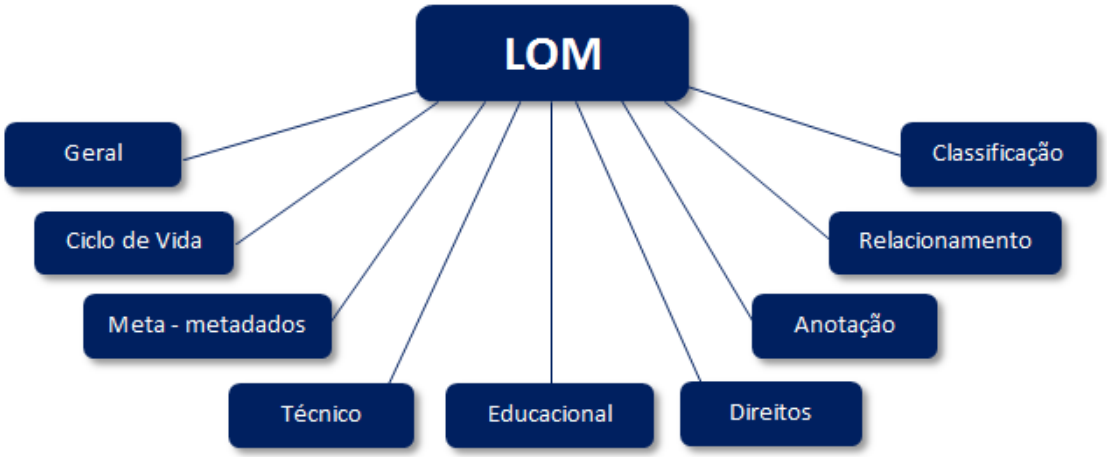


Figura 1: Categorias do LOM

Cada categoria mostrada no diagrama da Figura 1 possui um conjunto de elementos, que podem ser simples (indicarem um determinado valor) ou podem possuir novos elementos. O modelo pode ser comparado a uma estrutura de dados em árvore, na qual o elemento raiz poderia ser o LOM, os primeiros filhos seriam as nove categorias principais e os filhos posteriores seriam os outros nós da árvore. Os nós-folha são aqueles que apresentam as características inerentes ao objeto de aprendizagem.

A especificação traz detalhes importantes como nome, significado, tamanho (de dados) e exemplos de utilização para cada um dos elementos presentes na hierarquia do LOM. Deve-se ressaltar ainda que um elemento-folha desta estrutura pode apresentar uma lista de valores, nas quais a ordem pode influenciar no significado daquele metadado para o objeto que está sendo descrito.

Nas próximas seções deste trabalho serão apresentadas as nove categorias e seus respectivos elementos-filho. As informações dispostas nessas seções foram baseadas no *draft* da especificação do LOM (IEEE, 2002, p. 6-36).

2.2.1.1. Geral

Nesta categoria, encontram-se dados acerca do objeto de aprendizagem como um todo. Eles refletem algumas características como a identificação e também a composição do objeto. A Figura 2 apresenta a estrutura dos metadados presentes nessa categoria.

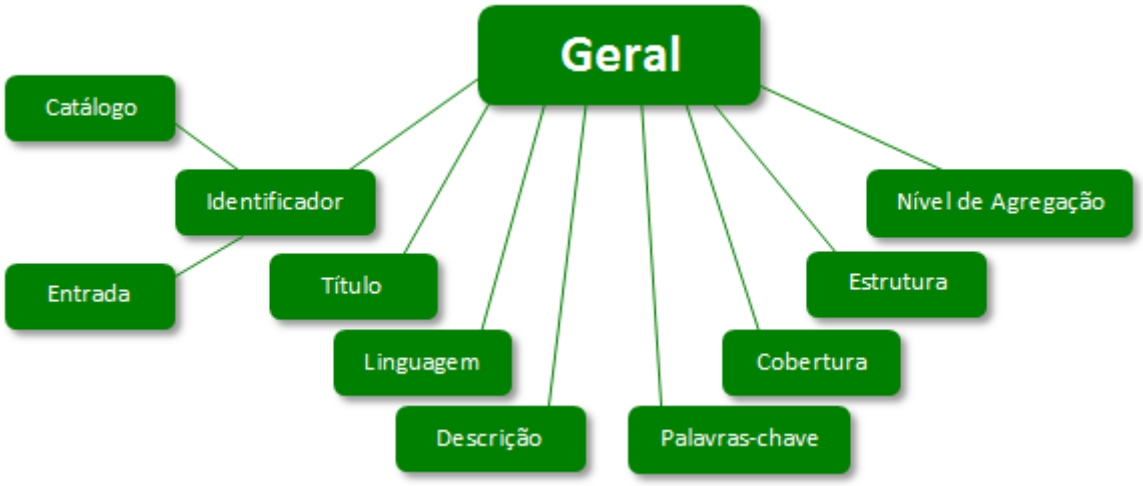


Figura 2: Metadados da categoria Geral

O primeiro elemento presente na categoria Geral é o **Identificador**. Esse metadado consiste em um elemento de identificação único para cada objeto de aprendizagem de um contexto. O identificador pode assumir diversas naturezas, como um número inteiro, uma sequência de caracteres ou até mesmo uma URI. Além disso, o identificador pode estar relacionado ao formato de identificação inerente a uma situação particular, como o armazenamento físico de livros em uma biblioteca (na qual cada obra é identificada por um valor). Sendo assim, esse

metadado apresenta dois elementos-filho, o **Catálogo**, que identifica o formato (tipo) do identificador e a **Entrada**, que é o valor da identificação.

Outro metadado dessa categoria é o **Título**, que consiste no nome do objeto de aprendizagem. O terceiro elemento da categoria é a **Linguagem** (ou idioma), que consiste no idioma ou lista de idiomas em que se encontra o objeto de aprendizagem. É recomendado que seja realizado o armazenamento do país a que se refere o idioma informado, visto que ele apresenta variações conforme a localização geográfica. Em alguns casos, esse metadado terá um valor nulo, pois o conteúdo do objeto pode ser uma imagem, por exemplo.

O quarto elemento é a **Descrição**, que armazena um texto descritivo acerca do conteúdo do objeto de aprendizagem. Deve-se ressaltar que a linguagem utilizada para o preenchimento deste atributo deve ser adequada ao contexto em que se encontra o artefato descrito. Isso significa que, em um ambiente acadêmico, o texto pode conter termos técnicos referentes à área de aplicação do material. Já em um ambiente escolar, em que os materiais são destinados a alunos das séries primárias, as frases utilizadas devem ser mais simples para que os usuários do material possam entender do que ele trata antes de visualizar o seu conteúdo.

Além dos metadados citados, a categoria Geral ainda engloba outros dados, como um conjunto de **Palavras-chave**, que armazenam os termos mais relevantes acerca do objeto. O correto preenchimento dessas palavras pode favorecer um sistema de busca, no caso dele não utilizar somente o título no momento de recuperar uma informação.

Seguindo a árvore de metadados da categoria, ainda é possível encontrar a **Cobertura**. Este atributo é utilizado para a definição de um contexto para o objeto de aprendizagem. Sendo assim, ele deve ser preenchido com informações geográficas, temporais, históricas e culturais. Como exemplos, tem-se coordenadas geográficas, século em que ocorreu o fato apresentado no OA, entre outras.

O sétimo elemento que compõe a árvore dos metadados encontrados na categoria Geral é a **Estrutura**, que define a estrutura de organização do objeto de aprendizagem. Esse metadado deve ser definido de acordo a um dos valores a seguir:

- atômica: objeto simples, sem relacionamentos;

- coleção: conjunto de objetos sem relacionamento entre si (ex.: repositório contendo apresentações de slide utilizadas em palestras semanais sobre assuntos diversificados);
- rede: conjunto de objetos com relacionamentos sem uma estrutura definida (ex.: conceitos de programação orientada a objetos que são utilizados em materiais acerca de estruturas de dados, para a criação dos nós de uma Árvore);
- hierarquia: conjunto de objetos que podem ser representados em formato de árvore (ex.: os metadados do LOM devem ser representados hierarquicamente seguindo a ideia dos nove grupos principais que contém outros metadados);
- linear: conjunto de objetos com alto grau de ordenação (ex.: conjunto de conteúdos da disciplina Banco de Dados, seguindo uma sequência desde à abstração dos problemas até o aprendizado da linguagem SQL).

O último metadado é o **Nível de Agregação**, que mostra a granularidade do objeto de aprendizagem. Tal característica é expressa através de quatro níveis, conforme as seguintes especificações:

- nível 1: menor nível de agregação (ex.: diagrama de sequência de projeto);
- nível 2: agrupa objetos de aprendizagem com nível de agregação igual a 1 (ex.: atividade sobre o diagrama de sequência de projeto);
- nível 3: agrupa objetos de aprendizagem com nível de agregação igual a 2 (ex.: conjunto de atividades sobre vários diagramas de sequência de projetos, provenientes de diferentes fontes);
- nível 4: maior nível de agregação, que agrupa objetos com nível de agregação igual a 3 ou a 4 (ex.: curso que concentra definições, características, exemplos e atividades sobre diagramas de sequência de projetos).

Com base nos dois últimos metadados apresentados, pode-se inferir que objetos de aprendizagem com nível de agregação igual a 1 certamente irão possuir estrutura atômica.

2.2.1.2. Ciclo de Vida

Os metadados da seção Ciclo de Vida descrevem a história e o estado do objeto de aprendizagem, assim como quem influenciou em seu conteúdo durante o desenvolvimento. Esses metadados refletem uma das características dos objetos de aprendizagem, que é a possibilidade de atualização do conteúdo. Além disso, ele em sua utilidade também em contextos nas quais são armazenadas versões antigas para um mesmo objeto, possibilitando sua posterior recuperação, se necessário. A árvore de metadados presentes nessa categoria é apresentada na Figura 3.

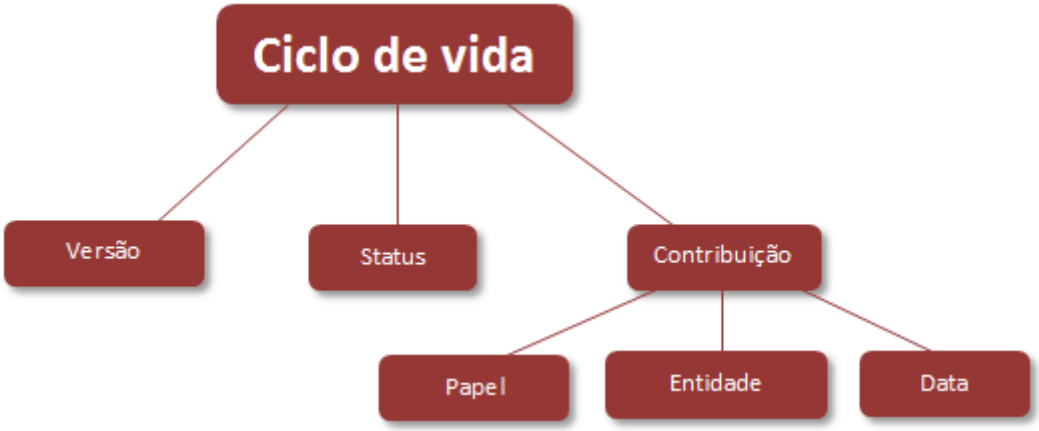


Figura 3: Metadados da categoria Ciclo de Vida

Conforme a árvore de elementos apresentadas na Figura 3, o primeiro metadado indicado é a **Versão**, número que identifica a versão (ou edição) do objeto de aprendizagem. Seguindo a árvore, é possível identificar o atributo **Status**, que por sua vez, identifica o estado do objeto. Os valores possíveis para o preenchimento deste metadado são: rascunho, final, revisado e indisponível. Supondo um ambiente virtual de aprendizado, no qual professores são responsáveis por compartilhar material didático com seus alunos, um determinado objeto de aprendizagem pode ser salvo com o status “rascunho” enquanto o professor está terminando o seu conteúdo e fazendo os devidos ajustes. Assim que a publicação é realizada, o material passa a ter o status “final” (ou finalizado). Se depois de publicado o conteúdo precisou sofrer alguma alteração, para correção de um dado, por exemplo, ele pode assumir o status “revisado”. Por fim, caso ele tenha sido retirado do

ambiente virtual por algum motivo, como para uma correção extensiva de seu conteúdo, o status pode ser alterado para “indisponível”.

O terceiro metadado do Ciclo de Vida é a **Contribuição**, que indica as pessoas ou organizações que influenciaram diretamente na construção ou publicação do objeto de aprendizagem que está sendo caracterizado. Deve-se ressaltar que o preenchimento deste metadado exige a avaliação de várias formas de contribuição, que devem ser adequadas ao contexto. Em alguns ambientes de aprendizagem, existe a figura de um revisor que participa do processo de publicação, e isso implica em uma menção a ele no momento de preencher esse metadado. Para que uma contribuição seja representada, é preciso identificar o **Papel**, o qual determina o tipo de contribuição (autor, editor, designer gráfico, validador técnico ou desconhecido, por exemplo); a **Entidade**, que indica a pessoa ou organização responsável pela contribuição; e a **Data**.

2.2.1.3. Meta-metadados

Os meta-metadados, como o próprio nome sugere, são dados referentes às características do esquema de metadados que está sendo utilizado para descrever um objeto de aprendizagem. Portanto, esses dados não descrevem o objeto de aprendizagem em si. A árvore dos meta-metadados é apresentada na Figura 4.

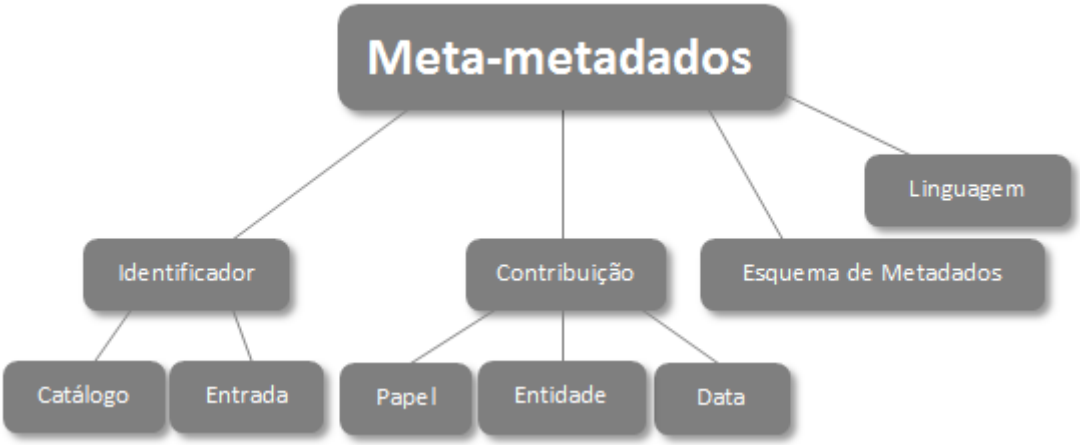


Figura 4: Metadados da categoria Meta-metadados

Assim como na categoria Geral, esse grupo também apresenta um elemento **Identificador** com os filhos **Catálogo** e **Entrada**, que servem para identificar o

esquema de metadados utilizado para um determinado objeto de aprendizagem. O significado dos atributos é o mesmo: o primeiro para armazenar o padrão de identificação e o segundo para indicar o valor, propriamente dito.

Além disso, nessa categoria também existe o metadado **Contribuição**, com os seus respectivos metadados **Papel**, **Entidade** e **Data**, cujos significados são semelhantes aos dados presentes no grupo Ciclo de Vida. Nesse contexto, ele indica a contribuição das pessoas ou entidades para a criação ou validação do esquema utilizado para representação dos metadados.

Outro atributo presente no grupo é o **Esquema de Metadados**, que possibilita a determinação de qual estrutura está sendo utilizada para a caracterização do objeto de aprendizagem. Um possível valor para preenchê-lo seria “LOMv1.0”, indicando que o esquema de metadados foi baseado na especificação 1.0 do LOM, criada pela IEEE.

O último metadado desse grupo é a **Linguagem**, que indica o idioma no qual foram especificados os metadados.

2.2.1.4. Técnico

O grupo de metadados técnicos descreve as características inerentes ao objeto de aprendizagem, assim como os requerimentos necessários de hardware e software para acessar seu conteúdo. A Figura 5 apresenta um diagrama contendo os dados desta categoria.

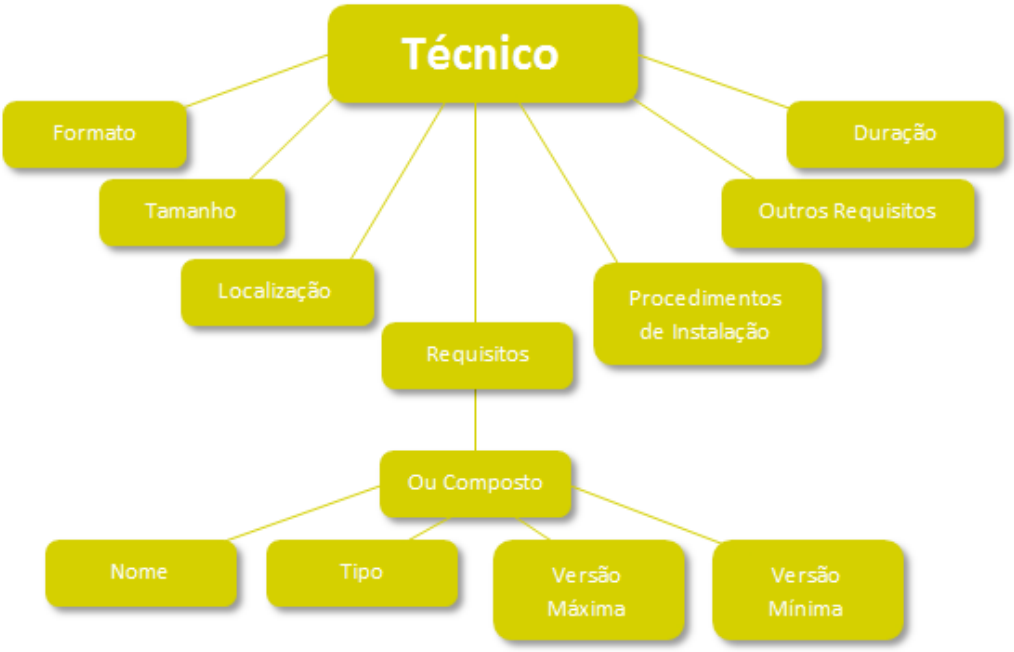


Figura 5: Metadados da categoria Técnico

De acordo com a Figura 5, o primeiro metadado desta categoria é o **Formato**. Ele deve ser preenchido com o tipo de dados (ou tipos) do qual é composto o objeto de aprendizagem. Alguns exemplos para o preenchimento desse campo seriam: text/html ou video/mpeg. Quando devidamente preenchido, este dado pode ser de grande importância para um dispositivo reconhecer se pode ou não prover acesso a um recurso sem precisar realizar um processamento maior para tal fim. A especificação da IEEE rege que o formato deve seguir os padrões definidos na RFC 2048:1996 (MIME *types*).

O segundo atributo apresentado na árvore é o **Tamanho**, que armazena a quantidade de bytes ocupada em disco pelo objeto de aprendizagem. Deve-se ressaltar que esse tamanho deve obrigatoriamente ser expresso em bytes e não em outra unidade como megabytes ou gigabytes. As devidas conversões para apresentação dos valores devem ser realizadas posteriormente. Além disso, esse valor deve refletir o objeto sem que ele esteja comprimido. O terceiro metadado do grupo é a **Localização**, que indica a URL ou URI na qual encontra-se armazenado um objeto de aprendizagem.

Nesse grupo de metadados técnicos, é possível informar os **Requisitos** necessários para que um objeto de aprendizagem seja executado da melhor forma possível. Esses requisitos devem refletir necessidades referentes à infra-estrutura, como hardware e software. A indicação desses requisitos é feita indicando-se grupos

para cada característica que se deseja contemplar. Isso significa, que pode ser criado um conjunto para definir os requisitos em relação à processamento da máquina, capacidade de armazenamento disponível, especificações de software, entre outras. Cada grupo de características é armazenado em um metadado denominado **Ou Composto**, que por sua vez, apresenta:

- **Tipo**, que determina o tipo do requisito (capacidade de processamento, sistema operacional, versão de um *framework*, entre outros);
- **Nome**, para indicar qual tecnologia está sendo requerida (considerando o tipo como sistema operacional, por exemplo, tem-se possibilidades como Windows 7, Mac OS, Linux, etc.);
- **Versão máxima** especifica a versão máxima da referida tecnologia;
- **Versão mínima** especifica a versão mínima da tecnologia indicada.

O nome “Ou Composto” advém da ideia de que somente um dos requisitos presentes no grupo precisa ser satisfeito. Tomando como exemplo a característica “processamento”, pode-se ter um Ou Composto indicando várias possibilidades, como Core i7 Extreme 980 ou Phenom II X6 1090T. Se a máquina na qual o objeto de aprendizagem estiver sendo acessado possuir um desses processadores, o requisito está satisfeito. Deve-se ressaltar ainda que existirão outros elementos Ou Composto para as demais características que deseja-se contemplar.

O próximo metadado do grupo de características técnicas chama-se **Procedimentos de Instalação**, que indica o que um usuário deve fazer para acessar e utilizar um objeto de aprendizagem. Como exemplos de indicações, pode-se citar a descompactação de uma pasta contendo várias páginas HTML e o posterior acesso ao arquivo index.html.

A especificação ainda permite que sejam indicados **Outros Requisitos**, que não puderam ser expressos nos metadados apresentados anteriormente. Por fim, há o atributo **Duração**, que indica o tempo de execução para objetos de aprendizagem como filmes, animações ou sons.

2.2.1.5. Educacional

O conjunto de metadados existentes na categoria Educacional reflete características de cunho pedagógico do objeto de aprendizagem, cujo objetivo é proporcionar um

aprendizado de qualidade para quem estiver utilizando o recurso. A Figura 6 apresenta a árvore de metadados existentes nessa categoria.

Figura 6: Metadados da categoria Educacional

O primeiro metadado apresentado é o **Tipo de Interatividade** que deve ser preenchido de acordo com o grau de interatividade proporcionada por um objeto de aprendizagem ao usuário que o acessar. Conforme a especificação, esse metadado pode apresentar um dos três seguintes valores:

- ativo: recurso que induz o usuário a executar ações ativas em relação ao objeto de aprendizagem, como fornecer entradas e tomar decisões. Exercícios para o qual é exigida uma solução e questionários de questões objetivas ou subjetivas são exemplos de objetos de aprendizado ativos;
- expositivo: objeto em que o usuário somente deve absorver informações para a construção do conhecimento. Exemplos: vídeos, fotos, apresentações de slide, etc.;
- misto: quando o objeto de aprendizagem possui características dos dois tipos citados anteriormente.

O **Nível de Interatividade** proporcionado pelo objeto de aprendizagem é indicado através de valor dentre cinco possibilidades: muito baixo, baixo, médio, alto e muito alto. Recursos com tipo de interatividade ativo certamente terão níveis de interatividade mais altos do que recursos expositivos.

O **Tipo de Recurso de Aprendizagem** indica a metodologia contemplada pelo objeto de aprendizagem. Na maior parte dos casos, um mesmo recurso

apresenta vários tipos, sendo que eles devem ser informados em ordem de predominância. Alguns exemplos são: exercício, imagem, questionário, simulação, slide, tabela, etc.

A **Densidade Semântica** armazena o nível de concentração de conceitos existentes em um determinado objeto de aprendizagem. Deve-se ressaltar que essa caracterização não reflete a dificuldade de aprendizado, visto que os conceitos podem ser de fácil compreensão por parte do usuário, e sim a quantidade de conceitos em relação ao tamanho total do conteúdo. Os valores possíveis para esse atributo são: muito baixa, baixa, média, alta e muito alta. Cabe às pessoas responsáveis pela implantação do LOM a definição de regras ou padrões que devem ser adotados no momento de realizar essa classificação.

No **Papel Destinado**, deve ser indicado o "papel" do usuário para o qual o objeto de aprendizagem é destinado. Os valores possíveis para este atributo são:

- professor: papel que caracteriza uma pessoa que apresenta os conceitos presentes em um objeto de aprendizagem. Em ambientes de aprendizagem virtuais, geralmente os professores são os responsáveis pela publicação do material e, em alguns casos, são também os autores de tais recursos;
- autor: pessoa responsável pela criação de um objeto de aprendizagem. Sendo assim, serão destinados a esse papel recursos como guias de autoria e produção de conteúdo;
- aprendiz: utiliza o objeto de aprendizagem com o intuito de aprender. Alguns exemplos de usuários que possuem esse papel são alunos do colegial, acadêmicos, profissionais em treinamento, etc.;
- gerente: entidade que gerencia o compartilhamento do objeto de aprendizagem, como uma universidade ou escola.

O sexto metadado da categoria serve para indicar o **Contexto** em que o objeto de aprendizado está sendo utilizado. A especificação fornece alguns valores possíveis, como escola, ensino superior e treinamento, mas sugere ainda que sejam adicionadas informações como o local de aplicação.

Outro metadado que é contemplado é a **Idade**, que expressa a idade (ou faixa de idade) na qual um usuário certamente terá melhor aproveitamento de um conteúdo oferecido pelo objeto de aprendizagem. Além disso, é possível indicar o nível de **Dificuldade** deste recurso, que pode assumir um dentre os seguintes valores: muito fácil, fácil, médio, difícil ou muito difícil. É preciso destacar que

determinar o grau de dificuldade de um objeto de aprendizagem é uma tarefa subjetiva, cabendo então aos responsáveis padronizar formas e métricas que auxiliarão nesse processo. Vilar (2006, p. 64) lista algumas formas de se fazer isso, como a utilização da média de conceitos novos introduzidos no objeto de aprendizagem ou ainda a análise do tempo gasto pelos usuários para visualizar (estudar) cada conceito apresentado.

Seguindo a árvore de dados da categoria, é possível encontrar o **Tempo de Aprendizado**, que apresenta uma estimativa do tempo que é gasto para o aprendizado dos conceitos presentes em um conteúdo ou realização de uma atividade proposta no objeto de aprendizagem. Tal atributo pode ser utilizado como parâmetro de controle e planejamento de aulas, cursos, palestras, etc.

Existe ainda nesta categoria o atributo **Descrição**, cujo objetivo é fornecer informações acerca de como o objeto de aprendizagem deve ser utilizado (suporte para uma palestra, exercício para ser respondido com o auxílio de outro objeto de aprendizagem, por exemplo). Por fim, há o metadado **Linguagem**, que informa a linguagem dos destinatários de um determinado recurso. Deve-se ressaltar que um material pode ter seu conteúdo escrito em “inglês”, mas ser destinado a brasileiros que estão aprendendo a língua, portanto, neste caso, esse metadado deveria ser preenchido com “pt-BR” .

2.2.1.6. Direito

Esse grupo descreve os direitos de propriedade intelectual sobre um objeto de aprendizagem. O diagrama da Figura 7 apresenta a árvore de metadados desta categoria.

Figura 7: Metadados da categoria Direito

A partir do diagrama, é possível observar os atributos **Custo** e **Copyright e Outras Restrições**, que devem ser preenchidos com os valores “sim” ou “não”, caso a característica se aplique ao objeto de aprendizagem que estiver sendo caracterizado. Além de representar tais estados, é possível armazenar comentários acerca das condições de uso deste objeto, no metadado **Descrição**. Sendo assim, é possível descrever com mais detalhes as restrições inerentes ao uso e compartilhamento do recurso.

2.2.1.7. Relacionamento

Esse grupo permite que sejam estabelecidos relacionamentos entre os objetos de aprendizagem. Caso esses relacionamentos tenham multiplicidade maior que um (um objeto se relaciona a mais de um objeto), deve-se instanciar o relacionamento mais de uma vez (sendo que se deve respeitar o numero máximo de cem relacionamentos). A árvore de metadados desta categoria é apresentada na Figura 8.

Figura 8: Metadados da categoria Relacionamento

O primeiro metadado desta categoria é o **Tipo**, no qual será armazenada a natureza do relacionamento entre os objetos de aprendizagem. Sendo assim, pode-se representar uma relação de dependência, referência, embasamento, entre outros. Isso é essencial para que um usuário adote sequências de estudos com maior nível

de aproveitamento intelectual, uma vez que poderá verificar a existência de conceitos interdependentes e ainda poderá buscar materiais relacionados para incrementar seus conhecimentos.

Além de estabelecer o tipo, é preciso indicar qual objeto de aprendizado participa do relacionamento. Para essa indicação, utiliza-se o metadado **Recurso**, que possui um **Identificador** (estruturado da mesma forma que os identificadores já apresentados neste documento, com um **Catálogo** e uma **Entrada**) e uma **Descrição**. A descrição deve contemplar somente o objeto-alvo e não descrever ou caracterizar o relacionamento entre os objetos.

2.2.1.8. Anotações

As **Anotações** armazenam comentários acerca de um objeto de aprendizagem. O metadado armazena o autor do comentário no atributo **Entidade**, a **Data** e o comentário, propriamente dito, no atributo **Descrição**. Dentre os comentários, pode-se citar: sugestões para melhorias no conteúdo e correções de alguma informação incorreta, detectada por um usuário.

A árvore de elementos deste conjunto é apresentada na Figura 9.

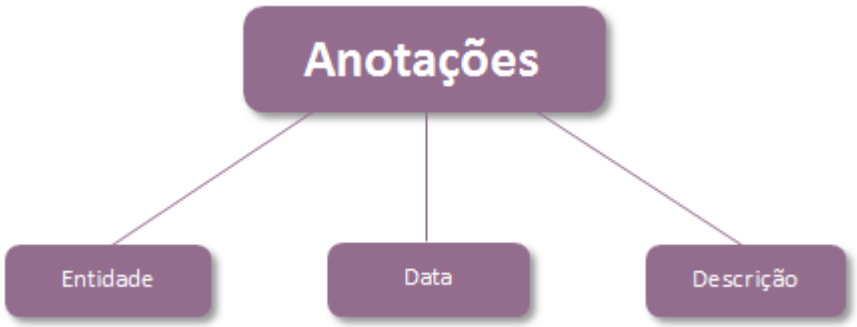


Figura 9: Metadados da categoria Anotações

2.2.1.9. Classificação

Apesar de alguns dos metadados expostos anteriormente proverem um formato de classificação dos objetos de aprendizagem (quanto à complexidade, nível de interação, quantidade de relacionamentos, entre outros), é possível propor novos

formatos de classificação, que atenda a particularidades de um determinado contexto. Como exemplo, pode-se criar um novo formato de classificação para os objetos de aprendizagem utilizados como material de instrução de uma disciplina ministrada na universidade, de acordo ao seu plano de ensino. Outra forma de classificação seria quando às competências e habilidades necessárias para o melhor aproveitamento do recurso educacional. A árvore de metadados presente nesse grupo é apresentada na Figura 10.

Figura 10: Metadados da categoria Classificação

O primeiro metadado dentro da categoria é a **Finalidade**, que expressa qual o objetivo principal do sistema de classificação que está sendo utilizado. Dentre os valores possíveis para preenchimento, tem-se disciplina, ideia, acessibilidade, restrições, etc.

Nesse contexto, é preciso apresentar a **Taxonomia**, no qual são armazenados os elementos da classificação. Para assegurar maior completude de informações, esse metadado apresenta um sub-metadado denominado **Fonte**, que indica o nome desse sistema de classificação e o **Táxon**, que representa um termo presente na taxonomia. Um táxon, por sua vez, apresenta um **Id**, que é o identificador do termo e a **Entrada**, que contém o valor.

Além disso, a estrutura do LOM propõe que sejam armazenadas uma **Descrição** e **Palavras-chave** acerca do objeto de aprendizagem inserido no sistema de classificação.

Conforme o que foi exposto nesta seção, pode-se perceber que a especificação do LOM define uma quantidade extensa de metadados para descrever um objeto de aprendizagem. Dependendo do contexto, alguns desses atributos

acabam sendo desnecessários, portanto é preciso adaptar o LOM proposto pela IEEE para que ele atenda somente às necessidades identificadas no domínio. Para essa adaptação, utiliza-se o nome de *Application Profile*.

O *Application Profile* consiste em um conjunto de metadados retirados de uma ou mais especificações (como o LOM) combinados por desenvolvedores e adaptado para uma determinada aplicação (HEERY e PATEL, 2000, *on-line*). Além de realizar as devidas adaptações nos padrões no LOM para criar um *application profile* coerente com o domínio, cabe ao desenvolvedor verificar quais os metadados que podem ser preenchidos automaticamente pela aplicação, diminuindo o esforço por parte dos usuários que irão gerenciar os objetos de aprendizagem no sistema.

A próxima seção deste trabalho apresenta os conceitos de representação de usuários.

2.3. Representação de usuários

A evolução das tecnologias voltadas para a construção e o compartilhamento de conteúdos cresceu muito nos últimos anos. Atualmente, percebe-se que elas se encontram na vida das pessoas em diversos momentos. As formas de se realizar uma pesquisa escolar, estudar para uma prova, fazer um treinamento profissional, tramitar processos dentro de uma organização, conhecer pessoas novas e até mesmo manter relacionamentos interpessoais, são alguns exemplos em que a tecnologia se apresenta como um recurso quase que indispensável.

Um desenvolvedor de software, além de ter que produzir sistemas que possam atender às necessidades de cada domínio, também deve considerar a diversidade de usuários que irá usufruir de uma determinada aplicação. Essa última característica em especial, é um dos pontos pesquisados em um campo de estudos denominado Modelagem de Usuários.

Segundo Brusilovsky e Schwarz (1997, p. 1), os usuários que utilizam as aplicações desenvolvidas principalmente para a web geralmente apresentam várias diferenças em relação ao nível de conhecimento e experiência no uso da internet. Deve-se destacar ainda a existência de usuários portadores de deficiência, para os quais as aplicações deveriam oferecer recursos de acessibilidade. Aliado ao fato exposto, é preciso considerar que o cenário tecnológico é marcado pelo avanço acelerado no que tange à criação de novos recursos, como funcionalidades mais

complexas e interfaces cada vez mais ricas em botões, animações, janelas, formulários, entre outros. Portanto, a modelagem de usuários apresenta-se como uma área de estudos de grande importância, para que os sistemas de adaptem às particularidades de cada usuário.

As características de um usuário são armazenadas em estruturas denominadas modelos de usuário. De acordo com Brusilovsky (1996, p. 300), tais modelos funcionam como abstrações do mundo real que representam várias características de um usuário, como seu histórico, habilidades, deficiências e preferências.

A modelagem de usuários pode assumir diversos formatos, conforme as necessidades de cada contexto em que os modelos são utilizados. Existem casos em que é preciso conhecer poucas informações sobre os usuários, como a idade e o sexo. Já em outros casos, é preciso conhecer algumas particularidades em relação ao comportamento ou ainda a preferências. Essas informações podem ser obtidas tanto de forma direta, em que o usuário é responsável pelo fornecimento dos dados a partir do preenchimento de formulários, por exemplo, como de forma indireta, a partir da análise de um conjunto de características realizada por um sistema.

O modelo de usuário pode ser utilizado para identificar um único indivíduo ou também para identificar grupos de usuários com características semelhantes. Nesse caso, os modelos de usuário recebem o nome de estereótipos. Conforme Rich (1979, p. 331), estereótipos são coleções na forma de chave = valor que descrevem grupos de usuários dentro de um sistema. Dependendo do contexto, os valores podem assumir o formato booleano (*true* ou *false*) ou também valores discretos.

Sendo assim, as adaptações do sistema serão realizadas em nível de grupo e não mais no âmbito individual. Pode-se definir, por exemplo, grupos de usuário novatos, intermediários ou experientes. A partir dos dados de um usuário, ele seria encaixado em um grupo e herdaria todas as funcionalidades e recursos de interface designadas para aquele grupo. É importante frisar que, com o passar do tempo, esse usuário poderia mudar de grupo, caso fossem detectadas características que permitissem tal fato. A Figura 11 apresenta um exemplo de construção possível de estereótipos para a modelagem de usuários.

Figura 11: Exemplos de estereótipos no curso de Sistemas de Informação

O modelo de estereótipos da Figura 11 ilustra quatro classes distintas de acadêmicos do curso de Sistemas de Informação. Conforme eles aprendem novos conceitos e adquirem novas habilidades no decorrer do curso, eles adotam um novo estereótipo. Um usuário que acabou de entrar no curso será classificado inicialmente como **Aprendiz**. Ao completar o terceiro período, poderá ser enquadrado como **Aluno Intermediário**, de acordo ao aproveitamento adquirido em sua vida acadêmica.

Um modelo de usuário exige não só o processo de aquisição de dados para sua criação, como também de atualização e manutenção enquanto o sistema está ativo, visto que os usuários demonstram diferentes características ao longo das interações com a aplicação. Sendo assim, o sistema deve monitorar vários dados,

como cliques do mouse, entradas de voz, toques em interfaces, tempo de visualização de conteúdos, frequências de acesso, entre outros (KOCH, 2001, p. 53). Em relação à captura dessas características ou análise de comportamento, existem várias técnicas computacionais que podem ser utilizadas, como redes bayesianas ou redes neurais.

A escolha das características que são utilizadas como chaves na construção dos estereótipos depende intrinsecamente do contexto. Conforme o escopo deste trabalho, objetiva-se criar uma representação do usuário que fornecerá dados para o algoritmo de sequenciamento de estudos. Sendo assim, não será abordada uma estrutura para a aquisição dos dados ou a atualização do modelo, visto que ele será necessário somente para o fornecimento dos dados que serão utilizados pelo algoritmo.

Como este trabalho é caracterizado pelo contexto educativo, foi necessário um estudo mais aprofundado acerca das particularidades dos modelos de usuário presentes neste cenário, que são apresentadas na próxima seção.

2.3.1. Modelo de usuário em um contexto educacional

“O modelo de um usuário estudante refere-se à representação dinâmica dos conhecimentos emergentes e habilidades de um aluno” (NWANA, 1990, 260). Este modelo deve abrigar o máximo de elementos que influenciem o desempenho e aprendizagem do usuário no contexto educacional. Tal representação é um dos maiores problemas encontrados nesse processo de modelagem, dada a limitada interface de comunicação entre o estudante e o computador.

Vanlehn (1988, p. 55-57) afirma que para representar um usuário nesse contexto é preciso trabalhar com dois módulos interligados:

- modelo do estudante: que consiste na estrutura de dados que armazena o estado atual de conhecimento do aluno acerca dos elementos existentes dentro do contexto de ensino-aprendizagem;
- módulo de diagnóstico: mecanismo responsável por utilizar os dados advindos do modelo do estudante para realizar as devidas adaptações no sistema educacional e ainda por coletar novos dados para averiguar e se preciso, atualizar o estado de conhecimento.

Como esses dois módulos estão interligados, devem ser planejados e implementados de forma conjunta. Esse conjunto composto por modelo de estudante e módulo de diagnóstico é amplamente utilizado para diversos fins, como a adaptação de explicações e conceitos às necessidades do aluno, oferecimento de explicações e ainda para a definição de sequências de estudo. Neste trabalho, não será desenvolvido um módulo de diagnóstico.

As próximas seções apresentam mais detalhes acerca desses dois módulos que compõem a abstração de um estudante em um sistema educacional.

2.3.1.1. Modelo do estudante

O modelo de estudante armazena os dados que são utilizados posteriormente pelo módulo de diagnóstico para a adaptação do sistema às necessidades do aluno. De fato, existe uma extensa gama de possibilidades que podem ser representadas neste modelo, sendo que os desenvolvedores do sistema e analistas de domínio devem entrar em consenso acerca do que é necessário representar para uma determinada aplicação. Apesar dessa dificuldade de apontar os elementos que devem ser contemplados no modelo de estudante, alguns estudiosos propuseram estruturas generalizadas, que podem ser adequadas de acordo às particularidades de cada domínio. Um dos padrões amplamente aceitos nessa área de pesquisas é o modelo de três dimensões proposto por Vanlehn (1988, p. 57-64).

A primeira dimensão do modelo de estudante é denominada **Volume de Dados** (*Bandwidth*), que consiste na quantidade e granularidade dos dados que serão armazenados. Nesse contexto, entende-se por granularidade o nível de detalhes acerca de uma entrada oferecida pelo usuário ao sistema. A granularidade propõe a classificação das entradas em três grupos distintos:

- estados mentais (*mental states*): esse grupo engloba informações que representam cada etapa do raciocínio utilizado no momento da aprendizagem. Os estados mentais possuem o maior nível de detalhamento dentre os grupos de granularidade existentes, portanto, é possível realizar um conjunto maior de inferências acerca do processo de ensino-aprendizagem. Na resolução de uma equação algébrica, por exemplo, as movimentações de números para o outro lado da igualdade e a realização de operações básicas entre os termos refletem estados mentais de um estudante;

- estados intermediários (*intermediate states*): nesse grupo, são representadas informações com um grau de detalhamento menor em relação ao grupo anterior. Retomando o exemplo da equação algébrica, cada expressão apresentada em uma linha pode ser considerada como um estado intermediário, resultante de um pequeno conjunto de operações sobre os elementos presentes na expressão;
- estados finais (*final states*): esse grupo apresenta o menor nível de detalhamento para as informações. Na resolução da equação algébrica, esse estado consiste somente no resultado final encontrado pelo estudante, desconsiderando qualquer raciocínio realizado para se atingir a resposta.

A segunda dimensão do modelo de estudante é o **Tipo de Conhecimento**, que pode ser classificado como:

- declarativo: indica “o que” é apresentado, ou seja, os fatos e informações pontuais, que podem ser representados através de conceitos;
- procedural: indica “como” as informações são manipuladas, possibilitando a abstração de quais metodologias e formas de raciocínio são utilizadas no momento de um aprendizado.

Dependendo do domínio, pode-se representar tanto o conhecimento declarativo como o procedural, possibilitando ao sistema interpretar o conjunto de conceitos que um estudante domina e também as estratégias utilizadas por ele no processo de ensino-aprendizagem.

A terceira dimensão do modelo de estudante contempla as **Diferenças Entre o Estudante e *Expert***. De acordo com essa abordagem, pode-se considerar que um estudante domina uma parte do conhecimento designado a um *expert*. Nesse contexto, pode-se utilizar a técnica *overlay* (indicando os conceitos que o estudante domina, a partir de um conjunto de conceitos elencados no âmbito de sistema, que seria a representação do *expert*). Além disso, pode-se ainda montar uma “lista de bugs” (*bug library*) para armazenar os principais erros de aprendizado do estudante durante suas interações com o sistema.

2.3.1.2. Módulo de diagnóstico

O modelo de estudante é constantemente utilizado em um sistema de cunho educacional, visto que, a partir dos dados armazenados nessa estrutura, serão

realizadas as devidas adaptações do sistema às necessidades do usuário. Além disso, é preciso que ele esteja em constante atualização, visto que o nível de conhecimento de um usuário acerca dos assuntos explorados no contexto pode mudar ao longo do tempo. Portanto, o módulo de diagnóstico representa a ponte existente entre o conhecimento real do usuário e o conhecimento que está armazenado em um sistema educacional.

Deve-se reconhecer que a interpretação dos dados de entrada fornecidos por um estudante ao sistema não é uma tarefa trivial, visto que exige um elevado nível de abstração e sistematização por parte do desenvolvedor. Ragnemalm (1996, p. 6-9) afirma que os procedimentos realizados pelo módulo de diagnóstico podem ser divididos em três fases sequenciais, apresentadas na Figura 12.

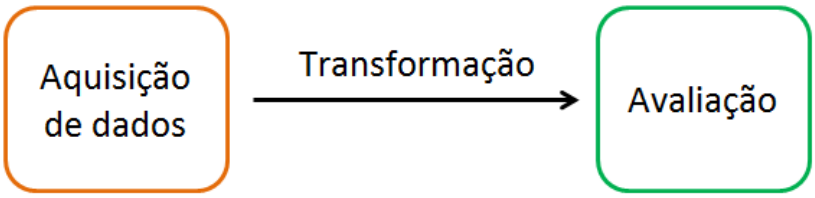


Figura 12: Procedimentos do módulo de diagnóstico

O primeiro procedimento apresentado na Figura 12 é a **Aquisição de Dados**. Muitas vezes, é necessário adequar a própria interface do sistema de acordo ao tipo de entrada desejada para representar o conhecimento. Tomando com exemplo um sistema educacional em que os usuários podem responder atividades, é possível apresentar várias formas de adequação:

- se o contexto exigir um nível de granularidade alto acerca do conhecimento do usuário, pode-se utilizar metodologias de resolução de problemas passo-a-passo, permitindo que sejam armazenadas as informações de cada etapa do processo (possibilitando a inferência de estados mentais);
- caso seja necessária uma representação menos detalhada, pode-se utilizar somente um formulário para que o acadêmico ofereça uma resposta final para o problema que foi apresentado (o que implica na representação de estados finais).

O segundo processo é a **Transformação**, que consiste na adequação da entrada do usuário em relação ao conhecimento representado no sistema para a

extração de informações acerca do aprendizado do aluno. Dependendo do nível de automatização presente no sistema, a transformação pode assumir vários níveis de complexidade. Considerando o exemplo do sistema de atividades, é preciso transformar a resposta oferecida pelo aluno de acordo com o formato que o sistema possa interpretar e corrigir. Essa transformação exige a utilização de técnicas como processamento de linguagem natural e aprendizado de máquina, o que caracterizam o processo como tendo alta complexidade. Já em um sistema de caráter mais expositivo, como um repositório de arquivos cujos estudantes podem visualizar materiais didáticos, deve-se converter o tempo que o aluno gastou para avançar por cada tópico de acordo ao formato que está sendo utilizado no sistema. Essa transformação exige somente um pequeno conjunto de operações matemáticas para a realização da conversão, o que caracteriza o processo de transformação como tendo baixa complexidade.

Depois de adquirida e devidamente transformada, a entrada do estudante passa pelo processo de **Avaliação**, que consiste na interpretação do valor para a devida construção ou adequação do modelo de estudante. Essa interpretação pode ser realizada a partir de um conjunto de regras que reflitam as particularidades do domínio da aplicação. Utilizando novamente o exemplo do sistema de atividades, a avaliação irá, de fato, verificar o quanto a resposta do aluno está correta e se aquilo representa ou não uma modificação imediata no modelo de estudante. Além disso, é preciso considerar se um aluno pode ou não fornecer respostas que se diferenciem em quaisquer aspectos do conhecimento representado no sistema.

A realização do diagnóstico para a construção e atualização do modelo de usuário pode ser realizada de várias formas diferentes, de acordo às características do domínio. No caso de um sistema cuja finalidade é lidar principalmente com um repositório de arquivos, pode-se criar um conjunto de regras para avaliar as entradas e definir as alterações do modelo de usuário. Se o sistema for mais complexo, com funcionalidades de correção de atividades, por exemplo, pode-se utilizar técnicas de inteligência artificial, como árvores de decisão e sistemas especialistas para averiguar o grau de coerência da resposta com a solução dos exercícios, para assim determinar o que deve ser registrado no modelo.

A próxima seção apresenta informações sobre a técnica do Ant System, que será utilizada para a definição das sequências de estudo.

2.4. Ant system

Atualmente, sabe-se que existem diversas abordagens para a resolução de problemas complexos na área de Inteligência Artificial. Entre elas, encontra-se a *Swarm Intelligente* (inteligência de enxames), cujos princípios são baseados no comportamento dos insetos e outros animais que vivem em sociedade (DORIGO, BIRATTARI, STÜTZLE, 2006, p. 28). Dentre esses animais, é possível encontrar abelhas, pássaros, peixes, formigas etc.

O algoritmo que será desenvolvido neste trabalho será baseado no *Ant System*, que é uma técnica presente em um ramo de estudos da *Swarm Intelligence* denominado *Ant Colony Optimization* (otimização por colônias de formigas), cujos estudos são inspirados no comportamento das formigas. A próxima seção apresenta alguns conceitos biológicos sobre esses animais e posteriormente, os fundamentos computacionais presentes no *Ant System*.

2.4.1. Inspiração biológica

As formigas surgiram por volta da segunda metade da era Paleozóica (aproximadamente 540 milhões de anos atrás) e atualmente são objetos de estudo em várias áreas da biologia, principalmente por serem insetos sociais. Essa designação advém do fato de que as formigas vivem em colônias, em que é possível identificar várias formas de organização, tanto para a construção do habitat, como para a aquisição de alimentos, reprodução e ainda cuidado com a prole (HÖLLDOBLER e WILSON, 1990, p. 3).

Um dos principais aspectos biológicos que inspiraram os estudos de computação voltados ao comportamento das formigas é o sistema de comunicação que alguns insetos utilizam. Grassé (1946, *apud* Dorigo, Birattari, Stützle, 2006, p. 29) percebeu durante seus estudos que algumas espécies de cupins produziam uma substância que oferecia novos estímulos tanto para o produtor, como para os outros cupins que entrassem em contato com ela. Isso foi classificado como uma forma de comunicação, denominada *Stimergy*, em que era possível identificar duas características primordiais:

- a comunicação era mediada através de modificações no ambiente, provocadas pelos insetos;

- os cupins que passavam pelos locais modificados recebiam novos estímulos.

O mesmo comportamento apresentado por cupins também pode ser encontrado em colônias de formigas no momento em que elas estão em busca de alimento. Esse processo de comunicação é mediado por uma substância denominada feromônio (DORIGO, BIRATTARI, STÜTZLE, 2006, p. 29).

Ferreira e Zarbin (1998, p. 3) afirmam que “feromônios são substâncias químicas secretadas por um indivíduo (nesse caso, um inseto) que permitem a comunicação com outro indivíduo da mesma espécie”. As diferenças dos feromônios utilizados pelas várias espécies de insetos podem ser percebidas em nível de estrutura molecular. Essas substâncias podem ser usadas com diversos fins: alarme, ataque, agregação ou marcação de trilhas, sendo este último uma das características atribuídas às colônias de formigas. Enquanto caminham, as formigas deixam trilhas de feromônio, que podem ser detectadas pelas outras formigas. Quanto mais feromônio existe em um determinado local, mais atrativo ele se torna, portanto, há uma maior probabilidade de que outras formigas passem pelo mesmo local. É importante lembrar ainda que o feromônio evapora com o passar do tempo (DORIGO e STÜTZLE, 2004, p. 12).

Denebourg et al. (1990, 159-163) fizeram um experimento com formigas operárias da espécie *Iridomyrmex humilis*, comumente conhecidas por formigas argentinas, que revelou como ocorre a exploração de um território no momento em que os insetos estão em busca de alimento. Inicialmente, foi feita uma ligação entre um ninho de formigas e uma fonte de alimento, localizados a 15 cm de distância. Essa ligação consistia em duas pontes, separadas por um ângulo de 60º, como apresenta a Figura 13.

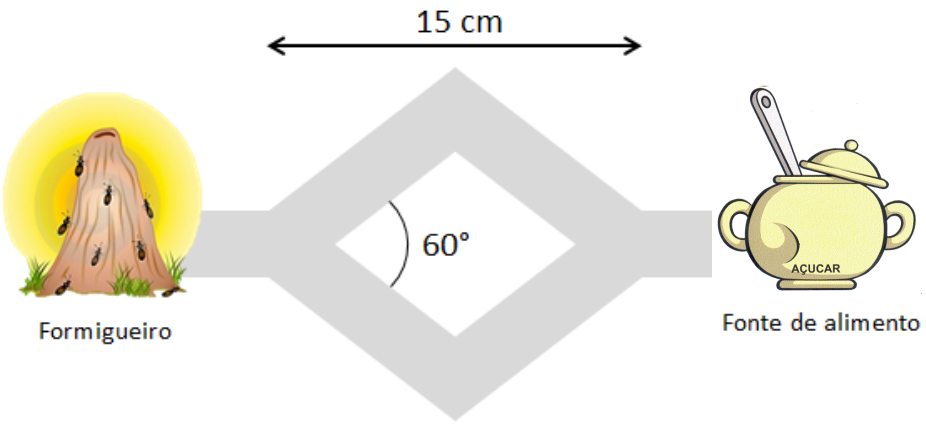


Figura 13: Pontes iguais (DORIGO e STÜZLE, 2004, p. 3, adaptado)

Considerando o cenário apresentado na Figura 13, ao sair do ninho, uma formiga começava a explorar o local, e quando percebia a fonte de comida, se dirigia até ela por uma das duas pontes, escolhida de forma randômica. Durante o percurso, ela depositava feromônio, tanto na ida quanto na volta. Após algum tempo, percebeu-se que a maioria das formigas estava optando pela ponte com maior concentração de feromônio. Ao passar por ela, mais feromônio era depositado, tornando essa ponte cada vez mais atrativa. Deve-se ressaltar que a escolha por uma das pontes foi ocasionada pela probabilidade, que em um certo momento, permitiu uma maior quantidade de formigas passar por uma das pontes.

Em um experimento realizado por Goss et al. (1989, 579-581), parecido com o anteriormente apresentado, foram utilizadas duas pontes de tamanhos distintos para ligar um ninho de formigas a uma fonte de alimento. O cenário deste experimento é apresentado na Figura 14.

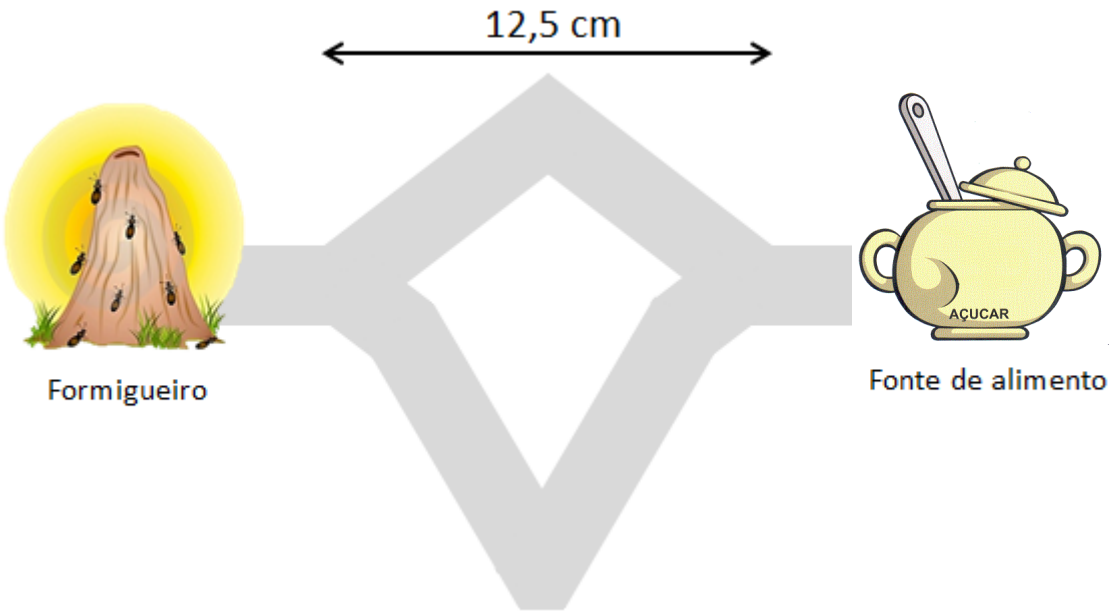


Figura 14: Pontes distintas (DORIGO e STÜZLE, 2004, p. 3, adaptado)

Durante os dez primeiros minutos, foi possível perceber que as formigas escolhiam randomicamente uma das duas pontes para seguir até a fonte de comida. Porém, rapidamente percebeu-se que a ponte menor era visivelmente preferida pelas formigas, pois apresentava uma maior concentração de feromônio. Tal fato é explicado devido à maior quantidade de formigas que passaram pela ponte menor

em relação à ponte maior, considerando um mesmo espaço de tempo. Para ilustrar o aumento de feromônio na ponte menor, sugere-se a seguinte hipótese:

- a ponte maior possuía duas vezes o comprimento da ponte menor;
- as formigas andavam nas pontes com a mesma velocidade, expelindo uma quantidade constante de feromônio enquanto caminham;
- as formigas gastavam um minuto para atravessarem a ponte menor (consequentemente, gastam 2 minutos para passarem pela ponte maior);

Considerando esses dados, ao colocar uma formiga na extremidade inicial de cada ponte, tem-se que após dois minutos, a ponte menor terá o dobro da concentração de feromônio em relação à outra ponte, visto que a formiga já conseguiu ir até a fonte de alimento e voltar para o formigueiro. Sendo assim, quando uma nova formiga se deparar com o cenário de escolha entre as pontes, irá detectar a trilha de feromônio mais acentuada na ponte menor, portanto, a probabilidade dela escolher essa ponte para realizar o percurso será maior.

Comparando os dois experimentos, percebe-se que o segundo caso foi influenciado não só pela probabilidade, como também pelo comprimento das pontes utilizadas para ligar o ninho até a fonte de alimento. Esse comportamento das formigas de optar pelo menor caminho pode ser aplicado a diversos problemas combinatórios complexos; muitos deles alvos de pesquisas na área da computação. É importante ressaltar que a importância desses insetos na inteligência artificial se dá ao considerar o trabalho em grupo e a vida em sociedade e não a simplicidade de uma única formiga. A próxima seção apresenta informações acerca do algoritmo *Ant System*, que foi construído a partir do comportamento das formigas apresentado nesta seção.

2.4.2. Abstração das formigas para a computação

De acordo com Dorigo e Stüzle (2004, p. 68), o *Ant System* (AS) foi o primeiro algoritmo proposto com base no comportamento das formigas, sendo que foi inicialmente aplicado no clássico *Traveling Salesman Problem*, em português: Problema do Caixeiro Viajante. Esse algoritmo serviu como base para várias técnicas que surgiram posteriormente, como o *Max-Min Ant System* e o *Elitist Ant System,* que propuseram algumas modificações no algoritmo original do AS para se

adequarem à outros tipos de problema e proverem melhor performance em cada contexto.

Dorigo, Maniezzo e Corloni (1996, p. 29-30) afirmam que o *Ant System* é versátil, permitindo que seja aplicado a variações de um mesmo problema com pequenas alterações para satisfazer seus requisitos. Além disso, ele possui robustez, tornando viável sua aplicação em várias áreas de análise combinatória. Outra característica do AS é o *feedback* positivo, visto que durante o processamento um agente modifica os níveis de feromônio nas trilhas, o que influencia diretamente a ação dos próximos agentes que passarem por esses locais, possibilitando uma conversão mais rápida para uma solução aceitável. Essa última característica também é conhecida por comportamento auto-catalítico.

Para trabalhar com o AS, inicialmente deve-se entender como ocorre a representação do comportamento das formigas naturais no computador. Posteriormente, é preciso saber qual o processamento é realizado sobre esses elementos, a fim de se ter soluções para um problema. As próximas seções deste trabalho apresentam tanto os elementos fundamentais do *Ant System* como a estrutura de processamento que ele implementa.

2.4.2.1. Os elementos fundamentais

Conforme os trabalhos realizados por Dorigo, Maniezzo e Corloni (1996); Dorigo e Stützle (2004); Dorigo, Birattari e Stützle (2006), para simular o comportamento das formigas no ambiente computacional, é preciso fazer a abstração de três elementos essenciais: o cenário da natureza, as formigas e o feromônio.

2.4.2.1.1. Cenário da natureza

O cenário da natureza é representado comumente como um grafo, em que um dos nós representa a fonte de comida e os outros nós são pontos de referência. Os elos entre os nós constituem as trilhas que são percorridas pelas formigas. Sendo assim, sabe-se que as formigas vão ser disparadas a partir de um nó do grafo com destino a essa fonte de comida, que é tida como objetivo, criando portanto um caminho, também chamado de solução.

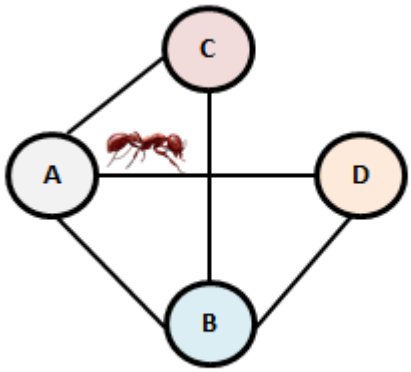
2.4.2.1.2. Formigas

As formigas são representadas através de agentes simples que fazem um percurso no grafo por uma determinada rota. Sendo assim, a partir de um nó do grafo, elas seguem por um elo, escolhido conforme uma taxa de probabilidade, até um nó vizinho. Essa probabilidade é influenciada pela quantidade de feromônio detectada pelo agente. A probabilidade deve existir em detrimento a uma escolha determinística, pois as formigas reais nem sempre optam pela trilha com maior quantidade de feromônio.

Além disso, é importante que esse agente armazene os nós já percorridos, para evitar que eles sejam visitados novamente durante a construção de uma solução. Sendo assim, ao chegar em um nó, a formiga pode escolher entre todos os nós vizinhos do nó em que se encontra, exceto aquele que ela acabou de passar (a não ser que esse seja o único nó para o qual ela pode seguir). Isso evita que o algoritmo crie *loops* infinitos sem chegar a um destino. Esse armazenamento do caminho percorrido também é importante para que no final seja possível conhecer a solução que foi gerada.

2.4.2.1.3. Feromônio

A quantidade de feromônio é expressa por um número, armazenado em cada nó do grafo. Esse armazenamento pode ser realizado conforme a estrutura apresentada na Figura 15.



(a) Grafo

Tabela do nó A	
B	10
C	15
D	20

(b) Tabela de feromônio

Figura 15: Estrutura para armazenar a quantidade de feromônio

Na Figura 15, pode-se perceber que o nó do grafo (a) indicado por A possui uma tabela (b), que o relaciona aos nós com os quais ele possui um elo. Nessa tabela, estão os valores que indicam a quantidade de feromônio presente em cada trilha. Cada nó do grafo deve possuir uma tabela indicando as quantidades de feromônio presentes em seus elos.

Após fazer a devida abstração dos elementos fundamentais que compõe o problema, é preciso implementar, de fato, as rotinas que irão calcular uma solução. A seção seguinte apresenta alguns detalhes acerca da estrutura do *Ant System*.

2.4.2.2. Estrutura do algoritmo

O algoritmo *Ant System* apresenta alguns pontos fundamentais durante o seu processamento (DORIGO e STÜTZLE, 2004, p. 12-15), conforme apresenta a Figura 16.

Pseudocódigo do algoritmo *Ant System*

Inicializar parâmetros;
Repetir (condição de parada)
{
Construir soluções;
//atualização do feromônio nas próximas 2 etapas
Evaporação do feromônio;
Atualização do feromônio pelas formigas;
}

Figura 16: Pseudocódigo do algoritmo Ant System

O primeiro passo consiste em inicializar alguns parâmetros, como o grafo a ser percorrido e a quantidade de agentes que irão participar do processo. Recomenda-se que o número de agentes seja igual ao número de nós presentes na estrutura que será percorrida. Além disso, pode-se iniciar o grafo sem nenhum feromônio, ou seja, a escolha pelos nós que serão visitados pelas formigas serão inteiramente randômicas. É possível também já iniciar o grafo com pequenas quantidades de feromônio, que deve ser aplicado igualmente em todos os elos. Posteriormente, deve-se criar um laço de repetição, que irá realizar um

processamento durante várias iterações, até se atingir uma solução aceitável para o problema. No primeiro ponto do laço de repetição, que corresponde à construção das soluções, pode-se adotar duas estratégias:

- implementação paralela: todas as formigas são liberadas ao mesmo tempo, a partir de uma origem até um ponto final;
- implementação sequencial: cada formiga deve executar sua trilha do início até o fim antes de outra formiga percorrer o grafo;

É importante ressaltar que, ao contrário das formigas verdadeiras, os agentes só atualizam as trilhas de feromônio após todas as formigas concluírem seus percursos. Isso ocorre para que a taxa de feromônio seja caracterizada pela qualidade do caminho que foi gerado na solução, de acordo ao contexto. Por exemplo: se o objetivo é determinar o menor caminho entre dois pontos do grafo, pode-se adicionar mais feromônio aos caminhos que apresentam um comprimento menor. Sendo assim, a escolha entre a implementação paralela ou sequencial não afeta os resultados do algoritmo. A convergência para uma solução aceitável se dá, portanto, ao aplicar o algoritmo em várias iterações. A probabilidade para a escolha de um nó, no momento da construção da solução, é obtida através da Equação 1.

$$p_{ij}^{k} = \frac{\tau_{ij}^{\alpha}}{\sum_{l \in N_i^k} \tau_{il}^{\alpha}}$$

Equação 1: Probabilidade de uma formiga visitar um nó

Na Equação 1, p_{ij}^{k} representa a probabilidade de uma formiga *k* visitar um nó *j*, a partir de um nó indicado por *i*. Esse nó *j* deve estar presente na vizinhança de *i*, da qual fazem parte vários nós, representados de forma geral por *l*. É importante observar que essa probabilidade depende da quantidade de feromônio presente no elo que liga o nó *i* ao nó *j*, representada por τ_{ij}^{α} e também da soma das quantidade de feromônio encontradas na vizinhança, representada pelo somatório presente no denominador da equação. O α indica o valor de uma constante, que pode ser utilizada para atenuar ou diminuir o impacto da quantidade de feromônio no problema. Geralmente esse α consiste no valor 1.

Portanto, pode-se concluir que a probabilidade de uma formiga passar por um elo consiste na divisão entre a quantidade de feromônio presente nesse elo pela

soma das quantidades de feromônio presentes nos elos de toda a vizinhança do nó na qual ela se encontra.

A segunda parte do AS corresponde à atualização do feromônio nas trilhas, que é realizada em duas etapas:

1. Assim como o feromônio natural, o algoritmo considera uma taxa de evaporação, possibilitando que soluções ruins tenham cada vez menos probabilidade de serem percorridas por formigas artificiais durante a execução. A evaporação do feromônio é obtida a partir da relação matemática apresentada na Equação 2.

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij}$$

Equação 2: Evaporação do feromônio

Assim que todas as formigas colocadas no grafo concluíram uma iteração, deve-se decrementar o feromônio presente em todas as trilhas do grafo. Esse decréscimo é realizado conforme um parâmetro *p*, que pode assumir valores presentes no intervalo (0,1] (ou seja, pode assumir valores maiores que 0 e menores ou iguais a 1).

2. Após a evaporação, as taxas de feromônio presentes na trilha criada por um agente são modificadas. A atualização dessas taxas é realizada de acordo com a Equação 3.

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta t^{k}$$

Equação 3: Incremento das taxas de feromônio

Conforme a Equação 3, a quantidade de feromônio presente em uma trilha que liga um nó *i* a um nó *j*, é incrementada a um valor expresso por Δt^{k}. Em casos simples, esse valor pode ser o mesmo para todas as formigas. Porém, dependendo do contexto, pode-se adaptar esse valor de acordo à trilha percorrida por cada formiga para representar particularidades do contexto. Por exemplo: em um problema para encontrar o menor caminho, pode-se adaptar esse valor de acordo ao comprimento de cada trilha, possibilitando que as trilhas menores tornem-se mais atrativas às formigas.

A próxima seção apresenta os materiais e a metodologia empregada no desenvolvimento deste trabalho.

3. MATERIAIS E MÉTODOS

Nessa seção, são apresentados os materiais utilizados no decorrer do trabalho e a metodologia utilizada para o desenvolvimento do algoritmo.

3.1. Materiais

Inicialmente foi realizada uma etapa de estudos acerca dos conceitos envolvidos no trabalho. Para isso, foram utilizados diversas fontes bibliográficas, como livros, monografias, dissertações, teses e artigos. Após os estudos, foi feita a modelagem e posteriormente o desenvolvimento do algoritmo.

Na fase de modelagem, foi utilizado o software Microsoft Visio 2010 para a criação de um diagrama de classes, representando as entidades envolvidas no contexto. Na implementação foi utilizada a linguagem Java, na IDE NetBeans 6.9.1. Além disso, é utilizada a API Jung 2.0, para a implementação e visualização dos grafos, que são compostos pelos objetos de aprendizagem envolvidos no sequenciamento dos estudos.

Esse ambiente foi utilizado no sistema operacional Windows 7 *Ultimate* – 32 bits, em uma máquina com processador Intel Dual Core 1,86GHz e memória RAM de 2 GB.

3.2. Metodologia

A primeira parte do trabalho consistiu na coleta e estudo de referencial teórico acerca do tema proposto, sendo que os principais materiais estudados foram livros e artigos científicos. A sequência de estudos foi iniciada a partir de conceitos sobre sequenciamento de estudos, para se obter um entendimento geral do contexto de trabalho, seguido do *Learning Object Metadata*, representação de usuário e por fim, o *Ant System*. A partir desses estudos, foi possível conhecer cada um dos temas e também realizar algumas abstrações acerca da interação entre elas, que foram importantes nas etapas seguintes.

A fase de execução do projeto ocorreu durante o semestre letivo de 2012/1, sendo que o primeiro procedimento consistiu em um estudo conjunto dos três temas, para a realização de um levantamento acerca do que deveria ser implementado.

Durante essa fase, foram designados os relacionamentos presentes entre os atributos do LOM e de uma representação do usuário acadêmico.

Posteriormente, foi criado um *Application Profile* (AP) a partir do LOM proposto pela IEEE. Os atributos escolhidos para a constituição deste AP foram baseados principalmente nos estudos realizados na fase anterior, visto que eles podem influenciar o comportamento do algoritmo. Para a constituição do AP, foi realizado um levantamento das características do LOM que poderiam ser relacionadas com os atributos de um usuário estudante. Das nove categorias estudadas, somente três foram escolhidas:

- geral: para o aproveitamento do identificador, palavras-chave e linguagens do objeto de aprendizagem;
- educacional: por possuir informações relacionáveis a um perfil, como dificuldade e nível de interatividade;
- relacionamentos: para a representação das relações inicias entre os objetos de aprendizagem, que forneceria uma base para a construção inicial da estrutura de dados do algoritmo.

Dentro de cada um desses grupos do LOM, foram escolhidos alguns atributos que poderiam ser representados de forma numérica, para a posterior utilização na realização de um cálculo de similaridade entre o usuário e o objeto de aprendizagem. Tal cálculo é explicado em detalhes na seção 4.2.6.

Depois de criado o AP, foi criada uma representação do usuário contendo os atributos necessários para o processamento do algoritmo. Os atributos do usuário foram escolhidos levando-se em consideração o AP criado na etapa anterior. Cada atributo do usuário teve uma característica equivalente a um atributo do objeto de aprendizagem. Essa relação de equivalência possibilitou a realização do cálculo de similaridade entre o objeto e o usuário.

Após a representação do usuário, foi iniciada a fase de desenvolvimento do algoritmo para a definição de sequências de estudo, baseado no *Ant System*. Antes de iniciar a codificação, foi realizado um breve estudo acerca de ferramentas e linguagens que poderiam ser utilizadas. A fase de desenvolvimento teve como produto os seguintes artefatos de software:

- diagrama de classes, para representar as classes utilizadas durante a implementação do algoritmo. Essas classes são caracterizadas tanto pela definição dos atributos, como também dos métodos que foram construídos;

- código do algoritmo.

Após o desenvolvimento do algoritmo, foi criado um cenário de teste para a realização de simulações. O cenário teve como base o contexto da disciplina de Estruturas de Dados. A partir dele, foi realizada uma análise do comportamento do algoritmo.

Na próxima seção, são apresentados os resultados atingidos com o desenvolvimento deste trabalho.

4. RESULTADOS E DISCUSSÃO

Após a realização dos estudos apresentados nas seções anteriores, foi feito um planejamento do algoritmo AntStudy seguido da etapa de codificação. Na etapa de planejamento, foi necessário estudar quais atributos do LOM seriam considerados nos objetos de aprendizagem que compõe o domínio do algoritmo e quais dados do usuário seriam relevantes.

Além desse estudo inicial, foi preciso sistematizar o algoritmo em classes e métodos, e ainda definir a comunicação entre esses elementos, conforme o processamento dos dados para gerar uma sequência de estudos. Os resultados do planejamento e implementação serão apresentados nas próximas subseções deste trabalho.

As próximas seções apresentam detalhes das classes implementadas no algoritmo AntStudy.

4.1. Diagrama de classes

A partir de um diagrama de classes, é possível identificar quais são as entidades que compõe um determinado contexto, juntamente com seus atributos e respectivos métodos. No AntStudy, as classes principais são:

- `AntSystem`: responsável pelo processamento do algoritmo para a construção de sequências de estudo;
- `ObjetoAprendizagem`: representa um material didático (os atributos dessa classe foram baseados nas características do LOM). Uma instância do objeto de aprendizagem pode ter vários relacionamentos (representados por uma classe auxiliar);
- `Usuario`: representa um usuário estudante, que possui um conjunto de características acerca do perfil de estudos;
- `Grafo`: estrutura de dados utilizada no algoritmo. Possui duas classes auxiliares: `Vertice` e `Aresta`.

Na Figura 17 é apresentado o diagrama de classes que representa o algoritmo AntStudy.

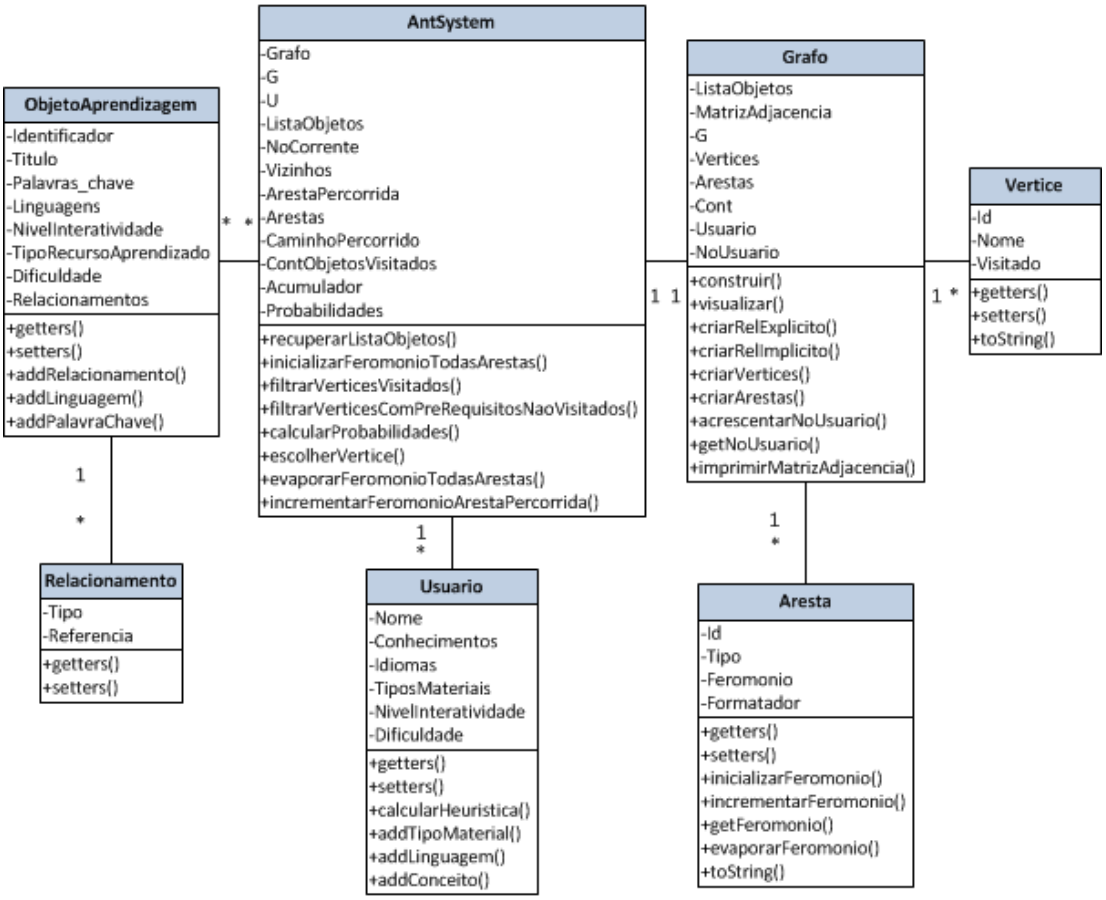


Figura 17: Diagrama de classes do AntStudy

4.2. Classes do algoritmo AntStudy

Nesta seção, serão apresentadas todas as classes do algoritmo e respectivos atributos e métodos. Inicialmente serão apresentadas as classes auxiliares e, por fim, a classe principal AntSystem, responsável pela definição da sequência de estudos.

4.2.1. Classe Grafo

A classe Grafo é responsável por realizar desde a montagem da estrutura do grafo de objetos de aprendizagem até a visualização dos componentes na interface em tempo de execução. Essa classe utiliza como base a implementação de grafo disponível a partir da API Jung. Para utilizar esse recurso, são requeridas duas classes, uma para representar os nós e outra para representar os elos da estrutura. Nesse trabalho foram criadas, portanto, as classes `Vertice` e `Aresta`, que serão apresentadas em detalhes nas próximas seções deste trabalho.

Os atributos presentes nessa classe são apresentados na Tabela 1, com seus respectivos significados.

Tabela 1: Atributos da classe Grafo

Atributo	Tipo	Significado
`ListaObjetos`	`ArrayList<ObjetoAprendizagem>`	Conjunto de instâncias dos objetos de aprendizagem.
`MatrizAdjacencia`	`int[][]`	Matriz contendo valores que representam os relacionamentos entre os objetos de aprendizagem.
`G`	`Graph<Vertice, Aresta>`	Objeto proveniente da API Jung, que fornece a implementação da estrutura de dados “grafo”.
`Vertices`	`ArrayList<Vertice>`	Conjunto de instâncias dos vértices.
`Arestas`	`ArrayList<Aresta>`	Conjunto de instâncias das arestas.
`Usuario`	`Usuario`	Objeto que representa o usuário para o qual está sendo definida a sequência de estudos.
`NoUsuario`	`Vertice`	Instância do vértice que representa o usuário no grafo.

O primeiro método presente nessa classe, chama-se `construir()`. Esse método serve para encapsular operações menores, que são realizadas para a construção do grafo com os objetos de aprendizagem. O primeiro passo para a construção do grafo consistiu na criação de uma estrutura semelhante a uma matriz de adjacências. Essa matriz apresenta na posição [i][j]:

- o valor 0, indicando que não há um relacionamento entre os objetos de aprendizagem identificados respectivamente por i e j;
- o valor 1, indicando que o elemento j tem como pré-requisito o objeto identificado por i. Nesse caso, os objetos apresentam um relacionamento explícito;
- ou o valor 2, indicando que os elementos i e j são pré-requisito para um terceiro objeto de aprendizagem. Esse relacionamento (chamado implícito) tornou-se necessário para que não fosse possível criar uma sequência de estudos desrespeitando os pré-requisitos determinados no domínio, composto pelos objetos de aprendizagem.

Na Figura 18 é possível visualizar um exemplo de matriz de adjacência e o respectivo grafo que poderia ser gerado a partir dela.

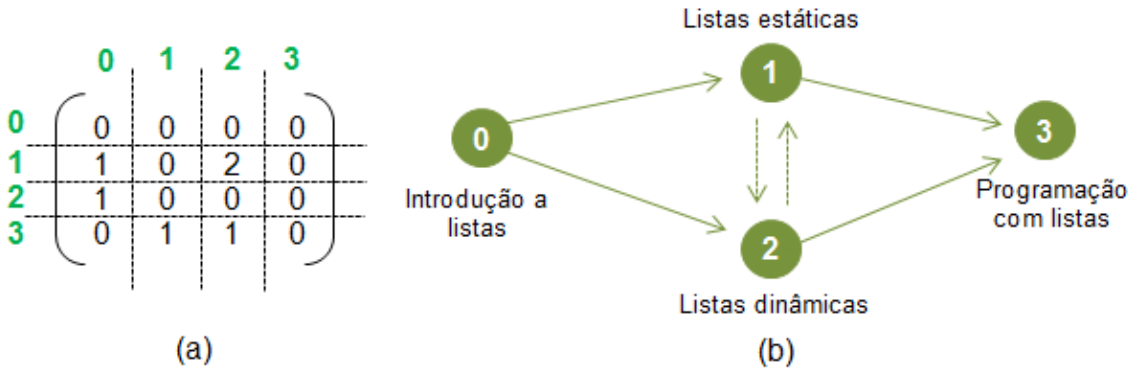


Figura 18: Exemplo de matriz de adjacência e respectivo grafo

A matriz de adjacência apresentada em (a) pode ser traduzida para o grafo apresentado em (b). A lista de objetos utilizada nesse exemplo apresenta os materiais na seguinte ordem: 0 - Introdução a listas, 1 - Listas estáticas, 2 - Listas dinâmicas e 3 - Programação com listas. O número 1 na posição [1][0] (considerando [linha][coluna]), indica que o material presente na posição 1 da lista de objetos (Listas estáticas) tem como pré-requisito o material 0 (Introdução a listas). Os números 1 presente na mesma linha, como ocorre em [3][1] e [3][2], indicam que os materiais presentes na posição 1 e 2 (Listas estáticas e dinâmicas) são pré-requisitos para o material presente na posição 3 (Programação com listas). Sendo assim, eles possuem um relacionamento implícito (representado na matriz pelo número 2). Nesse caso, os relacionamentos explícitos são representados por setas contínuas e os implícitos por setas tracejadas.

Para a criação da matriz de adjacências foram utilizados dois métodos, um para os relacionamentos explícitos `criarRelExplicito()` e outro para os relacionamentos implícitos `criarRelImplicito()`. O código para a criação dos relacionamentos explícitos pode ser visualizado na Figura 19.

```
104 public void criarRelExplicito() {
105   for (int i = 0; i < ListaObjetos.size(); i++) {
106     ObjetoAprendizagem corrente = ListaObjetos.get(i);
107     if (corrente.getRelacionamentos().size() > 0) {
108       for (int j = 0; j < corrente.getRelacionamentos().size(); j++) {
109         ObjetoAprendizagem preRequisito = corrente.
110                 getRelacionamentos().
111                 get(j).getReferencia();
112         int posicaoPreRequisito = ListaObjetos.indexOf(preRequisito);
113         if (MatrizAdjacencia[i][posicaoPreRequisito] != 1) {
114           MatrizAdjacencia[i][posicaoPreRequisito] = 1;
115         }
116       }
117     }
118   }
119 }
```

Figura 19: Método criarRelExplicito()

Na Figura 19, é feito um laço de repetição passando por cada objeto de aprendizagem presente em uma lista, denominada `ListaObjetos` (linha 105). Dentro desse laço, ocorre uma verificação para saber se objeto possui pré-requisitos (linha 107). Nesse caso, deve-se percorrer a lista de pré-requisitos, para verificar qual o número inteiro que identifica tal relacionamento (linha 108). A coleta desse número ocorre com a utilização do método `indexOf()` (linha 112), invocado a partir da instância da lista de materiais, que retorna a posição do objeto passado como parâmetro (que é o pré-requisito em questão). Após a recuperação desse identificador, é colocado o número 1 na matriz de adjacências, para informar a existência desse relacionamento, na devida posição que relaciona esses dois objetos (linha 114). Por exemplo: o objeto que está sendo analisado no laço de repetição é identificado pelo número 2 (está presente na segunda posição da lista). Esse objeto possui um pré-requisito que está na posição 7 da lista de objetos. Portanto, na posição [2][7] da matriz, será colocado o número 1.

A Figura 20 apresenta o método para a criação dos relacionamentos implícitos.

```
121 public void criarRelImplicito() {
122   for (int i = 0; i < MatrizAdjacencia.length; i++) {
123     for (int j = 0; j < MatrizAdjacencia.length; j++) {
124       for (int k = j + 1; k < MatrizAdjacencia.length - 1; k++) {
125         if (MatrizAdjacencia[i][j] == 1 &&
126                 MatrizAdjacencia[i][k] == 1) {
127           MatrizAdjacencia[j][k] = 2;
128         }
129       }
130     }
131   }
132 }
```

Figura 20: Métodos criarRelImplicito()

A partir da Figura 20, pode-se verificar que a lista de materiais não é percorrida novamente. A constituição dos relacionamentos implícitos ocorre com base nos valores presentes na matriz, que foi preenchida no método de criação de relacionamentos explícitos. Durante o percurso dessa matriz, é preciso verificar a existência de dois números 1 na mesma linha (linhas 125 e 126); caso em que ocorre a criação de um relacionamento implícito entre os objetos, portanto, coloca-se o valor 2 na matriz, na posição que relaciona esses dois materiais (linha 127).

Após a definição desses valores na matriz, é preciso criar os vértices e as arestas. Os vértices são criados no método `criarVertices()`, que instancia um vértice para cada objeto de aprendizagem presente na lista recebida como parâmetro no construtor da classe `Grafo`. Esse método é apresentado na Figura 21.

```
133 public void criarVertices() {
134   for (int i = 0; i < ListaObjetos.size(); i++) {
135     Vertice v = new Vertice(i, ListaObjetos.get(i).getTitulo());
136     Vertices.add(v);
137   }
138 }
```

Figura 21: Método criarVertices()

Analisando a Figura 21, observa-se que os vértices são instanciados (linha 135), mas ainda não fazem parte do grafo. Após a criação, eles são armazenados na lista denominada `Vertices` (linha 136).

O próximo método a ser executado é o `criarArestas()`, que faz uma varredura na matriz de adjacências (linhas 141 e 142) e estabelece os relacionamentos entre os vértices. Esse método é apresentado na Figura 22.

```
140 public void criarArestas() {
141   for (int i = 0; i < MatrizAdjacencia.length; i++) {
142     for (int j = 0; j < MatrizAdjacencia.length; j++) {
143       if (MatrizAdjacencia[i][j] == 1) {
144         G.addEdge(new Aresta(1), Vertices.get(j),
145                 Vertices.get(i), EdgeType.DIRECTED);
146         Cont++;
147       } else {
148         if (MatrizAdjacencia[i][j] == 2) {
149           G.addEdge(new Aresta(2), Vertices.get(j),
150                   Vertices.get(i), EdgeType.DIRECTED);
151           G.addEdge(new Aresta(2), Vertices.get(i),
152                   Vertices.get(j), EdgeType.DIRECTED);
153           Cont++;
154         }
155       }
156     }
157   }
158 }
```

Figura 22: Método criarArestas()

Para criar uma aresta, utiliza-se um método da API Jung, denominado `addEdge()` (linhas 144, 149 e 151), que é chamado a partir do atributo `G`. Esse método recebe como parâmetros uma instância da classe `Aresta`, as duas instâncias dos vértices que serão ligados e o tipo de ligação (direcionada ou não direcionada). Na criação das arestas, é importante ressaltar que:

- relacionamentos explícitos são representados por uma única aresta;
- relacionamentos implícitos são representados por duas setas tracejadas. É importante considerar duas setas, visto que não se sabe qual dos materiais será colocado primeiro na sequência de estudos. Sendo assim, independente da ordem, os pré-requisitos serão visitados.

Após criar os vértices e arestas no grafo, é preciso adicionar um vértice para representar o usuário. O método responsável por esse processo é apresentado na Figura 23.

```
160 public Vertice acrescentarNoUsuario() {
161   Vertice u = new Vertice(-1, Usuario.getNome());
162   for (int i = 0; i < MatrizAdjacencia.length; i++) {
163     boolean temPreRequisito = false;
164     for (int j = 0; j < MatrizAdjacencia.length; j++) {
165       if (MatrizAdjacencia[i][j] != 0) {
166         temPreRequisito = true;
167         break;
168       }
169     }
170     if (!temPreRequisito) {
171       G.addEdge(new Aresta(1), u, Vertices.get(i),
172               EdgeType.DIRECTED);
173     }
174   }
175   return u;
176 }
```

Figura 23: Método acrescentarUsuario()

O método `acrescentarUsuario()` faz uma varredura na matriz (linhas 162 e 164) para verificar quais objetos de aprendizagem não possuem pré-requisitos. Sendo assim, é estabelecida uma aresta entre o usuário e esses objetos (linha 171).

Nessa classe ainda existem alguns métodos auxiliares, como o `imprimirMatrizAdjacencia()`, responsável pela impressão da matriz em linhas e colunas, `getNoUsuario()`, que retorna o vértice referente ao usuário. Por fim, existe o método `visualizar()`, que gera um *JFrame* mostrando o grafo com os vértices, arestas, quantidade de feromônio. Na Figura 24, é apresentado um exemplo de visualização gerada pelo AntStudy.

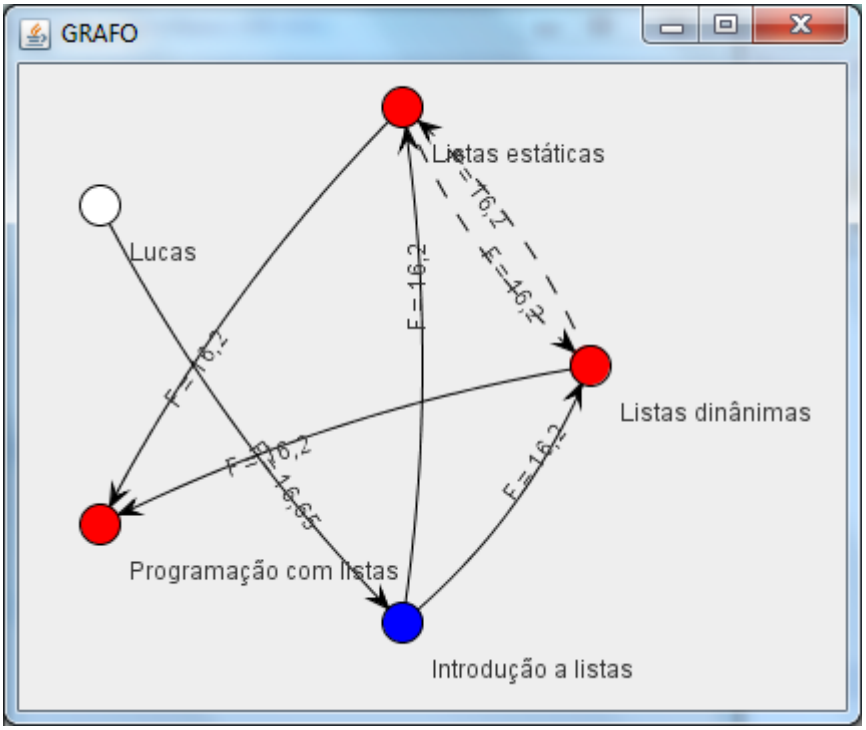


Figura 24: Exemplo de visualização no AntStudy

No *JFrame* apresentado na Figura 24 é possível perceber que existem vértices com três cores diferentes:

- branco representa o usuário;
- vermelho representa um objeto de aprendizagem não visitado;
- azul representa um objeto de aprendizagem já percorrido.

Em relação às arestas, há setas contínuas para os relacionamentos explícitos e setas tracejadas para os relacionamentos implícitos. Sobre cada seta é possível ver a quantidade de feromônio existente no elo.

Na seção seguinte são apresentados detalhes da classe `Aresta`, utilizada para a composição dos elos do grafo.

4.2.2. Classe Aresta

A classe Aresta representa uma ligação existente entre dois nós do grafo. No momento de definir uma instância dessa classe, é preciso passar como parâmetro somente um número inteiro definido o tipo da aresta: 1 - para relacionamentos explícitos e 2 - para relacionamentos implícitos. Além do atributo `Tipo`, as arestas

também possuem um atributo `Feromonio`, que é utilizado pelos métodos apresentados na Figura 25.

```
23 public double getFeromonio(){
24     return Feromonio;
25 }
26 
27 public void inicializarFeromonio(double Feromonio){
28     this.setFeromonio(Feromonio);
29 }
30 
31 public void incrementarFeromonio(double Incremento){
32     this.setFeromonio(this.getFeromonio() + Incremento);
33 }
34 
35 public void evaporarFeromonio(double TaxaEvaporacao){
36     this.setFeromonio((1 - TaxaEvaporacao) * this.getFeromonio());
37 }
```

Figura 25: Métodos que utilizam o atributo Feromonio

O método `inicializarFeromonio()` (linha 27-29) é invocado somente uma vez no início do algoritmo, para atribuir um valor inicial a todas as arestas que constituem o grafo. O método `getFeromonio()`(linhas 23-25) é utilizado na classe `AntSystem`, durante o cálculo da probabilidade de acesso a um determinado material. Já os métodos `incrementarFeromonio()`(linhas 31-33) e `evaporarFeromonio()` (linhas 35-37) são utilizados na última etapa do algoritmo para atualizar as taxas de feromônio após o usuário visualizar um determinado objeto de aprendizagem.

Para possibilitar a visualização das arestas, foi preciso sobrescrever o método `toString()`, visto que a API Jung invoca esse método no momento de desenhar o grafo no *JFrame*. Sendo assim, o retorno foi alterado para que fosse exibida a quantidade de feromônio presente na aresta, sendo que o valor foi formatado para apresentar somente dois números após a vírgula.

Na próxima seção, é apresentada a classe `Vertice`, responsável por representar os elementos ligados pelas arestas no grafo.

4.2.3. Classe Vertice

Essa classe é utilizada juntamente com a classe `Aresta` para a constituição dos elementos no grafo. De forma geral, um vértice pode representar tanto um objeto de aprendizagem como também um usuário. Assim como na classe `Aresta`, foi preciso sobrescrever o método `toString()` para que, no momento de visualização, fosse apresentado o nome do material.

Uma instância dessa classe possui como atributos: `Id`, para identificar o vértice, `Nome`, que no caso de um material recebe o título e no caso de um usuário recebe um nome próprio, e o atributo `Visitado`, que representa o estado do vértice (*true* ou *false*). Esse último atributo é importante principalmente na etapa inicial da definição da sequência de estudos, para que um usuário não passe pelo mesmo objeto mais de uma vez.

Na próxima seção do trabalho, é apresentada a classe `ObjetoAprendizagem`, responsável pela representação de um material didático no AntStudy.

4.2.4. Classe ObjetoAprendizagem

Essa classe representa um objeto de aprendizagem com seus respectivos atributos, que foram baseados na especificação do LOM. Tais atributos são apresentados em detalhes na Tabela 2.

Tabela 2: Atributos da classe ObjetoAprendizagem

Atributo	Tipo
`Identificador`	`int`
`Titulo`	`String`
`Palavras_chave`	`ArrayList<String>`
`Linguagens`	`ArrayList<String>`
`TipoRecursoAprendizado`	`String`
`NivelInteratividade`	`String`
`Dificuldade`	`String`
`Relacionamentos`	`ArrayList<Relacionamento>`

Das nove categorias definidas no LOM, foram contemplados alguns atributos de três categorias: Geral, Educação e Relacionamento. Os demais atributos não foram incluídos no algoritmo por não influenciarem no contexto deste trabalho. Para cada atributo escolhido, foram criados os seus respectivos *getters* e *setters*.

Para representar os relacionamentos entre dois objetos de aprendizagem, foi utilizada a classe auxiliar `Relacionamento`, que será apresentada na próxima seção deste trabalho.

4.2.5. Classe Relacionamento

A classe `Relacionamento` foi criada para auxiliar a classe `ObjetoAprendizagem`. Ela possui a função de englobar um relacionamento existente entre dois materiais; sendo assim, um objeto de aprendizagem possui uma lista de instâncias de `Relacionamento`, em que é possível indicar o **Tipo** e a **Referencia** a um objeto específico. Para cada um desses atributos foram criados seus respectivos *getters* e *setters*.

Na próxima seção é apresentada a classe `Usuario`, que é responsável pela representação de um usuário estudante no algoritmo de sequenciamento de estudos.

4.2.6. Classe Usuario

Essa classe promove a representação de um usuário, da qual são extraídas informações que influenciam na execução do algoritmo AntStudy. Os atributos escolhidos para a representação do usuário estão diretamente relacionados com os atributos dos objetos de aprendizagem. Eles são apresentados na Tabela 3.

Tabela 3: Atributos da classe Usuario

Atributo	Tipo
`Nome`	`String`
`Conceitos`	`ArrayList<String>`
`Linguagens`	`ArrayList<String>`
`TiposMateriais`	`ArrayList<String>`

`NivelInteratividade`	`String`
`Dificuldade`	`String`

Em relação às três dimensões indicadas para um modelo de usuário estudante, este trabalho contempla somente as duas primeiras, sendo o volume de dados definido como **estados finais** e o tipo de conhecimento **declarativo**. Tais conhecimentos são indicados a partir de um conjunto de palavras-chave, que informam quais conceitos o usuário possui conhecimento. Para o algoritmo definir as sequências de estudo não foi necessário desenvolver um modelo de diagnóstico para alimentar o modelo de estudante que foi definido. As características foram escolhidas a partir da análise de um objeto de aprendizagem, conforme a definição proposta pela IEEE.

O principal método presente nesta classe é o `calcularHeuristica()`. Esse valor heurístico é utilizado para definir a adequação de um objeto de aprendizagem a um determinado usuário. Para a realização do cálculo, foi necessário criar duas estruturas para armazenar de forma numérica as características definidas nas instâncias do objeto de aprendizagem e na representação do usuário. Essas estruturas foram definidas a partir de dois *HashMaps*, contendo chave e valor. Na Figura 26 é possível visualizar a composição da estrutura que é utilizada para caracterizar um material.

Estrutura geral

Palavras-chave	Linguagens	Tipo de recurso de aprendizado	Nível de interatividade	Dificuldade

Exemplo

Java	Estruturas	ptBR	Apresentação de slides	Nível de interatividade	Dificuldade
1	1	1	1	0.6	0.8

Figura 26: *HashMap* que caracteriza um objeto de aprendizagem

Na estrutura apresentada na Figura 26, as primeiras posições contemplam características booleanas. Sendo assim, no início dessa estrutura encontram-se todas as palavras-chave, as linguagens e o tipo de recurso de aprendizado do

material como chaves do *HashMap*, sendo que essas chaves referenciam o valor 1, para indicar a presença desses elementos. Para o nível de interatividade e dificuldade, os valores podem variar na escala de 1 a 5, sendo que após a indicação do valor, ocorre um processo de normalização (os valores são divididos por 5, que é o valor máximo desses atributos, para que fiquem representados em uma escala de 0 a 1).

Para representar o usuário é utilizada uma estrutura semelhante à do material, sendo que as palavras-chave correspondem aos conceitos que o usuário domina e as linguagens são os idiomas que ele consegue entender. É importante ressaltar que no *HashMap* do usuário também devem estar presentes as palavras-chave, linguagens e tipo de recurso de aprendizado do material em questão, para que o cálculo da adequação esteja correto. Supondo que um material está escrito em inglês e português, e o usuário domina somente o português, o *HashMap* deve ter uma entrada para português, referenciando o valor 1 e uma entrada para inglês, referenciando o valor 0, para indicar que ele não possui conhecimento nesse idioma.

Após a criação dos dois *HashMaps* é realizado o cálculo do co-seno entre os dois vetores de características. Nesse ponto poderia ser utilizada outra forma de determinar a similaridade entre os objetos. O co-seno foi escolhido somente para a realização de testes empíricos. Esse cálculo é realizado a partir da Equação 4.

$$sim = \frac{\sum_{i=1}^{n}(x_{ki} * y_{ki})}{\sqrt{\sum_{i=1}^{n}(x_{ki})^2} * \sqrt{\sum_{i=1}^{n}(y_{ki})^2}}$$

Equação 4: Cálculo do co-seno (SILVA, Edeilson et. al., 2011, p. 93, adaptada)

O cálculo da similaridade a partir do co-seno retorna um valor entre 0 e 1, sendo que quanto mais próximo de 1, mais similares são os dois vetores, portanto, mais indicado um objeto de aprendizagem a um determinado perfil de usuário. No cálculo apresentado na Equação 4, x_{ki} e y_{ki} correspondem aos valores presentes no *HashMap* do usuário e objeto de aprendizagem, respectivamente, em que i representa a chave (o cálculo é efetuado levando-se em consideração somente as chaves comuns das duas estruturas). Para exemplificar o cálculo, podem-se utilizar as duas estruturas apresentadas na Figura 27.

Objeto de aprendizagem

Programação	Estruturas	en	Apresentação de slides	Nível de interatividade	Dificuldade
1	1	1	1	0.4	0.8

Usuário

Programação	Estruturas	en	ptBR	Vídeo	Apresentação de slides	Nível de interatividade	Dificuldade
0	1	0	1	1	0	0.8	0.2

Figura 27: Estruturas para exemplo do cosseno

Nesse exemplo, pode-se observar que as chaves semelhantes são: Programação, Estruturas, en, Apresentação de Slides, Nível de interatividade e Dificuldade. O passo a passo do cálculo é apresentado na Figura 28.

Numerador

$$1 * 0 + 1 * 1 + 1 * 0 + 1 * 0 + 0{,}4 * 0{,}8 + 0{,}8 * 0{,}2 = 1{,}48$$

Denominador (parte do objeto de aprendizagem)

$$\sqrt{1^2 + 1^2 + 1^2 + 1^2 + 0{,}4^2 + 0{,}8^2} = \sqrt{4{,}8} = 2{,}19$$

Denominador (parte do usuário)

$$\sqrt{0^2 + 1^2 + 0^2 + 0^2 + 0{,}8^2 + 0{,}2^2} = \sqrt{1{,}68} = 1{,}29$$

Valor da similaridade

$$\frac{1{,}48}{2{,}19 * 1{,}29} = \frac{1{,}48}{2{,}82} = 0{,}52$$

Figura 28: Exemplo do cálculo do co-seno

Como o valor de similaridade está presente em uma escala de 0 a 1, pode-se inferir que o valor resultante do cálculo apresentado na Figura 28 corresponde a um grau de similaridade mediano entre o objeto e o usuário. Portanto, o objeto de aprendizagem em questão estaria parcialmente adequado ao perfil do usuário.

Na próxima seção do trabalho é apresentada a classe principal do algoritmo, denominada `AntSystem`.

4.2.7. Classe AntSystem

Essa é a principal classe do projeto, visto que nela se realiza a definição da sequência de estudos para um determinado usuário. Os atributos desta classe e seus respectivos significados são apresentados na Tabela 4.

Tabela 4: Atributos da classe AntSystem

Atributo	Tipo	Significado
`Grafo`	`Grafo`	Armazena uma instância da classe Grafo, para trabalhar com a construção da matriz de adjacências e do grafo de objetos de aprendizagem.
`G`	`Graph`	Recebe uma instância da estrutura de dados contendo os vértices e arestas.
`U`	`Usuario`	Instância do usuário para o qual será definida a sequência de estudos.
`ListaObjetos`	`ArrayList<Objeto Aprendizagem>`	Conjunto de instâncias dos objetos de aprendizagem.
`NoCorrente`	`Vertice`	Instância do vértice que está sendo visualizado por um estudante em tempo de execução.
`Vizinhos`	`List<Object>`	Conjunto de vizinhos do `NoCorrente`.
`ArestaPercorrida`	`Aresta`	Instância da aresta que acabou de ser percorrida, para que seja feita a atualização da taxa de feromônio.
`Arestas`	`List<Object>`	Conjunto de arestas presentes no grafo.
`CaminhoPercorrido`	`ArrayDeque<Vertice>`	Pilha para armazenar os nós que foram percorridos.
`ContObjetosVisitados`	`int`	Contador que armazena a quantidade de objetos visitados em tempo de execução. Esse atributo é utilizado como *flag* do principal laço de repetição executado no algoritmo.
`Acumulador`	`double`	Somatório das probabilidades de acesso aos vértices da lista `Vizinhos`.

`Probabilidades`	`double[]`	Vetor que armazena as probabilidades de acesso a cada um dos vértices presentes na lista `Vizinhos`.

O algoritmo presente nessa classe contempla as três fases propostas para um algoritmo genérico baseado na técnica do *Ant System*: construção do grafo, busca de uma solução e posterior atualização das taxas de feromônio. Para a primeira parte do processamento, foi utilizado o método `recuperarListaObjetos()`, para instanciar os objetos de aprendizagem com seus respectivos atributos e colocá-los em uma lista. Em trabalhos futuros, pode-se modificar este método para que ele faça acesso a banco de dados, por exemplo. O importante é que no final do processo, a lista de objetos seja devidamente preenchida.

Posteriormente, deve-se instanciar o usuário para o qual será definida a sequência de estudos. A construção do grafo é feita a partir da invocação do método construir, presente na classe `Grafo` (apresentada anteriormente). Com o usuário instanciado e o grafo já construído, devem-se inicializar as taxas de feromônio igualmente em todas as arestas do grafo. Para esse processo foi utilizado o método `inicializarFeromonioTodasArestas()`, cujo código é apresentado na Figura 29.

```
190 public static void inicializarFeromonioTodasArestas() {
191   Arestas = new ArrayList(Arrays.asList(G.getEdges().toArray()));
192   for (Object o : Arestas) {
193     ((Aresta) o).inicializarFeromonio(20);
194   }
195 }
```

Figura 29: Método inicializarFeromonioTodasArestas()

Para acessar as arestas, utiliza-se o método `getEdges()` (linha 191), fornecido pela API Jung. Esses objetos são retornados no formato `Object`, portanto, deve-se fazer um *cast* (conversão para a classe `Aresta`) para acessar o método da classe `Aresta` responsável pela inicialização do feromônio em cada elo (linha 193).

Depois de inicializar as taxas de feromônio, o algoritmo executa um laço de repetição para a realização de um percurso completo pelo grafo, contemplando

todos os objetos de aprendizagem. Nessa etapa, utiliza-se a variável `NoCorrente` para armazenar qual vértice está sendo visitado, no momento da execução. O percurso se inicia a partir do vértice que representa o usuário. Nesse algoritmo, a figura da formiga encontra-se implícita na execução do algoritmo ao sair desse `NoCorrente` e percorrer o grafo pelos objetos de aprendizagem para cada usuário. A cada objeto visitado, tem-se uma solução e, portanto, a atualização das taxas de feromônio. O percurso é finalizado quando todos os objetos são visitados.

O primeiro processo presente nesse laço de repetição é a recuperação dos vizinhos do `NoCorrente`. Eles são retornados pelo método `getSuccessors()` (API Jung) e armazenados na variável `Vizinhos`. Depois disso ocorre a primeira filtragem: não se pode incluir na sequência de estudos um objeto de aprendizagem que já foi visitado. O código dessa filtragem é exibido na Figura 30.

```
197 public static void filtrarVerticesVisitados() {
198   ArrayList<Vertice> verticesParaRetirar = new ArrayList<Vertice>();
199   for (int i = 0; i < Vizinhos.size(); i++) {
200     if (((Vertice) Vizinhos.get(i)).isVisitado()) {
201       verticesParaRetirar.add((Vertice) Vizinhos.get(i));
202     }
203   }
204
205   for (int i = 0; i < verticesParaRetirar.size(); i++) {
206     Vizinhos.remove(verticesParaRetirar.get(i));
207   }
208 }
```

Figura 30: Método filtrarVerticesVisitados()

Para realização dessa filtragem, percorre-se a lista de vizinhos (linha 199) e verifica-se o atributo `Visitado` de cada vértice (linha 200). Os vértices já visitados são incluídos em uma lista (linha 201), para a remoção (linhas 205-207). Posteriormente, deve-se realizar a filtragem dos objetos de aprendizagem que possuem pré-requisitos que não foram visitados. O código responsável por esse processo é apresentado na Figura 31.

```
210 public static void filtrarVerticesComPreRequisitosNaoVisitados() {
211   ArrayList<Vertice> verticesParaRetirar = new ArrayList<Vertice>();
212   for (int i = 0; i < Vizinhos.size(); i++) {
213     ObjetoAprendizagem o = ListaObjetos.get(((Vertice) Vizinhos
214             .get(i)).getId());
215
216     for (Relacionamento preRequisito : o.getRelacionamentos()) {
217       ObjetoAprendizagem ref = preRequisito.getReferencia();
218       Vertice v = Grafo.Vertices.get(ListaObjetos.indexOf(ref));
219       if (!v.isVisitado()) {
220         verticesParaRetirar.add((Vertice) Vizinhos.get(i));
221       }
222     }
223   }
224
225   for (int i = 0; i < verticesParaRetirar.size(); i++) {
226     Vizinhos.remove(verticesParaRetirar.get(i));
227   }
228 }
```

Figura 31: filtrarVerticesComPreRequisitosNaoVisitados()

Esse método percorre a lista de vizinhos (linha 212) e, para cada vizinho, recupera os seus pré-requisitos (linha 216) e verifica se eles foram visitados (linha 219). Caso isso não tenha ocorrido, esse vértice é colocado na lista auxiliar `verticesParaRetirar` (linha 220) e, em uma etapa posterior, ele é retirado da lista de vizinhos do `NoCorrente` (linhas 225-227).

Os objetos que passarem pela etapa de filtragem seguem para a próxima etapa do algoritmo, que indicará qual objeto será colocado na sequência naquele momento. Essa indicação é baseada em uma probabilidade, que é influenciada tanto pela quantidade de feromônio presente na aresta, como também por um valor heurístico, que indica o quanto um objeto de aprendizagem é adequado a um usuário. Nesse trabalho, a probabilidade foi calculada com base em uma sentença matemática proposta por Dorigo e Stützle (2004, p. 70) para o problema do Caixeiro Viajante, que é apresentada na Equação 5.

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\gamma_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\gamma_{il}]^{\beta}}, se\ j \in N_i^k$$

Equação 5: Cálculo da probabilidade de acesso a um objeto de aprendizagem

Na Equação 5, τ_{ij}representa um valor heurístico que caracteriza a aresta entre o vértice i e o vértice j, γ_{ij}remete-se à quantidade de feromônio presente nesse elo e os expoentes α e β determinam o peso que cada um desses dois valores vai ter na definição da probabilidade. A letra k representa uma formiga (nesse caso um usuário) e N_i^k é o conjunto de vértices que k pode acessar a partir do vértice i.

A Tabela 5 apresenta alguns valores de heurísticas e quantidades de feromônio para desenvolver um exemplo da Equação 5.

Tabela 5: Exemplo do cálculo de probabilidade

Vértice	Heurística	Feromônio
Listas estáticas	0,2	5
Listas dinâmicas	0,4	8

Um usuário encontra-se no vértice “Introdução a listas” e o algoritmo precisa definir qual o próximo material a ser acessado, sendo que as possibilidades são “Listas estáticas” ou “Listas dinâmicas”. Considerando α sendo igual a 1 e β igual a 2, as probabilidades de acesso aos dois vértices são apresentadas na Figura 32.

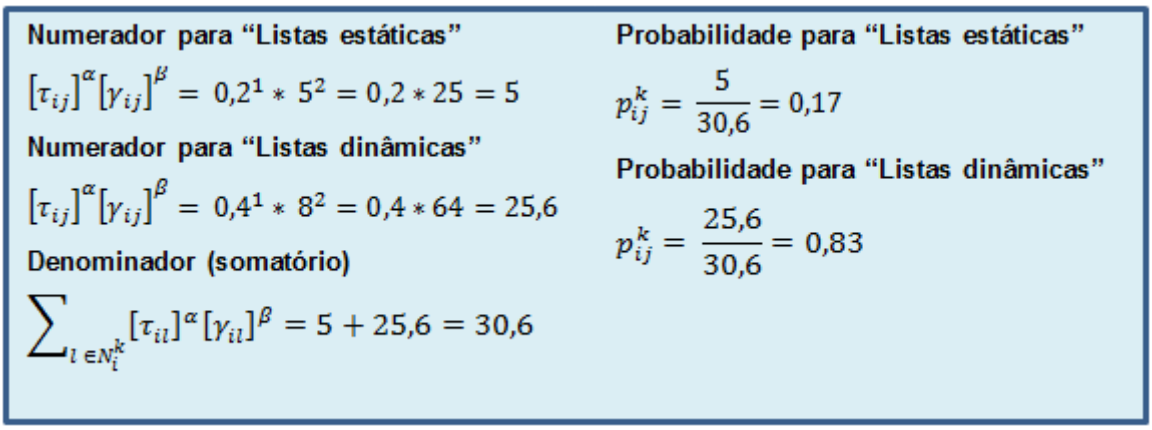


Figura 32: Exemplo do cálculo da probabilidade

Portanto, o material “Listas estáticas” tem 17% de chances de ser acessado e o material “Listas dinâmicas” tem 83%.

Inicialmente, o método `calcularProbabilidades()` encontra o valor do somatório utilizado no denominador da equação e posteriormente calcula os numeradores. Após a divisão ser realizada, a probabilidade de acesso ao objeto na

posição i da lista de vizinhos é armazenada em `Probabilidades[i]`, em que i identifica a posição do vizinho.

Após o cálculo das probabilidades, é invocado o método `escolherVertice()`, que irá indicar qual objeto será colocado na sequência. Para exemplificar a implementação desse evento, considera-se o exemplo apresentado a partir da Tabela 5. Essas duas taxas são colocadas em um intervalo, compreendido entre os números 0 e 1, portanto, cada objeto passa a ter como correspondência uma faixa de valores, conforme apresentado na Figura 33.

Figura 33: Exemplo das faixas de valores na escolha do vértice

A ordem dos valores é definida conforme a posição de cada um deles na lista de vizinhos. Posteriormente, gera-se um número *random* e verifica-se em qual intervalo ele se encontra; o objeto designado ao intervalo indicado é, portanto, escolhido como próximo vértice a fazer parte da sequência de estudos. O código para a escolha do vértice é apresentado na Figura 34.

```
256 public static void escolherVertice() {
257   double x = Math.random();
258   double inicioFrequencia = 0;
259   double fimFrequencia = Probabilidades[0];
260   List<Object> arestasEntreDoisNos = null;
261   for (int i = 0; i < Probabilidades.length; i++) {
262     if (x >= inicioFrequencia && x < fimFrequencia) {
263       arestasEntreDoisNos = new ArrayList(Arrays.asList
264               (G.findEdgeSet(NoCorrente,
265               (Vertice) Vizinhos.get(i)).toArray()));
266       CaminhoPercorrido.push(NoCorrente);
267       ContObjetosVisitados++;
268       NoCorrente = ((Vertice) Vizinhos.get(i));
269       break;
270     } else {
271       inicioFrequencia = fimFrequencia;
272       fimFrequencia += Probabilidades[i + 1];
273     }
274   }
275   ArestaPercorrida = (Aresta) arestasEntreDoisNos.get(0);
276 }
```

Figura 34: Método escolherVertice()

Analisando a Figura 34, observa-se que inicialmente é gerado um valor aleatório (linha 257). Para representar os intervalos contendo as probabilidades, são utilizadas faixas de valores, cujo início é representado pela variável `inicioFrequencia` e o fim pela variável `fimFrequencia`. O primeiro intervalo é iniciado em 0 (linha 258) e termina no valor da probabilidade de acesso ao primeiro objeto (linha 259). Se o valor do número aleatório estiver dentro dessa faixa (linha 262), o objeto que está relacionado ao intervalo será escolhido para a sequência de estudos. Nessa etapa, deve-se incrementar a quantidade de objetos presentes na sequência (linha 267). É preciso ainda armazenar a aresta entre o `NoCorrente` e esse objeto (linhas 263 e 275), para a que a quantidade de feromônio seja incrementada, em uma etapa posterior.

Antes de atribuir à variável `NoCorrente` o novo nó que pertence à sequência de estudos, o objeto de aprendizagem que está sendo visualizado é adicionado à uma pilha, denominada `CaminhoPercorrido`. Essa estrutura de dados auxilia no processo de continuação da sequência de estudos quando um usuário está visualizando um objeto de aprendizagem que não tem mais sucessores. Nesse caso, o algoritmo executa uma espécie de *backtracking*, passando pelos

vértices que foram visitados para verificar os sucessores que ainda não foram visitados. Essa situação é exemplificada a partir da Figura 35.

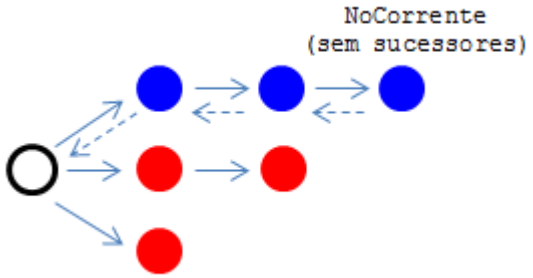


Figura 35: Exemplo de *backtracking*

No exemplo apresentado na Figura 35, os objetos de aprendizagem de cor azul já foram visitados e os objetos vermelhos ainda não foram percorridos. O vértice branco identifica o nó do usuário. No cenário apresentado, o `NoCorrente` não tem mais sucessores, portanto, o algoritmo realiza o processo de *backtracking* e vai retornando pelos vértices percorridos anteriormente (as setas tracejadas indicam o percurso de retorno).

Esses vértices são retirados da pilha que armazena o caminho percorrido. Cada elemento retirado da pilha é submetido ao processo: se há sucessores disponíveis, o algoritmo é executado, caso contrário, retira-se o topo da pilha e repete-se a verificação. O algoritmo iria retirar elementos da pilha até chegar ao vértice branco, que possui três sucessores (um visitado e dois disponíveis) e continua a execução do algoritmo para definir o novo material da sequência de estudos.

Para evitar que esse percurso fosse executado infinitamente, utiliza-se um contador como *flag* do *loop* principal: se esse contador for igual ao número de objetos presente na lista de materiais, o *loop* é encerrado e o algoritmo é finalizado. Após a escolha do vértice, inicia-se a etapa de atualização das taxas de feromônio no grafo:

1. primeiramente, é aplicada uma taxa de evaporação de 0,01 a todas as arestas do grafo, para possibilitar que os caminhos pouco percorridos se tornem depreciados pelo algoritmo com o passar do tempo. Essa taxa é equivalente a 1% da quantidade de feromônio atual. Na proposta de Dorigo e Stützle (2004, p. 71), essa taxa receberia um valor de 0,5. No AntStudy, não foi utilizado o valor original, pois em testes empíricos, identificou-se que as

taxas de feromônio decresciam de forma acelerada a cada execução. No algoritmo original, o valor de 50% mostrou-se adequado devido à escala de valores que eram trabalhadas no grafo (distâncias entre cidades, na resolução do Problema do Caixeiro Viajante). Sendo assim, esse valor foi sendo adequado até que finalmente utilizou-se a taxa de 1%.

2. posteriormente, a aresta percorrida recebe uma taxa de incremento de feromônio, calculada a partir da Equação 6.

$$\Delta\gamma_{ij} = \mu * \tau_{ij}$$

Equação 6: Incremento de feromônio no AntStudy

Na equação apresentada, a taxa de variação de feromônio ($\Delta\gamma_{ij}$) aplicada como incremento em uma aresta é resultado da multiplicação entre a nota fornecida pelo usuário ao objeto de aprendizagem (μ, que varia entre 1 e 10) e pelo valor heurístico calculado entre o usuário e o material.

Todos esses processos de filtragem, cálculo de probabilidades, escolha do vértice e atualização do feromônio se repetem até que todos os nós do grafo sejam visitados; finalizando, assim, o algoritmo. Tem-se então, uma sequência de estudos definida, com base nas características do usuário e dos objetos de aprendizagem presentes no contexto. Deve-se ressaltar que, a cada escolha de um material para a sequência de estudos, ocorre um incremento da quantidade de feromônio presente na aresta que foi percorrida.

A próxima seção possui detalhes do teste que foi realizado para demonstrar o funcionamento do algoritmo.

4.3. Exemplo de funcionamento do algoritmo

Para demonstrar o funcionamento do algoritmo, foi criado um cenário contendo algumas instâncias de estudantes e objetos de aprendizagem fictícios. A Tabela 6 apresenta as características dos usuários criados nesse cenário.

Tabela 6: Características dos estudantes utilizados na demonstração

Usuários / Atributos	João	Maria	Jorge	Joana
Idiomas	ptBR	ptBR en	ptBR	ptBR en
Conhecimentos	vetor, pilha, fila	lista estática, lista dinâmica, pilha, java, c, netbeans	recursividade, árvore, lista estática	lista estática, lista dinâmica, pilha, árvore, recursividade
Nível de interatividade	5	2	3	3
Tipos de materiais preferidos	vídeo slides	slides texto exercício	vídeo texto	vídeo slides apostila
Dificuldade almejada	1	4	2	3

Conforme os dados apresentados na Tabela 6, é possível perceber que foram criados usuários com diferentes características. João e Jorge só possuem domínio do idioma “português” e preferem materiais com baixo nível de dificuldade. Além disso, esses dois usuários possuem somente três conceitos vinculados à característica “Conhecimentos”. Já Maria e Joana conseguem lidar com materiais tanto em português como em inglês e preferem materiais com nível de dificuldade mais elevado. Ambas possuem uma quantidade maior de conhecimentos, em comparação a João e Jorge.

Em relação aos materiais, foram criados seis objetos de aprendizagem, também com características distintas. Todos os materiais criados possuem relação com a disciplina de “Estruturas de dados”, comum a qualquer curso de computação. Os detalhes dessas instâncias são apresentados na Tabela 7.

Tabela 7: Características dos objetos de aprendizagem

Objetos / Atributos	Introdução a listas	Listas estáticas	Listas dinâmicas	Programação com listas	Introdução a pilhas	Aplicação de pilhas
Linguagens	ptBR	en	ptBR	ptBR	en	ptBR
Palavras-chave	lista estática lista dinâmica	lista estática vetor	lista dinâmica	netbeans java lista estática lista dinâmica	pilha	netbeans java pilha

Nível de interatividade	2	2	4	2	1	2
Tipo de recurso de aprendizado	slides	apostila	exercício	vídeo	artigo	slides
Dificuldade	1	3	2	4	2	5

Dentre os materiais, pode-se perceber que são contempladas várias características, como linguagens diferentes (português e inglês), altos e baixos níveis de interatividade e dificuldade. Além das características apresentadas na Tabela 7, os materiais também possuem relacionamentos, conforme a Figura 36.

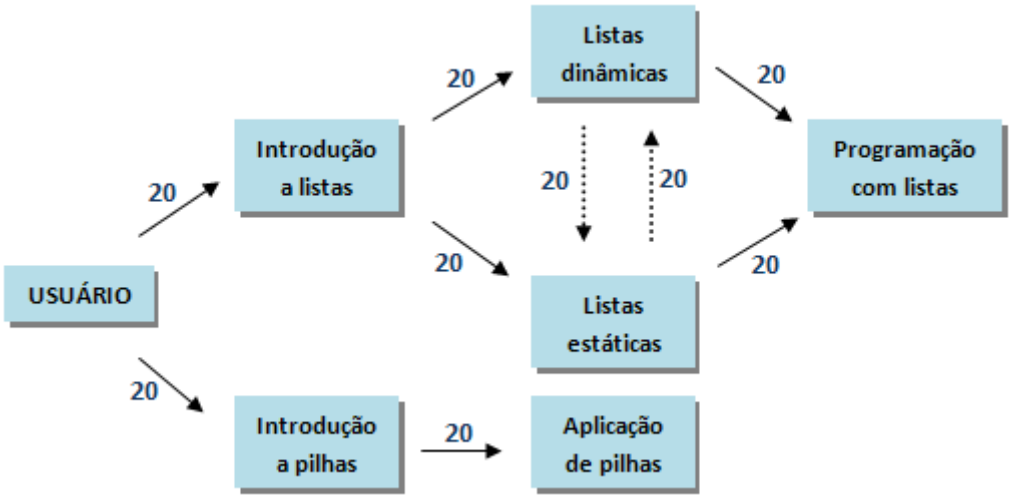


Figura 36: Disposição inicial dos materiais utilizados na demonstração

Na Figura 36, pode-se perceber tanto a existência de relacionamentos explícitos como relacionamentos implícitos entre os objetos de aprendizagem. Além disso, é possível visualizar a quantidade de feromônio inicial presente em cada aresta (20). A partir desse grafo, constata-se que existem dois ramos principais: um inicia-se a partir de “Introdução a listas” e o outro por “Introdução a pilhas”. Além disso, existe uma bifurcação no primeiro ramo, entre os materiais sobre “Listas dinâmicas” e “Listas estáticas”. Considerando todas as possibilidades, sabe-se que o total de sequências é igual a quatro:

1. Introdução a listas, Listas dinâmicas, Listas estáticas, Programação com listas, Introdução a pilhas e Aplicação de pilhas;
2. Introdução a listas, Listas estáticas, Listas dinâmicas, Programação com listas, Introdução a pilhas e Aplicação de pilhas;

3. Introdução a pilhas, Aplicação de pilhas, Introdução a listas, Listas dinâmicas, Listas estáticas, Programação com listas;
4. Introdução a pilhas, Aplicação de pilhas, Introdução a listas, Listas estáticas, Listas dinâmicas, Programação com listas.

Após instanciar os estudantes e objetos de aprendizagem, foi feita uma alteração no código do algoritmo, para que o laço de repetição principal (contemplando as etapas de construção da solução e atualização das taxas de feromônio) fosse executado para cada instância de usuário presente em uma lista. A partir dos dados apresentados na Tabela 7 e da Figura 36, o algoritmo foi colocado em execução. A matriz de adjacências gerada para a situação e o respectivo *JFrame* mostrando a visualização do grafo composto a partir de tais informações é apresentado na Figura 37.

Matriz de adjacências

0 – Introdução a listas
1 – Listas estáticas
2 – Listas dinâmicas
3 – Programação com listas
4 – Introdução a pilhas
5 – Aplicação de pilhas

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	1	0	2	0	0	0
2	1	0	0	0	0	0
3	0	1	1	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	1	0

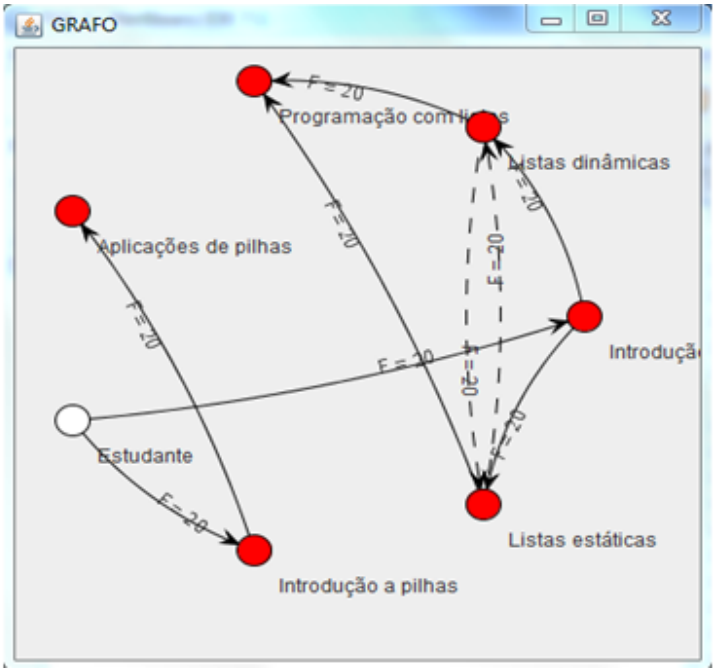


Figura 37: Matriz de adjacências e *JFrame* do teste

A partir da matriz de adjacências presente na Figura 37, observa-se a existência de um relacionamento implícito entre os objetos com identificadores 1 e 2 (considerando matriz[linha][coluna], matriz[1][2] = 2). Esses objetos, “Listas estáticas” e “Listas dinâmicas”, são pré-requisito para o objeto de “Programação com listas”. O relacionamento implícito é feito a partir da criação de duas arestas, representadas pelas setas pontilhadas, conforme apresentado no *JFrame*.

Na etapa de incremento da quantidade de feromônio no elo percorrido, o algoritmo utiliza uma nota, presente em uma escala de 0 a 10, fornecida pelo usuário. Para essa demonstração, foram utilizadas as notas presentes na Tabela 8.

Tabela 8: Notas utilizadas na demonstração

Objetos de aprendizagem	Notas			
	João	Maria	Jorge	Joana
Introdução a listas	5	10	5	9
Listas estáticas	6	8	8	8
Listas dinâmicas	7	8	6	9
Programação com listas	5	9	4	10
Introdução a pilhas	7	10	4	9
Aplicação de pilhas	8	9	4	10

Após a execução do algoritmo, obtiveram-se as sequências de estudo apresentadas na Figura 38.

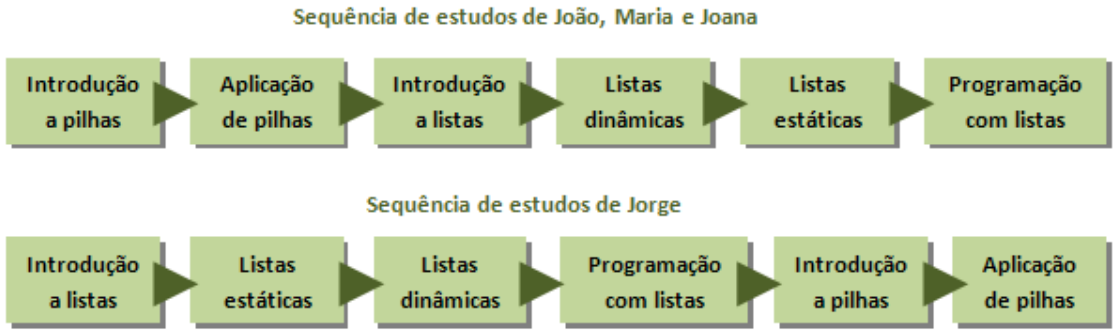
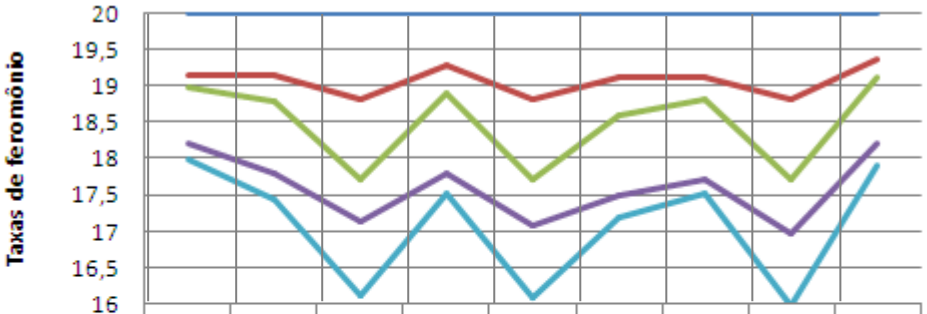


Figura 38: Sequências de estudo obtidas como resultado do teste

A partir da Figura 38, percebe-se que somente duas das quatro possíveis sequências foram originadas como resultados para os usuários instanciados. Nesse cenário, é possível observar que em 100% dos casos em que os estudos se iniciam com “Introdução a pilhas”, aparece o material de “Listas dinâmicas” como sucessor do material “Introdução a listas”. Já quando os estudos são iniciados a partir do ramo de listas, tem-se como sequência do material introdutório o recurso denominado “Listas estáticas”. Outro fato que se pode observar é que João, Maria e Joana possuem conhecimento do conteúdo “pilha” e as sequências de estudos desses três estudantes se iniciaram a partir de materiais abordando esse assunto.

O gráfico apresentado na Figura 39 mostra as variações de feromônio em cada aresta (numeradas de 1 a 9) após a geração da sequência de estudos de cada estudante utilizado nessa demonstração.



	1	2	3	4	5	6	7	8	9
Inicial	20	20	20	20	20	20	20	20	20
João	19,16	19,16	18,83	19,29	18,83	19,12	19,11	18,83	19,38
Maria	18,97	18,80	17,72	18,91	17,72	18,59	18,82	17,72	19,11
Jorge	18,20	17,79	17,12	17,80	17,08	17,50	17,72	16,97	18,20
Joana	17,99	17,43	16,12	17,51	16,08	17,17	17,51	15,98	17,91

Figura 39: Gráfico das variações de feromônio

A Tabela 9 apresenta os vértices que estão ligados pelas arestas indicadas pelos números de 1 a 9, no gráfico da Figura 39.

Tabela 9: Arestas da demonstração

Aresta	Vértice A	Vértice B
1	Usuário	Introdução a listas
2	Usuário	Introdução a pilhas
3	Introdução a listas	Listas estáticas
4	Introdução a listas	Listas dinâmicas
5	Listas estáticas	Listas dinâmicas
6	Listas dinâmicas	Listas estáticas
7	Listas estáticas	Programação com listas
8	Listas dinâmicas	Programação com listas
9	Introdução a pilhas	Aplicação de pilhas

A partir da visualização do gráfico, percebe-se que as taxas de feromônio decrescem após a passagem de cada estudante, mesmo que as notas de avaliação dos materiais tenham sido altas. Caso fossem geradas mais sequências de estudo,

essas taxas iriam se aproximar de zero rapidamente. Para resolver esse problema, duas alternativas poderiam ser implementadas: modificar a equação para incremento de feromônio nas arestas percorridas ou alterar a forma de evaporação do feromônio (por exemplo: decrementando o feromônio após a geração de toda a sequência e não da escolha de cada material).

Na próxima seção, são apresentadas as considerações finais deste trabalho.

5. CONSIDERAÇÕES FINAIS

Neste trabalho foi implementado o algoritmo AntStudy, responsável por gerar sequências de estudo para usuários com base a partir de um conjunto de materiais. O processamento para geração das sequências foi desenvolvido conforme a técnica do *Ant System* (AS), que se baseia na comunicação das formigas para criar soluções para problemas de alta complexidade.

Durante a criação de uma solução baseada no AS, deve-se fazer a modelagem do domínio e a abstração dos elementos fundamentais, como o ambiente, as formigas, o caminho e o feromônio. Além disso, é importante compreender e definir como ocorrerão os processos fundamentais propostos no algoritmo: inicialização, construção da solução e atualização das taxas de feromônio.

No AntStudy, o ambiente da natureza foi representado a partir de um conjunto de objetos de aprendizagem, descritos a partir de características propostas em um modelo de referência denominado LOM. Esse modelo proporciona a cobertura de uma grande quantidade de características e a possibilidade de se fazer um *Application Profile*, para utilizar somente os atributos necessários em cada contexto.

Uma das principais dificuldades encontradas para a composição do grafo a partir dos objetos de aprendizagem foi a representação dos relacionamentos existentes entre eles. Tal problema foi contornado a partir da utilização de um recurso matemático, denominado matriz de adjacência. A partir dessa matriz, foi possível representar tanto relacionamentos de precedência indicados no LOM (relacionamentos explícitos), como também fazer a inferência de relacionamentos entre esses objetos (relacionamentos implícitos).

Quanto ao usuário, considerou-se um conjunto limitado de características para fazer o cálculo da heurística com os atributos do LOM. Esse cálculo foi realizado a partir do coeficiente de similaridade do co-seno, adaptado da área de Recuperação de Informação.

Conforme a demonstração criada para ilustrar o funcionamento do algoritmo, foi possível visualizar a geração das sequências de estudo e as variações das taxas de feromônio em um grafo. Nessa fase do trabalho, percebeu-se a necessidade de modificar o incremento do feromônio nas arestas percorridas ou a forma de decremento de feromônio que acontece igualmente em todas as arestas.

Foi possível concluir que o AS pode ser aplicado a problemas que não são da área de otimização, conforme as aplicações convencionais da técnica. Isso foi possível a partir do entendimento dos processos propostos no algoritmo e posterior modelagem do contexto conforme as etapas do AS.

Como sugestão de trabalhos futuros, poderia ser desenvolvido um modelo de estudante mais elaborado, englobando um conjunto maior de características acerca do usuário. Além disso, propõe-se a criação de um conjunto de estereótipos, de forma manual ou automática (a partir da clusterização), para a definição de sequências de estudo para um grupo de pessoas com características semelhantes. Tal modelo poderia ser acompanhado de um modelo de diagnóstico, responsável por atualizar a localização dos usuários nos grupos, conforme eles adquirem mais conhecimentos e para agrupar os novos usuários que passarem a utilizar o sistema.

Outra ideia de trabalho futuro seria considerar um conjunto maior de características dos objetos de aprendizagem na definição das sequências de estudo. Para tal, deveriam ser acrescidas ainda novas informações para representar os usuários e implementadas modificações no cálculo da heurística que define a similaridade entre estudantes e materiais.

Além disso, pode-se modelar e implementar um banco de dados para armazenar os objetos de aprendizagem, usuários e informações do grafo (como feromônios e arestas percorridas), para possibilitar a utilização do algoritmo em um ambiente real. Sendo assim, os usuários poderiam acessar um sistema que utiliza o AntStudy para o sequenciamento de estudos em momentos distintos, sem que as informações do percurso já realizado sejam perdidas.

Na próxima seção são apresentadas as referências bibliográficas consultadas para o desenvolvimento deste trabalho.

6. REFERÊNCIAS BIBLIOGRÁFICAS

AUDINO, Daniel Fagundes, NASCIMENTO, Rosemy da Silva. Objetos de Aprendizagem : diálogos entre conceitos e uma nova proposição aplicada a educação. **Revista Contemporânea de Educação**. v.05, n.10, jul/dez. 2010. Disponível em: http://www.educacao.ufrj.br/artigos/n10/objetos_de_aprendizagem. pdf. Acesso em: 03 de set. de 2011.

BARBOSA, Ana Cristina Lima Santos. **Abordagens educacionais baseadas em dinâmicas colaborativas on-line**. 2008. 316 p. Tese (Doutorado em Educação) - Faculdade de Educação, Universidade de São Paulo, São Paulo.

BRUSILOVSKY, P. Adaptive hypermedia, an attempt to analyze and generalize. IN: P. Brusilovsky, P. Kommers, & N. Streitz (Eds.) **Multimedia, hypermedia and virtual reality: models, systems and applications**, Berlim, v. 1077, p. 288-304, 1996.

BRUSILOVSKY, P. SCHWARZ, E. User as student: towards an adaptive interface for advanced web-based applications. In: A. Jameson, C. Paris and C. Tasso (Eds.) **Proceedings of 6th International Conference on User Modeling**, Itália, p. 177-188, 1997.

BRUSILOVSY, Peter L. A framework for intelligent knowledge sequencing and task sequencing. In: Frasson, C., Gauthier, G. and McCalla, G. I. (Eds.). **Intelligent Tutoring Systems**. Berlim: Springer-Verlag, 1992. p. 499-506.

CORBETT, Albert T. KOEDINGER, Kenneth R .ANDERSON, John R. Handbook of human-computer interaction. In: HELANDER, Martin G. T. LANDAUER, Thomas K. PRABHU Prasad V. (Eds.). **Intelligent Tutoring Systems**. Amsterdam: Elsevier Science B. V., 1997. p. 849-874.

DENEUBOURG, J.-L.; ARON, S.; GOSS S.; PASTEELS J.-M. The self-organizing exploratory pattern of the Argentine ant. **Journal of Insect Behavior**, v. 3, p. 159-168, 1990.

DORIGO, Marco; BIRATTARI, Mauro; STÜTZLE, Thomas. Ant colony optimization. **IEEE Computacional Intelligence Magazine**, v. 1, n. 4, p. 28-39, 2006.

DORIGO, Marco; MANIEZZO Vittorio; COLORNI Alberto. Ant System: Optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics-Part B**, v. 26, p. 29-41, fev. 1996.

DORIGO, Marco; STÜTZLE, Thomas. **Ant colony optimization**. Massachusetts: The MIT Press, 2004. 305 p.

DOWBOR, Ladislau. **Tecnologias do conhecimento:** os desafios da educação. 2001. Disponível em: <http://dowbor.org/tecnconhec.asp>. Acesso em: 21 de set. de 2011.

FERREIRA, J. Tércio B.; ZARBIN, Paulo H.G. Amor ao primeiro odor: a comunicação química entre os insetos. **Química nova na escola**, n. 7, p. 3-6, maio 1998. Disponível em: http://qnesc.sbq.org.br/online/qnesc07/quimsoc.pdf. Acesso em: 18 de set. de 2011.

GOSS, S., ARON S., DENEUBOURGJ.-L., PASTEELS J.-M. Self-organized shortcuts in the Argentine ant. **Naturwissenschaften**, v. 76, p. 579–581, 1989.

HEERY, Rachel, PATEL, Manjula. **Application Profiles:** mixing and matching metadata schemas, Ariadne Issue 25, 2000. Disponível em: http://www.ariadne.ac.uk/issue25/app-profiles/. Acesso em 09 de set. de 2007.

HÖLLDOBLER, Bert; WILSON, Edward O. **The Ants**. Massachusetts: Harvard University Press, 1990. 739 p.

IEEE. Learning Technology Standarts Comittee (LTSC). **Draft Standart for Learning Object Metadata (IEEE - 1484.12.1)**. 2002. 44 p. Disponível em: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf. Acesso em: 30 de ago. de 2011.

KOCH, Nora Parcus. **Software engineering for adaptive hypermedia systems reference model: modeling techniques and development process**. 2001. 371 p. Tese (Doutorado em Ciências Naturais) - Fakultät für Mathematik und Informatik of Ludwig Maximilians University, Alemanha.

NWANA, Hyacinth S. Intelligent Tutoring Systems: and overview. **Artificial Intelligence Review**, Liverpool, n, 4, p. 251-277. 1990.

ONG, Jim. RAMACHANDRAN, Sowmya. **Intelligent Tutoring Systems**: the what and the how. Disponível em: http://www.astd.org/LC/2000/0200_ong.htm. Acesso em: 04 de set. de 2011.

RAGNEMALM, Eva L. Student Diagnosis in Practice: Bridging a Gap. **User Modelling and User-Adapted Interaction,** v. 5, n. 2, p. 93-116, 1996.

RICH, Elaine. User Modeling via Stereotypes. **Cognitive Science**, v. 3, p. 329-354, 1979.

SILVA, Edeilson et. al. SWEETS: um Sistema de Recomendação de Especialistas aplicado a uma plataforma de Gestão de Conhecimento. **Revista de Informática Teórica e Aplicada**, v. 18, n. 1, p. 83-111, 2011.

SPINELLI, Walter. **Os objetos virtuais de aprendizagem**: ação e criação do conhecimento. 2007. Disponível em: http://www.lapef.fe.usp.br/rived/textos complementares/textoImodulo5.pdf. Acesso em: 03 de set. de 2011.

VANLEHN, Kurt. Student Modeling. IN: POLSON, Martha C. RICHARDSON, J. Jeffrey (Eds.). **Foundations of Intelligent Tutoring Systems**. Nova Jersey: Lawrence Erlbaum Associates, 1988. p. 55-78.

VILAR, Bruno Siqueira Campos Mendonça. **Um framework para o desenvolvimento de sistemas adaptativos a partir de ontologias**. 2006. 146 p. Monografia (Graduação em Sistemas de Informação) - Centro Universitário Luterano de Palmas, Palmas - TO.

WILEY, David A. **Learning Object Design and Sequencing Theory**. 2000. 131 p. Tese (Doutorado em Filosofia) - Brigham Young University, Provo.