ANDRÉIA OLIVEIRA LIZARDO

FERRAMENTAS DE APOIO AO APRENDIZADO DE ALGORITMOS DE BUSCA HEURÍSTICA POR MEIO DA VISUALIZAÇÃO DAS ÁRVORES DE MEMÓRIA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Alexandre Ibrahim Direne

CURITIBA

2007

ANDRÉIA OLIVEIRA LIZARDO

FERRAMENTAS DE APOIO AO APRENDIZADO DE ALGORITMOS DE BUSCA HEURÍSTICA POR MEIO DA VISUALIZAÇÃO DAS ÁRVORES DE MEMÓRIA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Alexandre Ibrahim Direne

CURITIBA

2007

ANDRÉIA OLIVEIRA LIZARDO

FERRAMENTAS DE APOIO AO APRENDIZADO DE ALGORITMOS DE BUSCA HEURÍSTICA POR MEIO DA VISUALIZAÇÃO DAS ÁRVORES DE MEMÓRIA

Dissertação aprovada como requisito parcial à obtenção do grau de Mestre no Programa de Pós-Graduação em Informática da Universidade Federal do Paraná, pela Comisso formada pelos professores:

Orientador: Prof. Dr. Alexandre Ibrahim Direne
Departamento de Informática, UFPR

Prof. Dr. Andrew Ricardo Pimentel
SPEI

Prof. Dr. Fabiano Silva
Departamento de Informática, UFPR

Curitiba, 30 de agosto de 2007

AGRADECIMENTOS

Agradeço a Deus, por sempre me acompanhar e me abençoar durante a minha vida com Saúde e Paz para o desenvolvimento deste trabalho.

Ao Professor, Dr. Alexandre Ibrahim Direne, que além de um execelente orientador é um grande amigo, pela motivação e confiança com que conduziu este trabalho e pelo grande envolvimento e dedicação durante suas aulas e orientações (sempre me será um exemplo na arte de ensinar).

Aos meus pais, Elizio e Iraci, pelo apoio e incentivo durante toda a minha vida.

Ao meu irmão, Adriano, amigo e querido em todas as horas.

Ao meu noivo, Roberto, pela compreensão e carinho.

À minha família como um todo, pela confiança transmitida e principalmente pelas orações mesmo estando distante.

Ao Departamento de Informática da Universidade Federal do Paraná, pelo profissionalismo e comprometimento com suas atividades.

Aos amigos que compartilharam desta experiência, e que colaboraram de forma direta ou indireta para a realização deste trabalho.

Dedico este trabalho a minha família, pois sem o apoio que me deram nada disso seria possivel.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Este trabalho aborda conceitos e ferramentas que auxiliam o aprendizado de algoritmos de busca heurística por meio da visualização de árvores de memória. A resenha literária ressalta a lacuna deixada por pesquisas passadas ao não enfatizarem os aspectos de expansão da memória da máquina para apoiar o aprendizado do comportamento de um algoritmo complexo. O artigo mostra exemplos de como a evolução da representação gráfica das estruturas de árvores pode contribuir para o melhor entendimento do comportamento de um algoritmo de busca heurística típico da área de Inteligência Artificial. A noção de dificuldade do algoritmo é definida em relação às múltiplas representações externas necessárias para abranger todas as suas características específicas (profundidade, grau de reavaliação, duplicação de nodos, etc). A solução apresentada no artigo foi implementada no protótipo de software VIMAP, o qual se baseia em estudos cognitivos da literatura passada, assim como em princípios pedagógicos referentes ao aprendizado por visualização e investigação. Uma breve discussão é delineada no final do texto, seguida de perspectivas futuras de pesquisa.

Palavras-chave: Ensino de algoritmos, micromundos de aprendizagem, múltiplas representações externas.

ABSTRACT

This work approaches concepts and software tools to assist the learner of heuristic search algorithms through the visualisation of memory trees. A literature survey highlights the lack of past research in the emphasis on computer memory expansion aspects as a support for learning a complex algorithm's behaviour. The article shows examples of how the evolution of graphical representations of tree structures can contribute to a better understanding of the behaviour of a typical heuristic search algorithm in the area of Artificial Intelligence. The notion of difficulty of the algorithm is defined in terms of the multiple external representations needed for embracing all its specific features (depth, processing overhead, node duplication, etc) The solution portrayed in the article has been implemented in the VIMAP prototype software tool, which is based on cognitive studies of the published literature as well as on pedagogic principles linked to visualisation and exploratory learning issues. A brief discussion closes the the text along with future research directions.

Word-key: Teaching algorithms, microworlds, visual representation.

CAPÍTULO 1

INTRODUÇÃO

1.1 O problema central

Na disciplina de Inteligência Artificial (I.A.) no curso de Ciência da Computação da UFPR são ensinados aos aprendizes (alunos) algoritmos de busca heurística. Esses algoritmos são utilizados para resolver problemas complexos, que não podem ser tratados de maneira eficiente por meio de algoritmos tradicionais. Os algoritmos de busca heurística diferem dos algoritmos de busca tradicionais por possuírem informações do domínio.

Por meio das informações do domínio são implementadas as funções heurísticas, que são critérios ou métodos computacionais para decidir o caminho mais eficiente entre várias alternativas de ação. As funções heurísticas buscam encontrar um determinado objetivo sem precisar testar todas as opções, enquanto um método de busca tradicional testa todas as opções gerando uma explosão combinatória das mesmas.

Adicionalmente, é de grande importância que sejam implementadas heurísticas que forneçam soluções de boa qualidade (próximas da solução ótima) em um tempo computacional razoável, melhorando a eficiência do processo de busca. Visto que nos algoritmos tradicionais existe uma perda por causa da exploração de todas as alternativas (completude).

Devido à complexidade desses algoritmos há uma certa dificuldade dos aprendizes em compreender as mudanças de estado que ocorrem no espaço de busca durante a execução do algoritmo, como este encontra o estado ou caminho solução do problema. É importante salientar que o aprendiz desenvolve essa perícia, adquirindo assim capacidade para resolver problemas complexos com alto nível de abstração.

No desenvolvimento de perícias, o aprendizado por visualização e o aprendizado por investigação auxiliam o aprendiz em adquiri-las [4]. O aprendizado por visualização pode garantir um grau de abstração ao utilizar técnicas de representações visuais, mesmo sendo

difícil definir quais as representações visuais mais adequadas para representar o problema. Enquanto o aprendizado por investigação permite ao aprendiz interagir de acordo com a sua necessidade.

Apesar dos avanços na área de aprendizagem por visualização de informação, pode-se verificar nos sistemas tutores inteligentes (STI) que o nível de interatividade e visualização ainda são bastante primitivos. Apenas realizam o experimento e exibem o resultado final não permitindo ao aprendiz acompanhar a seqüência de passos [21].

De acordo com a pesquisa realizada e trabalhos correlatos, não foi encontrado nenhum projeto para o ensino de algoritmos de busca heurística o qual utiliza o aprendizado por meio da visualização de elementos de memória como método de abstração de conhecimento, possibilitando a aprendizagem por investigação por meio de ações de interação disponíveis ao aprendiz. Deste modo, este estudo efetiva-se como uma importante contribuição na área de ferramentas de software voltada à educação.

1.2 Objetivos gerais

De acordo com o problema em foco, este estudo tem o objetivo de criar ferramentas de apoio à visualização da alocação gradual de memória para o desenvolvimento de perícia em resolução de problemas solucionados por algoritmos de busca heurística da IA. A técnica de visualização a ser utilizada será a visualização da árvore de memória permitindo ao aprendiz interagir com a interface gráfica de exposição (e reconstrução) da hierarquia de estados de busca. Para atingir o objetivo proposto foram definidos os seguintes objetivos específicos:

- Definição dos elementos de representação visual;
- Definição das ações disponíveis ao aprendiz para este interagir com a ferramenta;
- Implementação do protótipo em linguagem Java;
- Adequação dos conceitos pedagógicos criados.

1.3 O contexto do projeto

Com os avanços tecnológicos na área de informática surgiu a necessidade de estudos referentes ao uso da informática na educação. Tem-se, então, a difusão da informática na educação como uma das principais modalidades de apoio ao ensino e à aprendizagem. Dentre os vários seguimentos de aplicação da informática como apoio ao processo de ensino-aprendizagem, o campo de sistemas tutores inteligentes oferece vantagens de aplicações que se distinguem das aplicações convencionais pelo fato de que este campo possibilita a adaptação individualizada do software a um aprendiz. Isto terá impactos cada vez mais positivos na sociedade onde a educação constitui uma demanda crescente para software adaptativo.

O projeto se encontra no contexto de ampliar os conceitos e ferramentas de apoio ao aprendizado, para que se desenvolva o conhecimento pericial por meio de representações visuais e interações. Direne [3]descreve como conceitos cognitivos e computacionais podem ser aplicados no desenvolvimento da interface e no modelo aprendiz em interações tutoriais.

Para o desenvolvimento de perícias em diagnósticos na área de Radiologia, foi projetado o ambiente RUI, o qual é formado por um conjunto de ferramentas de ensino por meio de imagens de raio-X. Na Figura 1.1 é apresentada a interface da ferramenta de ensino do RUI a qual utiliza vários elementos visuais para destacar anomalias nas imagens [2],[3].

Já o trabalho de Pimentel [8] em adaptação ao aprendiz visa possibilitar a escolha adequada da próxima imagem a ser trabalhada com o aprendiz, apresentando um conjunto de medidas para descrever cognitivamente a base de imagens radiológicas do sistema RUI [2].

No ensino de programação de computadores, Santos [23] definiu ferramentas e linguagem de apoio ao ensino de linguagem de programação. Foi implementada uma arquitetura de um sistema de autoria e um interpretador tutorial, com o foco voltado às limitações das ferramentas existentes que restringem demais a criatividade do aluno ou falham em prover um *feedback* com alto valor cognitivo agregado. No apoio à programação

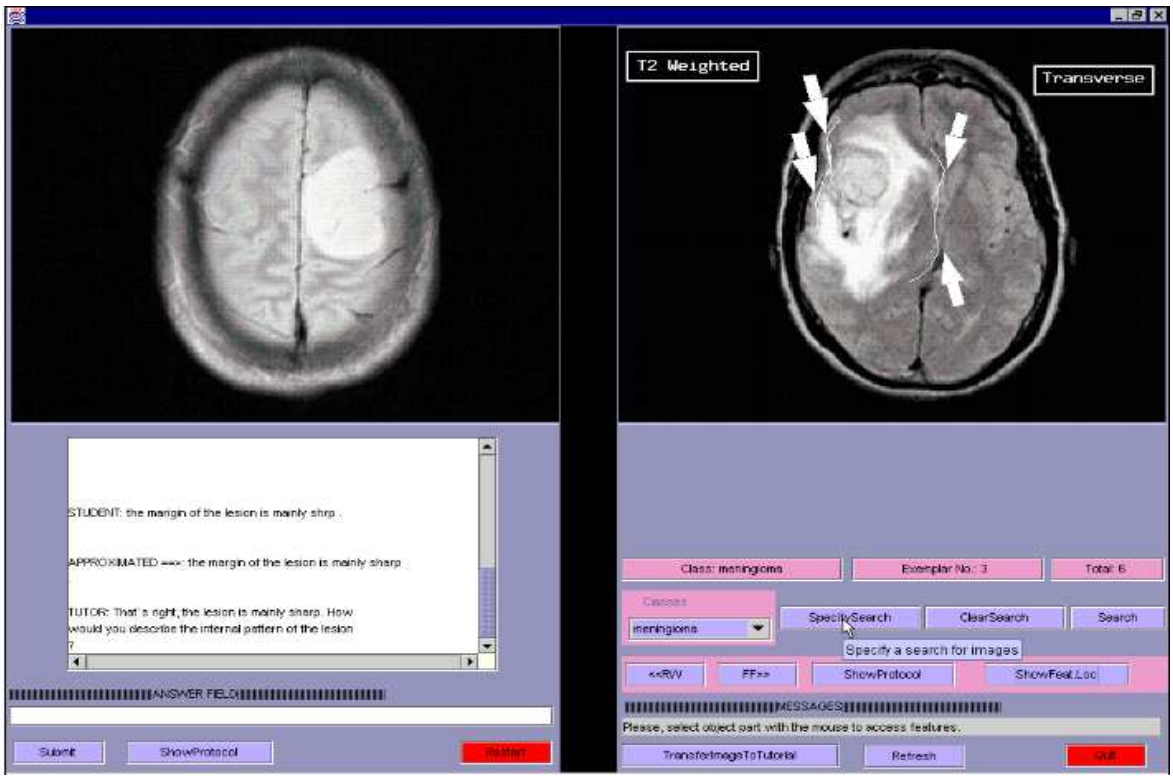


Figura 1.1: Ferramenta de ensino do ambiente RUI

de dispositivos digitais com mecanismos de interpretação independentes de linguagem foram definidas representações e ferramentas por Nascimento [20].

Portanto este trabalho assim como os outros trabalhos citados acima, os quais foram desenvolvidos pelo grupo de pesquisa da UFPR atuam diretamente no desenvolvimento de perícias dos aprendizes com o uso de ferramentas e linguagens de aprendizagem tanto para as ferramentas de ensino como autoria.

Após esta introdução a qual fez uma descrição geral do projeto, especificando o problema central, os objetivos a serem alcançados e o contexto da aplicação na qual o projeto se encontra, no capítulo 2, são apresentados alguns trabalhos relacionados. No capítulo 3 é feita uma descrição em detalhe dos conceitos fundamentais nos quais foram embasados a solução do problema. O capítulo 4 apresenta o protótipo implementado. capítulo 5 uma discussão dos resultados esperados com o desenvolvimento deste trabalho e, por fim, o capítulo 6 refere-se à conclusão e apresentação de idéias sobre possíveis extensões futuras.

CAPÍTULO 2

TRABALHOS RELACIONADOS

Atualmente a utilização do computador com abordagens da I.A. no ensino é uma forma de diversificar os modos de apoio ao aprendizado e sua adaptação às experiências requisitadas pelos estudantes.

2.1 Autoria de conteúdos inteligentes

A partir da aplicação de I.A. às ferramentas de software educacional no início dos anos 80 foi possível construir tutores mais poderosos e flexíveis que o tradicional, o qual é definido pelos quatro módulos (Tutor, Aprendiz, Pedagógico e Interface). Um exemplo disso foi o sistema PIXIE [18], no qual registros mais detalhados sobre os aprendizes foram construídos para tentar maior capacidade de adaptação por parte da máquina. Os sistemas tutores inteligentes (STIs) passaram a oferecer as perspectivas de grandes vantagens sobre os ambientes puramente passivos de aprendizagem. Porém, sua construção tinha um alto custo e se tornava cada vez mais difícil de ser generalizada por incluir diversos aspectos interdisciplinares, principalmente os de caráter pedagógico.

Durante a década de 1990 ocorreu um progresso no campo de ferramentas de autoria para STIs. Murray [50] descreveu os tipos de STI construídos com ferramentas de autoria, detalhando a interface, a representação de conhecimento e as técnicas de aquisição de conhecimento que foram usadas para permitir que não programadores construam STIs por meio de ferramentas de autoria. Ele detalha categorias e métodos usados por sistemas de autoria para simplificar e automatizar o processo de aquisição de conhecimento de conteúdos eletrônicos. As principais metas de tais ferramentas são:

- Diminuir o esforço por fazer os tutores inteligentes;
- Diminuir as habilidades técnicas necessárias para construir os tutores inteligentes;

- Ajudar o autor a definir e organizar o domínio do conhecimento pedagógico;
- Apoiar aos princípios de *design*;
- Acelerar a disponibilidade de protótipos de *designs* de tutores inteligentes.

Para atender a essas metas foram definidos vários métodos ou características. A maioria dos métodos atinge mais de uma meta (por exemplo, uma característica que auxilia o autor a definir uma estratégia pedagógica também diminuirá o esforço para construir um STI). Abaixo são listados os métodos:

- Articulação de conhecimento;
- Inserção de conhecimento específico e conhecimento padrão;
- Administração de conhecimento;
- Visualização de conhecimento;
- Definição de conhecimento e administração do fluxo de trabalho;
- Desenvolvimento e validação de conhecimento sobre o *design*;
- Reuso de conhecimento;
- Criação de conhecimento automatizado.

De acordo com Murray [50] o processo de autoria por meio da aquisição do conhecimento fornecido pelo autor oferece facilidade e flexibilidade. Todavia, muito pouco foi feito na prática a respeito da necessidade de garantir a consistência do conhecimento que está relacionada ao que foi chamado de "validação de conhecimento sobre o design". Mesmo o uso de um formalismo de representação do conhecimento existente não é suficiente para auxiliar um autor humano a criar a interface de um sistema complexo, como o de autoria de um STI. O processo de autoria é uma tarefa demorada e a qualidade da representação obtida depende da experiência do perito, de aspectos específicos do domínio e do *design* da interface.

A necessidade em se construir ferramentas de autoria que visam a redução dos custos, assim como a produção automatizada de material de curso para um maior número de autores, resultou em grandes feitos. Durante os anos 90, e mesmo depois disso, alguns exemplos de sistemas de autoria e "shells"[1] para STI foram bem sucedidos em suas idéias. O SIMQUEST [9] foi projetado para ser um simulador genérico com uma ferramenta de Autoria a ele acoplada. O REDEEM [33], o EON [49], o Demonstr8 [47] e o COCA [32] foram criados para enfatizar a autoria e a interpretação dos diferentes modelos ou módulos internos de um STI: domínio, aprendiz, pedagógico e interface conforme é apresentado na figura 2.1.

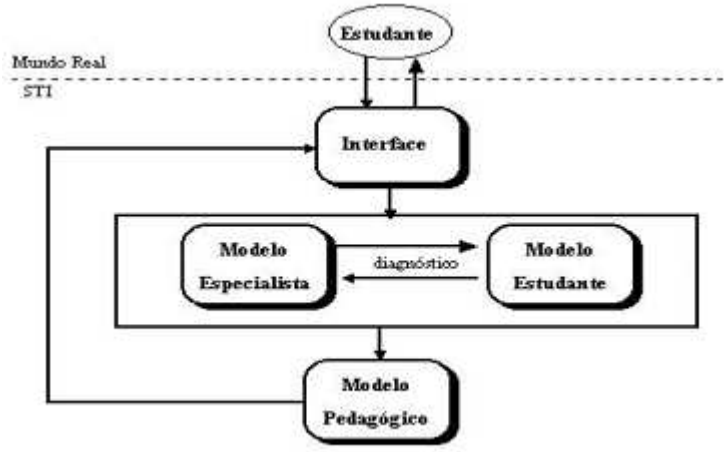


Figura 2.1: Estrutura básica dos modelos de um STI

2.2 Consistência de ensino em sistemas tutores inteligentes

Os sistemas tutores inteligentes são programas de computador utilizados para o ensino. Tais programas valem-se de técnicas da IA para saberem o que/como/a quem ensinar [24]. A maioria dos STIs segue o padrão de separar, de maneira modular, a representação do conhecimento específico de domínio, a estratégia de ensino[2], o modelo do aprendiz e o desenvolvimento da interface com o usuário. Tais STIs podem incluir também catálogos de erros ou outros módulos fundamentais, dependendo tanto da ênfase de suas arquiteturas quanto das suas potencialidades dinâmicas.

[1] Shell é uma ferramenta de ensino, que não possuir uma interface definida

[2] A teoria pedagógica adotada

Historicamente, o STI surge como um avanço em relação às potencialidades apresentadas pelo sistema CAI (Computed Aided Instruction). Um dos aspectos desse avanço é que o STI utiliza uma base de conhecimento enquanto o CAI trabalha com bases de dados convencionais. A utilização de bases de conhecimentos propicia ao STI simular o processo do raciocínio humano dentro de um determinado domínio, podendo assim oferecer maior garantia na oferta de um ensino com consistência do seu conteúdo. A utilização de bases de conhecimento permite, além de tratar de fatores de consistência, auxiliar em estratégias nas soluções de problemas ou nas tomadas de decisões. Todavia, Sharples[19] ressalta que alguns projetos da área de IA possuem técnicas poderosas em representação de conhecimento, porém são inúteis na prática educacional por não possuírem representações internas sobre o conhecimento pedagógico.

Anderson[27], em sua pesquisa sobre o ACT (Adaptive Control of Thought) uma teoria de aprendizado baseada nos processos de memória, contribuiu com uma teoria geral de cognição com ênfase sobre aquisição de conhecimento. Nesta teoria, temos esclarecida a relação entre psicologia cognitiva e STI: as características estruturais da psicologia cognitiva (simular e compreender a cognição humana, compreender como as pessoas organizam o conhecimento e como produzem o comportamento inteligente) são as ferramentas que possibilitam inserir parâmetros que facilitam a interface do STI com o aprendiz. No modelo cognitivo, as funções cognitivas podem ser representadas por meio de regras de produção. Os mecanismos de modelo de aprendizado incluem a idéia de que o conhecimento é, inicialmente, adquirido de modo declarativo por meio de instruções. Após esse primeiro momento de aprendizagem, a simulação de problemas e a sua solução fazem com que as instruções sejam convertidas e reorganizadas em procedimentos por meio da experiência. Tal aprendizado é chamado de compilação do conhecimento.

O ACT, comparado com outros modelos de aprendizado por exemplo: STEP (VANLEHN, 1998) e GRAPES (SAUERS e FARRELL, 1982), tem maior sucesso no aspecto de aquisição de habilidades humanas em vários domínios, incluindo linguagem natural. Ele permite ao aprendiz o desenvolvimento de habilidades em realizar tarefas ligadas à perícia por meio de um processo automatizado, que não requer atenção e nem proces-

samento consciente no longo prazo. A teoria ACT trata o aprendiz como um objeto errante e mantém um processo sempre rápido de retroalimentação sobre as ações erradas do aprendiz. Tal procedimento permite o acompanhamento, por parte da máquina, de tarefas mais complexas de solução de problemas, pois a máquina conta com fragmentos do conhecimento procedimentalizado do ser humano como padrão comparativo [27].

A parte mais sensível no desenvolvimento de um STI é, certamente, a aquisição de conhecimento, pois esta não pode limitar-se à adição de novos elementos à base de conhecimento; é necessário integrar o novo conhecimento ao conhecimento já disponível, por meio da definição de relações entre os elementos que constituem o novo conhecimento e os elementos já armazenados na base. Outro aspecto importante é o tratamento de incoerências. Dependendo da forma como o novo conhecimento é adquirido, pode haver erros de aquisição. Estes erros podem resultar tanto da própria natureza do conhecimento (como em dados obtidos por meio de sensores sujeitos a ruído) quanto podem ser gerados pela interface humana existente entre o mundo real e o sistema de representação.

Técnicas foram desenvolvidas para evitar erros de aquisição, como, por exemplo, a especificação de regras de aquisição em que o tipo de conhecimento esperado é definido. Estas técnicas são comuns aos sistemas de representação de conhecimento e aos sistemas de gerenciamento de bancos de dados. Contudo, uma base de conhecimento pode ser também examinada periodicamente com a finalidade de detectar incoerências eventualmente introduzidas no processo de aquisição. Este segundo método, o de exame periódico, é limitado pelo fato de que linguagens de representação razoavelmente expressivas não contam com procedimentos completos de verificação conhecidos. Deve-se observar que a adequação do formalismo de representação ao tipo de conhecimento do mundo real a ser representado é fundamental para a eficiência do processo de aquisição. O conhecimento prévio é importante, por exemplo, para trabalhar com um tutor de produto cartesiano pode ocorrer falha por falta de conhecimento dos tipos de conhecimento especialista de algum componente ou até mesmo de conhecimento de conceitos básicos.

2.3 Ensino de conceitos visuais

O processo de aquisição de conhecimento especializado para o ser humano é um processo lento. Pode-se levar vários anos para adquirir perícia em um determinado domínio. Com o objetivo de diminuir esse tempo são utilizados recursos computacionais, como, por exemplo, o reconhecimento visual.

Lesgold[7] apresentou importante contribuição para a área de reconhecimento visual especializado. Em seu trabalho, descreve aspectos muito particulares sobre a aquisição de perícia: focalizou atividades que necessitam de muito tempo de treinamento para se adquirir. O trabalho revela a necessidade de aquisição de conhecimento dos princípios e das práticas no domínio específico de conceitos visuais que se quer estudar.

Lesgold também aponta algumas habilidades que um radiologista (Médico) deve possuir em diferentes fases de sua evolução profissional, cobrindo desde a situação em que se classifica como iniciante até aquela em que é considerado especialista. Tal classificação, segundo consta no relato das pesquisas, afeta os detalhes do treinamento. Cada etapa da aquisição do conhecimento acontece de maneira diferente: a aquisição propriamente dita e o comportamento dos aprendizes. Apesar de Lesgold não ter construído sistemas tutores inteligentes para o ensino de conceitos visuais até 1992, seus trabalhos serviram (e ainda servem) como uma sólida base teórica para permitir a concepção mais completa e adequada desses sistemas.

Posteriormente, Lesgold[22] descreveu as habilidades para diagnosticar imagens de raio-X. No estudo, foram considerados os diagnósticos produzidos por médicos iniciantes até especialistas experientes sendo feita a análise das diferenças dos resultados obtidos. Foi possível explicar como especialistas e iniciantes fazem os diagnósticos de imagens por meio da organização de vários princípios: fisiologia, anatomia, teorias médicas e geometria projetiva de radiografia. A perícia é um conhecimento que o especialista possui, mas nem ele próprio consegue definir de forma organizada como realizar um diagnóstico preciso.

Sharples[29] define um conceito visual como uma construção mental associada a um conjunto de imagens. Ele também formaliza alguns princípios importantes no ensino de conceitos visuais auxiliados por computadores, destacando características da perícia em

diferentes níveis de competência, do iniciante até o perito. Neste estágio do desenvolvimento da sua pesquisa, Sharples preocupou-se principalmente com aspectos de ordem das instruções e casos radiológicos como forma central de estruturar o ensino ou a aprendizagem.

Depois de vários anos de pesquisa, Sharples [19] definiu uma metodologia para o desenvolvimento de um sistema tutor que considera também os aspectos sócio-cognitivos na aquisição de perícia. Com isso, os elementos pedagógicos e psicológicos passaram a contar com definições mais formais, as quais foram aplicadas em um sistema de treinamento em neuroradiologia: o MR-Tutor.

Para o ensino de conceitos visuais de uma base de imagens radiológicas, Pimentel [8] apresenta um conjunto de medidas cognitivas visando possibilitar uma escolha adequada da próxima imagem a ser trabalhada com o aprendiz. Segundo Pimentel, o método proporciona facilidade ao aprendizado por reduzir a complexidade relativa da solução dos problemas concretamente representados por meio de duas imagens consecutivas na ordem de cálculo das referidas medidas cognitivas. A redução das dificuldades é importante, pois a demanda cognitiva para classificar imagens, identificar características e descrever anormalidades são parte importante do treinamento de especialistas em conceitos visuais, tais como os radiologistas.

Encontrar e definir medidas de relevância cognitiva para ordenar uma base de exemplos permite que o modelo pedagógico de um STI modifique a ordem de apresentação das imagens. Estas medidas servem para quantificar o potencial que a imagem tem em exercitar o aprendiz em uma determinada capacidade que ele deve desenvolver para tornar-se perito. Tais medidas também têm o objetivo de medir e representar computacionalmente a carga cognitiva de imagens e tentar reduzir os componentes subjetivos envolvidos na ordenação de imagens. Isso possibilita atingir algum grau de individualização do ensino, ou seja, aplicar estratégias pedagógicas, de longo e curto prazo, que mais se adaptem ao aprendiz e realizar mudanças para corrigir falhas de formação da habilidade diagnóstica.

2.4 Visualização computacional

A área de visualização computacional é muito ampla e envolve o desenvolvimento e aplicação de técnicas gráficas para apresentação e entendimento dos mais variados conjuntos de dados. Os tipos de representações variam desde diagramas explicativos e gráficos de barra até dados meteorológicos, médicos, e científicos.

Por ser uma área complexa, ela está subdividida em várias áreas, por exemplo: visualização científica, visualização de dados, visualização de informação. Porém não está definido o limite exato entre uma e outra, pois o uso predominante de visualização em uma área não quer dizer que não possa utilizar conceitos ou recursos que envolva uma outra área.

Pode-se dizer que a visualização no geral é a representação no formato gráfico de dados e informações, com o intuito de facilitar a interpretação dos mesmos[39].

2.4.1 Visualização de informação

Visualização de informações é uma área de aplicação de técnicas de computação gráfica, geralmente interativas, visando auxiliar o processo de análise e compreensão de um conjunto de dados, através de representações gráficas manipuláveis[1].

As técnicas de visualização de informações procuram representar gráficamente dados de um determinado domínio de aplicação de modo que a representação visual gerada explore a capacidade de percepção do aprendiz possa interpretar e compreender as informações apresentadas e deduzindo novos conhecimentos[1].

Shneiderman [10] classificou as técnicas de visualização, por tipo de dados e por tarefas, do seguinte modo:

- Unidimensionais (1D);
- Bidimensionais (2D);
- Tridimensionais (3D);
- Multidimensionais (nD)

- Temporais;
- Dirigidas à visualização de hierarquias(árvores);
- Dirigidas à visualização de relacionamentos (grafos)

De acordo com o trabalho de Carmo [31] foi observado que nesta classificação se detectam quatro linhas básicas: dimensão do espaço de trabalho (1D, 2D e 3D), evolução ao longo do tempo (dados temporais), tratamento de bases de dados relacionais (dados multidimensionais) e estruturas complexas (árvores e redes).

As várias técnicas de visualização podem ser aplicadas a um problema de maneira combinada, permitindo mecanismos de interação que possibilitam ao usuário manipular essa representação da melhor maneira de compreender o conjunto de dados representado.

O nível de abstração dessa representação é mais alto, porque freqüentemente não há relação direta entre os dados e uma entidade física ou geométrica, ou o usuário não está interessado em dados brutos, mas em observar características ou padrões no conjunto de dados.

Devido a existência das várias técnicas, a construção de sistemas para visualização de informações pode ser muito complexa devido as seguintes caracteristicas [1]:

- Necessidade de criação de uma metáfora visual que permita codificar visualmente o conjunto de informações com o grau de fidelidade necessário à aplicação;
- Mecanismos de interação necessários para manipular os freqüentemente volumosos e complexos conjuntos de dados;
- Necessidade de implementar algoritmos geométricos complexos tanto para a criação da representação visual como para sua manipulação.

Um dos problemas da visualização de informação é a definição das representações gráficas adequadas a uma determinada informação, isto é, associar representações visuais a uma localização no espaço a cada elemento de informação. Neste processo os atributos

da informação são muitas vezes ligados a atributos gráficos, alteração das cores, geometria do objeto, ou dimensão [31].

Os atributos gráficos interferem diretamente nas características do projeto como interfaces gráficas, usabilidade e adequação das representações visuais utilizadas.

2.4.2 Visualização científica

A visualização científica é uma área de pesquisa que estuda estratégias e algoritmos para mapear informações científicas(em geral, numéricas) em representações gráficas. Com isso, possibilita uma compreensão do conteúdo de grandes conjuntos de dados e dos fenômenos que geram esses dados de maneira fácil e legível.

Uma das definições de visualização científica é entendida como uma forma de comunicação que transcede as aplicações e os limites tecnológicos [1]. O termo foi usado para sensibilizar a Fundação Nacional de Ciência para a importância do uso de métodos de computação gráfica associado às simulações com supercomputadores.

Pode-se observar que a visualização científica é bem semelhante a visualização de informação, porem é mais específica porque refere-se a dados científicos.

2.4.3 Visualização da execução de programas

Esta área é antiga, um caso especial da visualização científica entretanto está diretamente ligada com este trabalho pois tem o foco especial da visualização de programas no momento da execução e a sua importância no aprendizado.

A utilização de visualizações como instrumento de apoio ao aprendizado ja está bastante difundida. Pode-se citar alguns projetos que possuem uma certa similaridade com este trabalho.

Juell [34] foi coordenador do projeto de Programa Visual para construir visualizações para ajudar os estudantes a entender programas e técnicas de I.A. na solução de problemas.

O projeto foi construido para web e utilizou imagens 3D conforme a figura 2.2 onde são apresentadas algumas imagens do projeto. O conteúdo das imagens direciona as visualizações permitindo ao estudante explorar as informações. A tarefa central destas

visualizações é apresentar uma descrição do processo ou parte dele, talvez a parte mais complexa do processo.

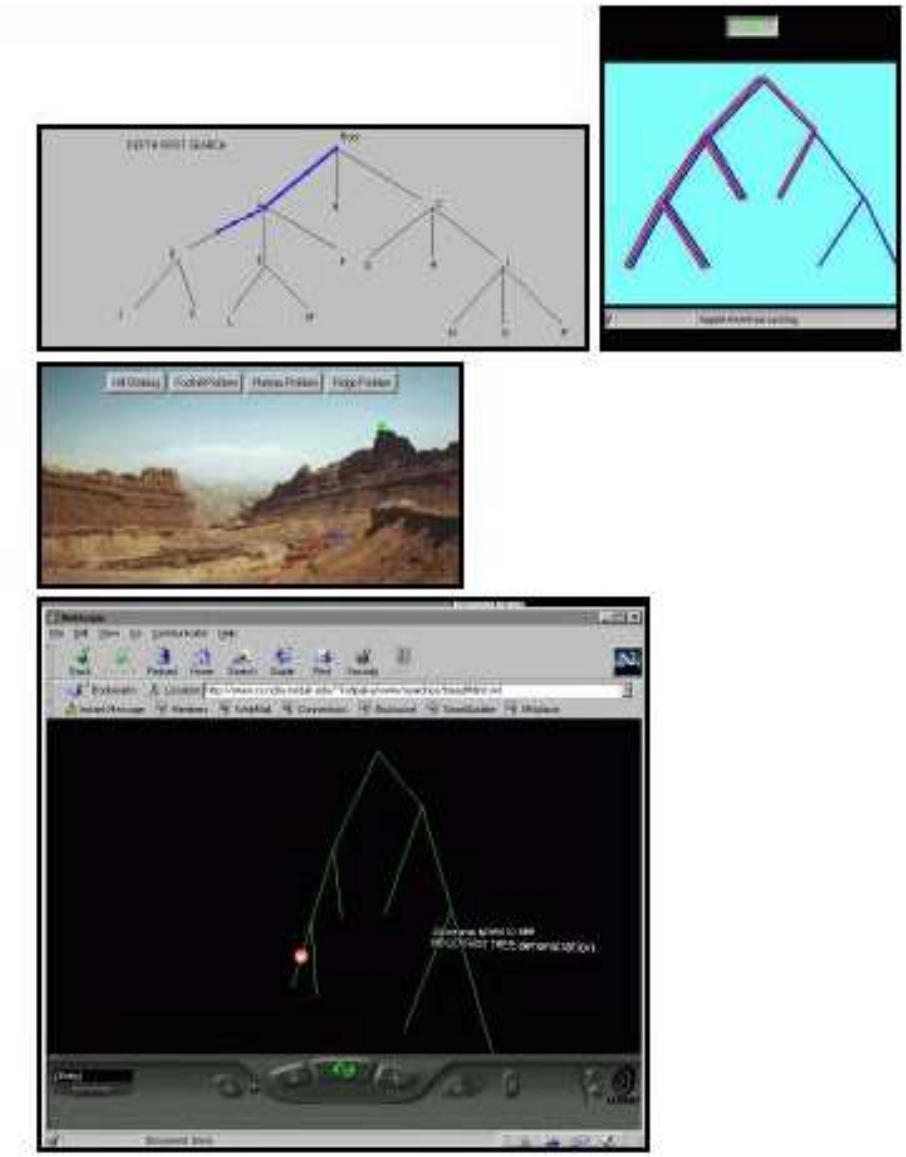


Figura 2.2: Projeto programa visual

O projeto utiliza variás técnicas de visualização porém o foco principal está relacionado ao fator tempo(temporais) e para isto foi utilizado no projeto três técnicas para mostrar a passagem do tempo: filmes, falsa coloração e tempo representando como uma dimensão de espaço.

Na tentativa de efetivar e quantificar o grau de contribuição da visualização no processo de ensino/aprendizagem Juell em seu artigo [4] descreve as avaliações realizadas que indicaram melhor resultado de aprendizado nos estudantes que utilizaram o processo de

visualização. Inclui assuntos como sistemas mais interativos e uma gama mais larga de informação.

Uma das pesquisas de Juell [4] está baseada em uma sucessão de três experiências com três visualizações diferentes, com a disciplina de introdução à informática. Os instrutores que participaram no estudo usaram as visualizações em algumas turmas e em outras não. As visualizações mostram o conceito de um objeto, interações dentro de uma classe e recursão. Foi medido o tempo que cada visualização foi acessada por um estudante e a contagem do estudante em perguntas de teste relacionadas ao conceito manifestado pela visualização.

A ferramenta mais interativa produz resultados estatísticos significantes para os grupos tratados e diferentes para grupos sem tratar. As outras experiências mostraram alguma melhoria para partes dos grupos tratados, mas não a um nível estatístico significante.

Em outro trabalho de pesquisa[14] do mesmo grupo descreveu-se os esforços na construção de visualizações de conhecimento voltado para a aprendizagem baseada em problemas. Foram identificados problemas que estudantes têm e as suas difilculdades em aprender programação orientada a objeto. Para resolver o problema colocaram os estudantes dentro da sala de aula e foram utilizadas visualizações existentes criadas por estudantes de classes anteriores, então os estudantes melhoraram visualizações existentes, ou criaram novas visualizações para uso futuro em outras turmas.

O trabalho está baseado no problema de aprendizagem, o foco principal está voltado ao processo de construir conhecimento, nos detalhes das visualizações, observações e os méritos desta aproximação. Devido aos alunos construirem as suas representações mentais.

2.5 Múltiplas representações

Uma das razões para explorar múltiplas representações em ambientes de apredizagem é tirar proveito das representações visuais como papéis complementares ou que diferenciam as informações que estão dentro de cada representação contribuindo como apoio ao processo de aprendizagem [42].

É possível que uma única representação possa prover todas as informações necessárias para apoiar a conclusão exigida do aprendiz, porém, a representação seria extremamente complexa para ser interpretada.

Foram realizadas pesquisas [43] as quais definiram o desenvolvimento de múltiplas representações, cuja função é direcionar os ambientes de aprendizagem para que as múltiplas representações apoiem realmente a aprendizagem e as tarefas cognitivas que devem ser empreendidas por um estudante ao interagir com as múltiplas representações. Quando os estudantes podem interagir com uma representação apropriada seu desempenho é maior.

Se são apresentados aos estudantes uma opção das representações que podem ser selecionadas para atender melhor as suas necessidades, é uma evidência que pode melhorar a aprendizagem [17] apud [46]. Isto pode ser feito para diferir perícias como forma alternativa de representação por causa de diferenças individuais mais estáveis [12] apud [46].

Com a utilização de múltiplas representações era esperado que os estudantes fossem ser beneficiados das propriedades de cada uma das representações e que isto conduziria a um entendimento mais aprofundado do assunto ensinado. Porém, pesquisas que avaliaram a efetividade de ambientes multirepresentacionais de apoio ao aprendizado produziram resultados mistos. Vários estudos [37],[30] apud [46] mostraram que os estudantes acham o funcionamento de ambientes com múltiplas representações difíceis de interagir e visualizar.

De acordo com outras pesquisas existem evidências abundantes dos papéis importantes nos quais representações externas apoiam a aprendizagem [5],[11] apud [30]. Na figura 2.3 do ambiente ambiente de aprendizagem DEMIST pode-se observar um conjunto multi representacional.

As descrições de representações foram fundadas na análise de [45] apud [41] onde é proposto que qualquer representação particular deveria ser descrita em termos de:

- O mundo representado;
- O mundo representante;
- Os aspectos do mundo representado;

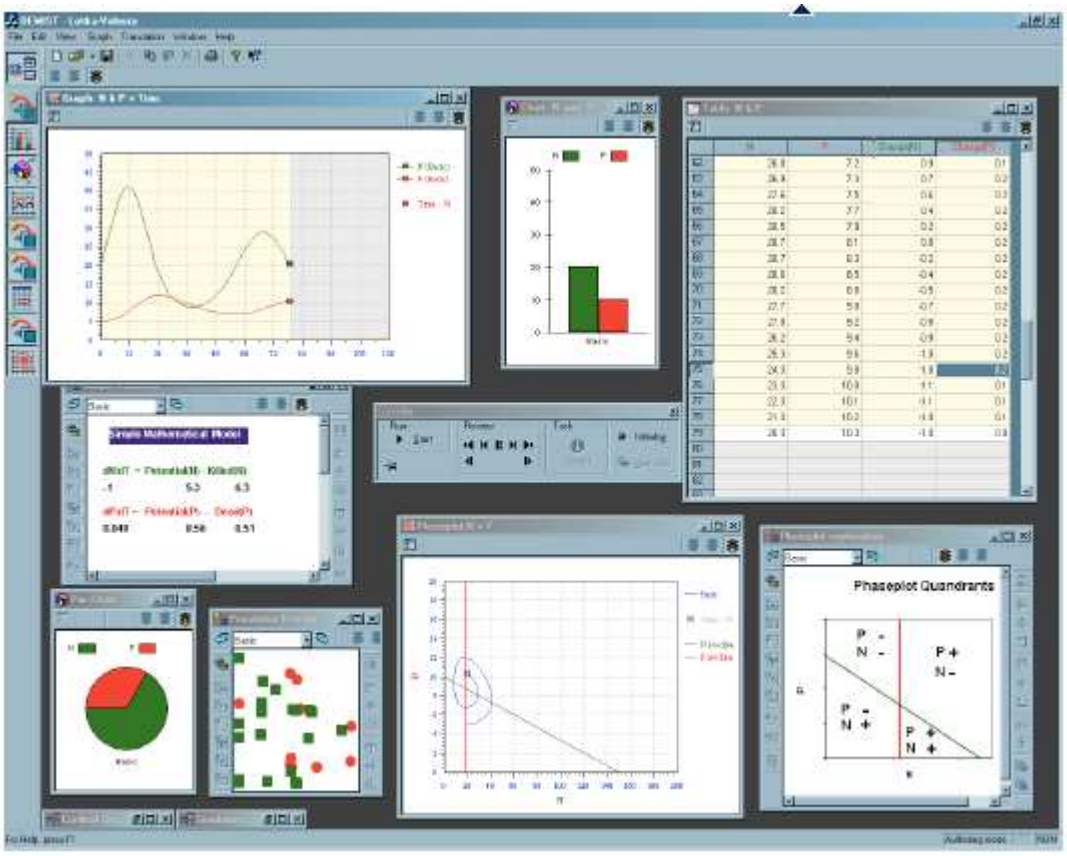


Figura 2.3: Ambiente DEMIST

- Os aspectos do mundo representante que fazem parte da modelagem;
- A correspondência entre os dois mundos.

Um uso adicional de múltiplas representações é explorar os processos computacionais variados apoiados por diferentes representações. Por exemplo, [48] apud [41] propôs um esquema de percepção que processa e vai agrupando informações pertinentes e conseqüentemente torna o processo de procura e reconhecimento mais fácil.

Outro uso de múltiplas representações é ajudar os estudantes a desenvolverem um entendimento avançado de um determinado domínio devido a limitação a interpretação das representações e tarefas. Isto pode ser alcançado por duas maneiras:

- Empregando uma representação familiar ou concreta para apoiar a interpretação de um segunda representação abstrata menos conhecida;
- Explorando propriedades inerentes de uma representação para forçar a interpretação de uma segunda representação.

Geralmente ambientes de simulação exploram múltiplas representações empregando uma representação familiar ou concreta para apoiar a interpretação de um segunda representação abstrata menos conhecida. Por exemplo, micro-mundos como DM3 (Direct Manipulation of Mechanical Microworlds) [30] apud [41].

Para que as múltiplas representações possam ter um efeito positivo melhor foi definida a taxonomia funcional de MRE [41], que consiste em três elementos como mostra a figura 2.4:

- Papéis complementares;
- Restrigir interpretação;
- Construção de compreensão aprofundada.

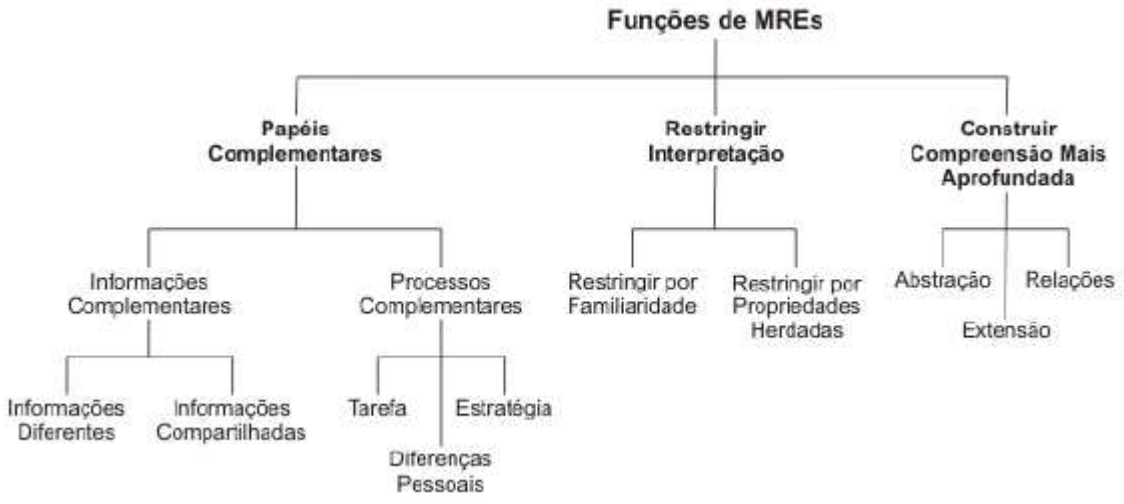


Figura 2.4: Taxonomia funcionalista de MRE

Combinando estes três elementos e sua subdivisões para propor uma combinação de desenvolvimento para cada uma das funções de múltiplas representações. Assim, as múltiplas representações podem servir para muitas funções benéficas, especialmente quando são projetados sistemas para minimizar demandas de aprendizagem.

Porém, nem todos os pesquisadores são otimistas sobre o potencial para sistemas multi- representacional. Em particular, Pimm [16] apud [41] adverte o uso de múltiplas representações juntas pois elas podem não ser neutras. Ele sugestiona para que uma

representação predomine e que assim não será visto como uma única representação. Assim, os significados não serão associados com a relação entre representações, mas com a representação dominante. Lowe[38] apud [41] também sugere isso ao utilizar múltiplas representações porque os estudantes focalizam a atenção em uma representação dominante.

Finalmente, vários estudos por Sweller e colegas [41] (por exemplo [25], [26]) demonstraram que quando a informação é apresentada em várias representações em lugar de uma única representação, ocorre a divisão de atenção a qual conduz ao aumento da carga cognitiva e aprendizagem menos efetiva.

Contudo esses argumentos e as evidências apresentadas indicam que múltiplas representações apóiam o aprendizado, mas para que elas tenham êxito os projetistas de software devem considerar cuidadosamente como serão utilizadas as múltiplas representações para que estas tenham o efeito desejado com a combinação de diferente representações ao aprendizado. Portanto, os critérios importantes para pesquisa nesta área é definir representações precisamente adequadas.

CAPÍTULO 3

CONCEITOS FUNDAMENTAIS

Este capítulo apresenta os conceitos fundamentais para o entendimento da solução desenvolvida no presente trabalho de Mestrado.

3.1 Micro-mundos de exploração

De acordo com a classificação de softwares educativos este projeto se enquadra como um ambiente de aprendizagem do tipo micro-mundos (microworlds) de exploração. O conceito de micro-mundos de acordo com Fisher, Brown e Burton [21] está inserido no paradigma de desenvolvimento de ambientes de aprendizagem que se adéquam o máximo possível ao aprendiz.

Os micro-mundos surgiram ainda na década de 60, sendo que o mentor da idéia foi o pesquisador Seymour Papert [13]. Os micro-mundos se diferenciam dos sistemas CAI porque é o ambiente que permite ao aluno trabalhar de acordo com o seu próprio ritmo permitindo a ele construir sua própria solução utilizando os recursos que o ambiente oferece.

Entretanto, foi consolidado por meio da linguagem LOGO e do "micro-mundo da tartaruga"[1] onde a ênfase da aprendizagem está na construção do conhecimento por parte do aluno e não apenas na transmissão de conhecimentos estático no sentido professor–aluno. Além da evolução do conceito, também se multiplicaram os trabalhos relacionados à área, procurando explorar as potencialidades pedagógicas relacionadas às diversas disciplinas, por exemplo, a matemática e a geometria.

Uma das características de micro-mundos é inicialmente apresentar um mundo simples ao aprendiz e durante o aprendizado, o mundo vai se tornando mais complexo. Outra característica é a possibilidade de interação do aprendiz com os "objetos da interface",

[1] jogo desenvolvido na linguagem LOGO

os quais possibilitam ao aprendiz trabalhar de forma diversificada, segundo seu próprio ritmo.

Um sistema ou ambiente pode ser considerado um micro-mundo, quando este possuir algumas características: tais como utilizar objetos e funções para atingir o resultado computacional e gráfico desejado; representar um domínio abstrato; proporcionar várias opções de soluções para atingir um objetivo; e principalmente permitir a manipulação direta de objetos pelo aprendiz. Esta última é a que diferencia um micro-mundo de um STI, porque nesse último, é o sistema quem controla o aprendizado, captando informações do aprendiz e por meio destas faz uma avaliação, a qual toma uma decisão sobre qual será a próxima ação executada ou disponibilizada ao aprendiz.

O conceito de micro-mundo será utilizado nesse projeto como instrumento fundamental de abstração e compreensão, na representação das mudanças de estados que ocorrem durante a execução do algoritmo de busca. A abordagem pedagógica será voltada à visualização das árvores de memória durante a execução de programas e a interação que permitirá ao aprendiz realizar a sua investigação na tentativa de entender o processo de solução automático de problema por um algoritmo de busca heurística.

A visualização de memória é um conceito de representação de conhecimento para descrever eventos estereotípicos. Assim a memória é organizada dentro de estruturas que reúnem eventos com características similares através de abstrações e hierarquias do tipo todo-parte. Com relação ao conteúdo, a alocação de memória forma uma estrutura de conhecimento que representam experiências.

Os exemplos representam eventos através de visualização que incluem situações e são representadas através de informação normativa e descritiva. As visualizações de exemplos são expectativas associadas às situações de uma experiência, conseqüentemente elas estão sujeitas a mudar quando há mudança dos exemplos. A entidade básica do modelo de memória dinâmica permite representar computacionalmente um modelo de organização de memória que compreende recordar, entender, experienciar e aprender.

3.2 Representação da árvore de memória

Um dos modos de representação da alocação dinâmica da memória realizada por um algoritmo de busca heurística é a utilização de representações na estrutura de árvores. Geralmente um professor em sala de aula faz desenhos no quadro negro de partes dessa árvore para que o aluno compreenda o funcionamento do algoritmo.

Os dados são uma estrutura relativamente complexa, as estruturas hierárquicas são na realidade um caso particular de grafos. Horowitz, por exemplo, define uma árvore como um grafo conexo sem ciclos [44] apud [31]. No entanto, a importância da sua utilização e a diversidade de visualizações desenvolvidas justificam o destaque dado às estruturas hierárquicas.

No contexto de grafos, o seu desenho constitui por si só uma área de estudo e existem conferências dedicadas especificamente à discussão deste assunto. Foi encontrada uma bibliografia de vários pesquisadores com mais de 300 artigos relativos ao desenho de grafos. Entretanto o desenho de grafos *Graph Drawing* é diferente da visualização de grafos *Graph Visualization* a qual é considerada uma sub-área da visualização de informação [31].

3.2.1 Árvores OU

As estruturas de dados de árvores representam uma relação hierárquica em que um nodo pode ter vários sucessores e somente um antecessor. De acordo com [15], uma árvore é um conjunto finito de um ou mais nós, tais que:

- Existe um nó denominado raiz;
- Os demais nós formam: (a) m >= 0 conjuntos disjuntos; (b) s1, s2, ... sm, tais que cada um desses conjuntos também é uma árvore (sub-árvore).

As árvores podem ser representadas de diversos modos:

- Representação hierárquica;
- Representação por conjunto;

- Representação por expressão parentetizada;
- Representação por expressão não parentetizada.

A representação hierárquica apresenta melhor visualização, assim está integralmente relacionada com o trabalho. A representação parentetizada e não parentetizada são úteis para guardar em arquivos os dados de uma árvore.

A representação em árvore pretende facilitar o entendimento, pois é um tipo de estrutura de dados já conhecida por alunos de ciência da computação. Além disso, a representação hierárquica na visualização da alocação de memória permite ver quais estados do mundo serão atingíveis diretamente a partir de um outro estado pela sua representação em nodos e arestas, o que pode ser uma forma intuitiva de representação visual ao aprendiz.

Geralmente os algoritmos de busca de IA se utilizam de árvores OU, pois um algoritmo que quer resolver um determinado problema, ao atingir um nó arbitrário, opta por um dos ramos de saída, baseado em algum critério e, após um certo número de nós pesquisados, encontrando a solução, também definida por algum critério, pode encerrar a busca e parar.

A correção dessa abordagem se justifica pois cada ramo de saída de cada nó é disjuntivo (OU) e então, se a solução é verdadeira quando percorremos um dos ramos, não precisamos verificar outros ramos da árvore de busca. A Figura 3.1 representa de forma gráfica uma árvore OU.

Algoritmos de busca heurísticas são técnicas de inteligência artificial aplicadas a problemas de alta complexidade teórica [35] que não são resolvidos com técnicas de busca convencionais, principalmente as de natureza puramente numérica.

A "complexidade"de um problema está diretamente relacionada ao seu "Espaço de Busca"correspondente. As estratégias de busca estão baseadas em diversos critérios [35] como:

- Espaço - é a quantidade de memória necessário para realizar a busca, ou seja, representar o espaço de estados;
- Completude - a estratégia de busca sempre encontra uma solução se ela existir ou quando existir mais de uma solução;

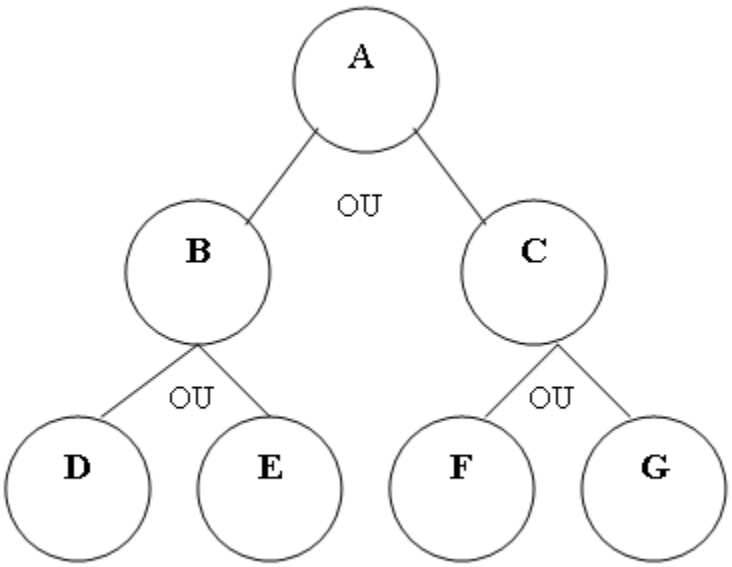


Figura 3.1: Representação de uma árvore OU

- Otimalidade - a estratégia encontra a melhor solução (em geral a de menor custo) quando existem diferentes soluções;

- Tempo - quanto tempo é gasto para encontrar a solução (como tempo dos algoritmos de busca heurística não é foco do trabalho presente, foram omitidos todos os detalhes desta natureza);

Para um algoritmo de busca heurística, é necessária uma representação do conhecimento do domínio em estados, os quais se enquadram em modelo computacional por meio de variáveis de memória. A solução do problema, cujo espaço de busca é formado por transformações sucessivas de estados em ordem de geração de acordo com a função heurística, é um processo complexo. É exatamente o uso de heurística que provoca a redução da explosão combinatória de possibilidades de busca.

Sendo assim, dado um estado inicial e o estado final, a ferramenta indica como o algoritmo encontra uma trajetória (ou plano) que leve de um estado ao outro. Em outras palavras, a solução é o caminho ou o conjunto de estados necessários (ou ações) para passar do estado inicial até o estado final.

No ensino de busca heurística, é sempre difícil ao aprendiz entender o funcionamento do algoritmo. Essa dificuldade dá-se principalmente no que se refere às mudanças de estado, isto é, na realização das operações. Geralmente, quando é executado um algoritmo de busca heurística este exibe ao aprendiz somente o "estado solução ou o caminho solução". Devido a isso, o aprendiz não consegue desenvolver o conceito mental de como o algoritmo chegou àquela solução.

As buscas em árvores OU são tipicamente aplicadas a classes de problemas como os de encontrar o bom caminho (caminho cujo custo é menor do que o de escolha aleatória). Todavia, o problema de encontrar o caminho mínimo pode ser interessante porque completude e otimalidade muitas vezes são aspectos exigidos no enunciados de problemas desse tipo de busca.

3.2.1.1 Algoritmo A*

O algoritmo de busca A* tenta minimizar o custo total da solução (plano, trajetória ou caminho), combinando a busca econômica com uma busca completa. É um algoritmo ótimo e completo porque encontra o caminho de custo mínimo desde que a função heurística nunca superestime o custo real.

Este algoritmo evita expandir caminhos que estão com o custo alto, considerando os estados que têm menor expectativa de custo. Portanto, a função de avaliação que determina a ordem de inserção de uma trajetória em sua lista de prioridades, e consequentemente o próximo nodo a ser expandido, é a seguinte: f(Ec) = g(Ec) + h(Ec), onde h é a função heurística definida para calcular a estimativa de custo de percurso do estado corrente (Ec) até o estado final (Ef), e g é o custo real acumulado para o percurso do estado inicial (Ei) até o Ec. De acordo com a demonstração formal, reproduzida no conhecido livro de Russell e Norvig [35], quando a função h utilizada é admissível (nunca superestima o custo real), além de completo, o algoritmo de busca A* é minimalista (encontra a trajetória de menor custo).

Para o algoritmo encontrar a solução mínima não é necessário carregar na memória todos os nodos em amplitude, pois o objetivo da heurística é permitir que o algoritmo

expanda o caminho de menor custo que, em qualquer momento, passa por um certo estado corrente (Ec). Isso é feito de maneira a cobrir uma ou mais trajetórias do estado inicial até o nodo final, definindo uma seqüencia de estados ou ações sem percorrer todo o espaço de busca.

Neste trabalho, a ferramenta projetada e implementada fornece elementos de visualização da memória tanto do algoritmo A* quanto de qualquer outro escolhido para que as características inerentes fiquem mais claras do que apenas os conceitos apresentados nos livros. O algoritmo A* não é um dos melhores algoritmos de busca mas no aspecto pedagógico ele é excelente para a transmissão de aspectos importantes e genéricos sobre busca heurística.

3.2.1.2 Outros algoritmos de árvore OU

Esta seção apresenta alguns dos vários algoritmos de busca heurística que existem além do A*. O algoritmo de busca heurística IDA* *Iterative Deepening A** é um dos mais fortemente baseados nas idéias do A*. Para o aprendiz ter um conhecimento melhor sobre seus conceitos básicos, além de conhecer o A*, é preciso abordar os seguintes elementos:

- Busca de custo fixo;
- Aumento iterativo do custo da busca;
- Contornos de custo do espaço de busca.

Este algoritmo foi criado para suprir a deficiência do A*, onde o consumo de memória cresce exponencialmente. Além da busca de custo fixo limitar o consumo de memória, ela é capaz de evitar as deficiências da busca em profundidade, que em geral acha soluções muito longas.

O algoritmo IDA* precisa impor um corte na trajetória ao atingir o custo máximo sem incluir o E(f) para dar continuidade à busca por meio de outra trajetória que ainda pode incluí-lo (Ef) dentro do custo máximo definido.

Dependendo do custo máximo, a busca de custo fixo é completa mas não é minimalista. Portanto o aumento iterativo do custo, combinado com a busca de custo fixo inclui as

características que deixam o algoritmo minimalista e completo. Entretanto, o IDA* não lida com custo negativo. Pode-se apenas dizer que o IDA* é minimalista e completo porque o algoritmo inicia com altura zero (uma questão formal que traz resultados interessantes quando o limite passar a ser o custo da trajetória e não o seu tamanho). Cada ciclo que termina sem incluir o Ef é seguido de outro cujo limite de custo é exatamente o da trajetória de menor custo que ultrapassou o limite anterior.

Para se compreender melhor o conceito de contorno de custo do espaço de busca, inicia-se a explicação por meio da associação do mesmo a um conjunto de trajetórias cujo custo é menor ou igual a um valor-limite. Isso resulta na possibilidade de se manter a expansão de trajetórias cujos valores de f estão comprendidos dentro do valor-limite. O valor-limite do primeiro contorno de custo é f(Ei).

Vários contornos de custo podem ser ajustados durante o processo para que um novo ciclo de busca possa se reiniciar, sempre a partir do Ei, e tente incluir o Ef. Pode-se perceber que o algoritmo tem suas vantagens em relação ao A* devido ao baixíssimo consumo de memória (guarda apenas uma trajetória), mantendo as mesmas vantagens do A* pois sempre encontra o caminho mínimo (se existir um).

Outro algoritmo interessante é o SMA* *Simplified Memory-Bouded A**, o qual tenta corrigir algumas limitações do algortimo IDA*. Para compreender melhor os seus conceitos básicos, além de conhecer o A* é necessário conhecer a busca de tamanho fixo de memória, pois a lista de prioridades com as trajetórias sofre controle por meio de um valor de tamanho máximo que representa a soma dos tamanhos das trajetórias registradas no momento.

Esse algoritmo elimina a(s) pior(es) trajetória(s) para que uma nova trajetória seja inserida. Porém, o SMA* também diminui a repetição de expansão de estados por meio do registro do custo de uma trajetória eliminada. Em síntese, ele realiza a busca de forma limitada pela memória disponível, armazenando os nodos explorados até o limite dessa memória e eliminando trajetórias caras da memória quando necessita de espaço para explorar novos nodos.

Como simplificação ainda maior da busca A*, aproximando-se da busca em profundidade, pode-se citar um outro algoritmo de busca heurística bastante conhecido no meio acadêmico. Seu nome é "Melhor Escolha"(Best First) e ficou assim conhecido por ser uma variação da busca A* reorientada apenas pelo menor valor da função heurística h, a ser aplicada aos descendentes do Ec. Em outras palavras, o algoritmo é guiado apenas pela estimativa de menor distância do Ec ao Ef. Suas características são:

- Pouco uso de memória;
- Não garante completude nem otimalidade da solução.

Cabe ressaltar que o A* e suas variações constituem atualmente os algoritmos mais utilizados pelos "quebra-cabeças"(não pelos "jogos adversaristas") para resolver problemas de determinação de caminho em ambientes bidimensionais e tridimensionais. E no meio acadêmico por ser um algoritmo que envolve os aspectos importantes como otimalidade e completude.

3.2.2 Árvores E-OU

As árvores E-OU foram originalmente estudadas por pesquisadores preocupados com a representação de problemas de sentido comum de uma forma genérica, tentando seguir o raciocínio utilizado por um humano na busca de uma solução para o mesmo [35]. Com a intenção de suprir as deficiências das situações representadas apenas por árvores OU, como no caso dos "jogos adversaristas", as Árvores de Jogos, que são variações das árvores E-OU, foram propostas para resolver esse tipo de problema, principalmente sob a forma de cálculo do "próximo lance plausível".

Se uma solução for encontrada por meio de um estado de terceiro nível (ou mais profundo ainda) da árvore de busca, por exemplo, a vitória da máquina, a busca deve continuar. Isso ocorre pois o jogador adversário pode escolher outro caminho em alguma bifurcação de nível superior ao do nodo solução, em busca de vantagem (ou até a vitória), e não aquele ramo que a máquina pretende seguir. Para garantir a correção para esses casos é necessário que os nodos do adversário tenham todas as suas saídas extensivamente

exploradas e, portanto, os ramos de saídas são conjuntivos (E) conforme representado graficamente na figura 3.2. Assim, conceitua-se as árvores E-OU a partir dos seus casos mais comuns, as árvores de objetivos e as árvores de jogos.

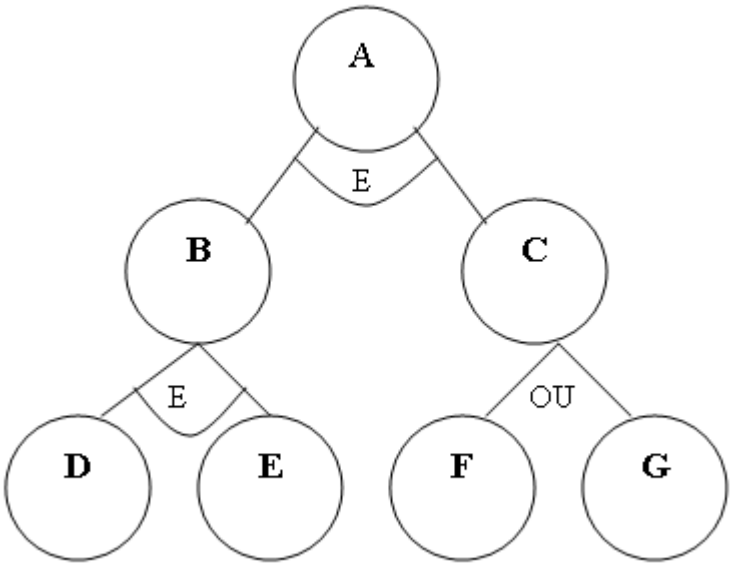


Figura 3.2: Representação de uma árvore E-OU

Uma árvore de objetivos descreve a situação na qual um objetivo principal pode ser atingido por meio da aplicação de um método de solução de problema. Se houver mais de um método disponível, um algoritmo de solução de problemas pode ter que tentar vários deles de forma combinada. Isto se resume a um problema de busca tradicional ou outro método (OU). Entretanto, caso a execução dos métodos escolhidos necessite da execução de duas ou mais etapas em série então a trajetória correspondente a esse método precisa incluir um nodo E.

Uma árvore de jogos é uma árvore de objetivos na forma de uma árvore E-OU sem restrições nos nodos E. Ela inclui opções de caminhos de movimentos que configuram todas as partidas possíveis do jogo onde, alternadamente, cada jogador faz o seu movimento. Cada lance leva sempre a um conjunto finito de estados (configuração de peças) totalmente previsíveis. A poda da árvore de jogo por ser feita por meio de uma função de avaliação estática, a qual é de natureza heurística pois expressa a vantagem material ou posicional de um jogador qualquer [35].

Em resumo, os conceitos de completude e otimalidade (capacidade de encontrar a solução ótima) não se aplicam tão simplesmente no mundo de jogos como foi o caso da busca em grafos OU. Em boa parte, isso ocorre porque uma função de avaliação estática em jogos não possui caractrísticas de monotonicidade que tipicamente encontramos na busca em grafos OU.

3.3 Representações visuais

Representações visuais ou gráficas correspondem às "figuras"ou "imagens"empregadas para retratar o conjunto (ou subconjunto) de dados sob análise. Além dos gráficos tradicionalmente utilizados para apresentação de dados como os gráficos de pontos, de linha, de barras, de torta e histogramas de freqüência, que permitem observar relações entre atributos, uma série de representações gráficas mais ou menos complexas são empregadas para codificar através de elementos visuais (cores ou símbolos geométricos) tanto valores como relacionamentos entre entidades ou elementos de dados [1].

As mais simples representações gráficas até as representações visuais foram definidas como um modelo de representação mental que contribui diretamente na aquisição de conhecimento. Sobre aquisição de conhecimentos, Santaella e Nöth [28] relatam que "modelos de representação mental do conhecimento são tão antigos quanto a filosofia cognitiva". Eles também elencam a existência de quatro modelos de representação mental que descreveriam a forma da representação mental enquanto os seguintes aspectos:

- Idéias: no sentido de uma matéria mental estruturada (não se restringindo apenas a considerar as coisas existentes fisicamente enquanto matéria estruturada, mas as próprias idéias);
- Imagens:(considerada por alguns teóricos da atual ciência cognitiva e questionada por alguns representantes da teoria simbólica da representação);
- Símbolos (defendido por alguns teóricos da imagem que consideram que a linguagem é representada mentalmente na forma de símbolos e outros postulando a tese de que mesmo imagens na forma de símbolos são representadas mentalmente);

- Estados neurofisiológicos (representa-se mentalmente o conhecimento na forma de processos de ativação ou inibição fisiológica de ligações sinápticas em redes neuronais).

Portanto o Conhecimento Estrutural descreve as estruturas do conhecimento, ou seja, a forma como o conhecimento está estruturado na mente de um especialista. A estrutura mental mais conhecida é definida por:

- Conjunto de regras
- Relações entre conceitos
- Relações entre conceitos e objetos

Com o intuito do aprendiz adquirir conhecimentos especilizados de forma estruturada esse projeto utiliza técnicas de visualização para representar a solução de problemas básicos de planejamento, em que a solução é um caminho para um estado no mundo (conjunto de estados ou ações), de maneira diferente de problemas de busca onde a solução é encontrar apenas o estado solução (como no exemplo da montagem de palavras cruzadas).

Elementos semânticos abstratos de representação estudados para serem utilizados neste trabalho como parte da solução de um problema de busca estão definidos em cada subseção seguinte. Tais elementos se efetivam como representações de estados do mundo ou domínio.

3.3.1 Hierarquias

A visualização de estruturas hierárquicas normalmente provoca dificuldades de gestão do espaço da tela, que se agravam à medida que o número de elementos da estrutura aumenta. Uma das representações mais usuais de hierarquias é a utilização de um grafo bidimensional em que cada nó está ligado por segmentos aos seus descendentes [31].

Com base nisso, a árvore de memória de cada algoritmo de busca heurística será aqui apresentada na tela em níveis de acordo com a figura 3.3, representando o estado (nodo) pai e seus respectivos estados (nodos) filhos em um nível abaixo. Portanto o aprendiz

poderá visualizar e comparar a dimensão da árvore, que é o espaço de busca de estados do domínio.

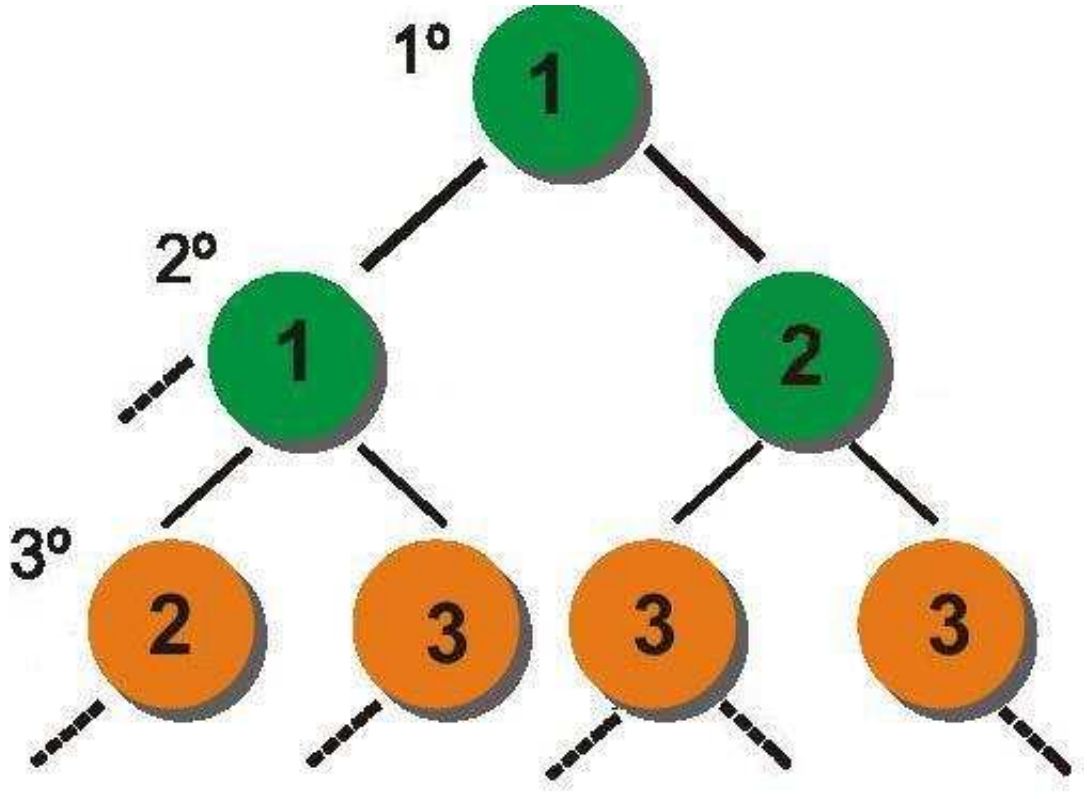


Figura 3.3: Representação de uma estrutura hierárquica

3.3.2 Cores

A cor é fundamental em visualização de informação e o olho humano é extremamente sensível às variações de cor. Embora seja possível que a percepção de cor de um aprendiz possa ser muito diferente de outro, evidências experimentais sugerem que os relacionamentos entre cores são, em muitos aspectos, relativamente livres de influências culturais e individuais [40]. Historicamente, a cor tem sido caracterizada pela consideração da sua aplicação, que inclui:

- características físicas da cor;

- mecanismos do sistema visual humano;
- aplicações para codificação e reprodução;
- aplicação no design e interatividade.

Neste trabalho, foi estabelecido um padrão de cores para representar os estados da memória. Os nodos que ainda não foram percorridos serão representados na cor cinza claro para que fique melhor para o aprendiz visualizar que aqueles estados ainda não estão na memória. Para os que estão na memória instantânea serão coloridos na cor azul e para os nodos que já foram desalocados será usado o amarelo. O "caminho solução"estará destacado com a cor vermelha. A aplicação desta definição pode ser vista no exemplo da Figura 3.4.

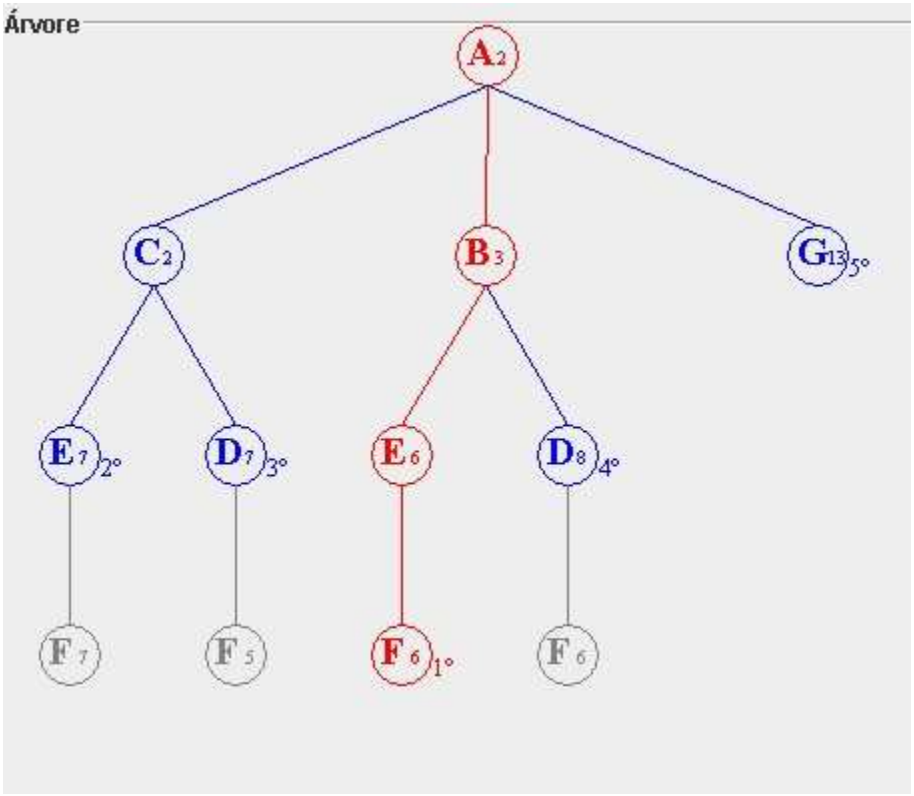


Figura 3.4: Cores definidas para estados de memória

A aplicação de cores adequadas a uma interface de trabalho própria para jogos de vídeo e com o propósito de ensino, é extremamente importante. As cores podem diminuir o esforço cognitivo do aprendiz ao visualizar a interface.

As cores possuem interpretações óbvias como por exemplo a direção. As cores quentes como os vermelhos e os laranjas devem vir sempre na frente, pois estas sempre conduzem nossas atenções[6]. Sendo assim, a utilização destas cores facilita o contraste. Por outro lado, um certo cuidado deve ser tomado com as cores frias, como os azuis e os verdes, pois elas tendem a passar desapercebidas por nosso aparelho visual.

3.3.3 Destaque de elementos

Na presente abordagem, foi decidido que os elementos de maior relevância serão destacados dinamicamente durante a execução da busca. O nodo com melhor f, o qual será o próximo a ser expandido, será destacado com um tom de cor mais forte. Isso possibilita ao aprendiz acompanhar cada mudança ou expansão de nodos instantaneamente.

3.3.4 Sub-janela

Foram também utilizadas sub-janelas ou "pop ups"para exibir informações em texto sobre o conteúdo de cada nodo. Deste modo, o aprendiz terá à sua disposição dados como o custo acumulado de transformações, o resultado da função heurística aplicada a um estado qualquer, assim como a representação de outros dados de conteúdo de um estado no caso de refinamentos futuros do trabalho aqui realizado. Para ter acesso a essas informações o aprendiz só precisa clicar sobre o nodo desejado na árvore e se abrirá a janela gráfica.

3.3.5 Legendas

Serão elementos de visualização de grande importância para facilitar o entendimento do aprendiz, pois a diversidade de cores e representações pode vir a dificultar a compreensão do que está sendo apresentado. O uso de legendas contribui para minimizar o risco do aprendiz adquirir conhecimento inconsistente.

Além da ajuda do sistema, a legenda será exibida no canto inferior direito para qualquer dúvida de informação instantânea e para o aprendiz não ter que acionar constantemente as telas de ajuda do sistema. Ele simplesmente poderá visualizar essas informações

rapidamente como mostra a figura 3.5

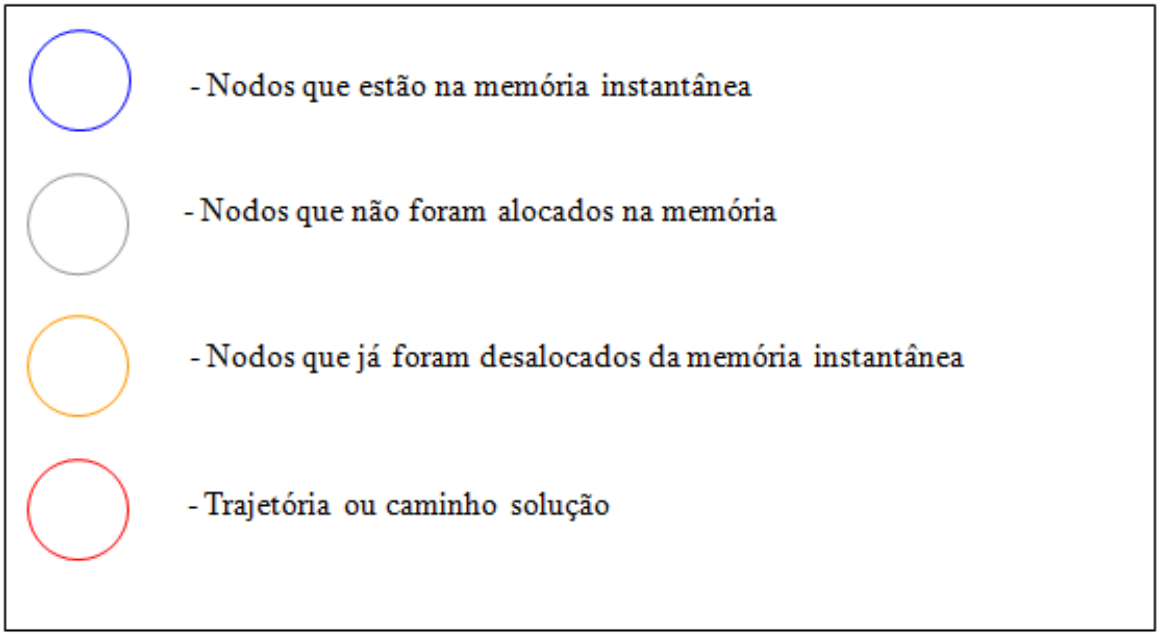


Figura 3.5: Exemplo da legenda

3.3.6 Contraste

O objetivo do contraste é acrescentar atrativos visuais ao design de uma página, ou seja, criar hierarquia organizacional entre diferentes elementos. Porém, para que apareça esta hierarquia, o contraste deverá ser realmente significativo.

Existem três categorias sugeridas com relacionadoras dos elementos de um layout [6]:

- concordante: é quando não há contraste algum entre dois elementos em uma página, gerando um resultado normalmente insosso. Como exemplo é possível citar: margens com o mesmo tamanho, título e texto feitos na mesma letra, etc.;
- conflitante: quando o designer busca inovar sem ousar muito. Surgem, assim, diferenças de tipo, corpo e estilo de texto, imagens com pequenas variações no estilo, etc., gerando uma similaridade entre os elementos. Com isso, a similaridade dos conteúdos irá dificultar a leitura;

- contrastante: este, por sua vez, atrai a visão por imediato e cria uma real curiosidade e interesse por parte do usuário. Neste caso, há uma variação de tamanho, peso, estilo, forma e cor. Quanto maior variação na quantidade ou intensidade dos contrastes, mais interessante poderá ser o efeito.

Outro fator importante para o desenho de um *layout* são os espaços vazios. Locais para a sua utilização são as entrelinhas, colunas e margens do *layout*. É através destes espaços que se busca equilibrar, reforçar a unidade de grupos, harmonizar áreas e aumentar contraste. São exatamente esses espaços em branco que irão dar formas ao *design* da interface, apesar de que muitas pessoas os considerarem áreas perdidas, buscando preenchê-las e assim poluindo a tela.

São dois os propósitos básicos do contraste: (1) criar interesse sobre uma página se ela tiver um aspecto interessante, atrairá mais a leitura; (2) organizar a informação para o usuário ser capaz de compreender de imediato a maneira através da qual os dados são estruturadas, incluindo o fluxo lógico de um item para outro do dado. Os elementos que estão em contraste não podem confundir o leitor ou criar um foco que não seja correto.

3.4 Elementos de interação

Além de elementos semânticos abstratos de representação, o aprendiz terá disponível os elementos de interação os quais possibilitam executar um conjunto de ações com as quais efetivará uma investigação que lhe propicie compreender, ao menos momentaneamente, fatores dinâmicos. Abaixo estão listadas algumas ações deste conjunto.

Uma vez que uma representação visual estática por si só freqüentemente não é suficiente para propiciar as condições necessárias para a compreensão de grandes conjuntos de dados. Normalmente são disponibilizadas funções pelas quais um usuário pode explorá-los, através de ações em diferentes níveis. Estas ações ocasionam alterações na representação visual de modo que novos aspectos do conjunto de dados possam ser observados. [1]

Assim o aprendiz pode verificar que alterando o estado inicial ou estado final, por exemplo, a busca tem um comportamento diferente pois varia a hierarquia, a altura ou

a largura. Principalmente o espaço de busca e o consumo de memória são diferentes. Quando se compara algoritmos diferentes essas variações permitem o entendimento do aprendiz passo a passo de cada aspecto do algoritmo.

3.4.1 Mudar de algoritmo

Esta opção serve para o aprendiz selecionar outro Algoritmo de Busca. Ao iniciar, o aprendiz seleciona o algoritmo que pretende estudar e visualizar o seu funcionamento. Entretanto, em qualquer outro momento, ele pode parar o que está sendo executado e reiniciar a execução com outro algoritmo.

Isso permite ao aprendiz visualizar e diferenciar o comportamento de cada algoritmo, por meio da visualização da alocação de memória deixando visível como o algoritmo parte do estado inicial e progride até o estado final. Porém, em sua versão atual, o protótipo aqui implementado (ver Capítulo 4) só permite a seleção de algoritmos de busca heurística previamente implementados na base interna da ferramenta.

3.4.2 Alterar o estado inicial

Para o domínio definido nesse caso busca em grafos, o aprendiz poderá determinar qual será o raiz da árvore, ou seja, o estado inicial. Este estado poderá ser selecionado dentro das opções possíveis dos nodos existentes no grafo. Caso o aprendiz não selecione nenhum nodo em especial, então a seleção do estado inicial será feita automaticamente pelo estado *default* [2].

3.4.3 Alterar o estado final

De forma análoga ao estado inicial, este estado também pode ser definido pelo aprendiz ou ser utilizado um estado *default*. Pois dependendo da posição do nodo estado final o comportamento da busca é diferente. O que antes era o caminho solução após alterar o estado final pode não ser mais o caminho solução.

[2]Estado padrão definido no tutor

3.4.4 Execução manual ou passo a passo

Além da execução automática, onde o aprendiz apenas observa o algoritmo executando e montando a árvore de memória sozinha, é disponibilizado ao aprendiz esse tipo de execução, onde é possível avançar, passo-a-passo, a cada ciclo de execução do algoritmo com um *click* no *mouse*. Assim, o aprendiz poderá acompanhar a execução no tempo que deseja.

3.4.5 Retornar ao passo anterior

Assim como o sistema possibilita toda a execução passo a passo, será possível também retornar ao passo anterior no momento desejado ou naquela mudança em que não ficou bem definido ao aprendiz, possibilitando voltar ao passo anterior. Após o retorno, é possível continuar a execução do próximo passo normalmente, ou acionar a execução automática

3.4.6 Reiniciar a execução

Durante a execução, caso o aprendiz não consiga acompanhar as ações e não saiba mais como está sendo executado o algoritmo, então o aprendiz poderá reiniciar a execução quantas vezes for necessário para adquirir o conhecimento.

3.5 Informações adicionais

Cada nodo irá conter armazenadas as seguintes informações que ficarão visíveis a partir de alguma ação do usuário:

- O conteúdo de um estado qualquer (mesmo que representado apenas de forma textual);
- A sua profundidade na árvore de busca;
- O valor da heurística (representado por h);

- O custo-guia de geração de estados (representado por g).

Os detalhes de representações internas necessárias ao funcionamento da máquina de visualização foram estudados com critérios necessários como parte do desenvolvimento desse projeto.

CAPÍTULO 4

FERRAMENTA DE VISUALIZAÇÃO

Para simplificar o trabalho de implementação de um protótipo que valida parcialmente os conceitos desenvolvidos neste trabalho, foi decidido que apenas a organização da memória utilizada por um algoritmo de busca teria seus aspectos visuais revelados pela ferramenta de software. Por si só, isso já se constitui em apoio ao aprendiz para que ele tenha uma idéia mais clara da complexidade algorítmica de uso da memória por meio da visualização das áreas alocadas ou devolvida ao sistema operacional em cada "ciclo"do algoritmo. Adicionalmente, com o intuito de ampliar o entendimento das múltiplas faces de busca heurística em geral, as representações visuais utilizadas para denotar cada elemento da árvore de memória e seus atributos são altamente abstratas. Mesmo assim, ainda ficam claros os diversos aspectos dinâmicos de como o algoritmo encontra a solução de um problema específico.

O nome da ferramenta é VIMAP baseado na visualização de árvores de memória passo-a-passo. A VIMAP foi construída seguindo a arquitetura funcionalista, a qual reflete a modelagem e representação do conhecimento no contexto de visualização da execução de programas, um caso especial da visualização científica. Os conceitos da psicologia cognitiva foram de grande importância, além dos conceitos referentes às características dos aprendizes (peritos e iniciantes) desenvolvidas por Lesgold[22] e Direne [3].

O conceito de que a perícia é formada por exposições de exemplos é amplamente aplicado neste trabalho por meio do robusto equipamento de visualização do protótipo. Tal equipamento permite ao professor ou mesmo o aprendiz cadastrar o grafo desejado para visualizar o comportamento da busca neste determinado grafo.

Além do apoio ao desenvolvimento de perícias, a ferramenta irá atuar também no desenvolvimento de conhecimento na fase intermediária de princípios e perícia. Ressalta-se que, para o aprendiz utilizar a ferramenta, ele deverá ter o conhecimento prévio dos

princípios de algoritmos de busca heurística.

O protótipo pode ser classificado como um micro-mundo elaborado em torno do domínio do conhecimento de algoritmos de busca heurística. Do ponto de vista educacional, ele será explorado por meio da interação entre o aprendiz e a ferramenta de aprendizagem a qual disponibiliza objetos ao aprendiz através da interface. Essa, por sua vez, assume um papel fundamental na aprendizagem por permitir a configuração da árvore de memória de acordo com o desejo do usuário.

A linguagem de programação utilizada para a implementação foi Java[1]. As características que levaram à seleção desta linguagem foram:

- Familiaridade com a linguagem;
- Uma linguagem de programação orientada a objetos, um paradigma em que princípios e técnicas favorecem a implementação de tarefas de manipulação de eventos em questão;
- A comunidade Java dispõe de uma extensa biblioteca de componentes de software para diversas finalidades;
- Portabilidade, pois a compilação gera um código que será interpretado pela plataforma de execução (denominada Máquina Virtual Java[2]), teoricamente, independe do hardware e do sistema operacional utilizados;
- Interfaces personalizáveis, qualidade que pode subsidiar a Interação Humano-Computador e atrair visualmente os aprendizes do ambiente desenvolvido.

4.1 Visão geral da arquitetura

A arquitetura funcionalista da VIMAP é ilustrada na figura 4.1, composta pelas seguintes unidades funcionais para confirmar a solução apresentada: galeria de algoritmos de busca(A.B.), interface (E/S), carregador de A.B., gerenciador de estado de memória, alternador de

[1]JDK
[2]Do inglês Java Virtual Machine (JVM)

contexto, processador de busca gradual, mapeador gráfico, visualizador, combinador completo, gerenciador de grafos e galeria de grafos.

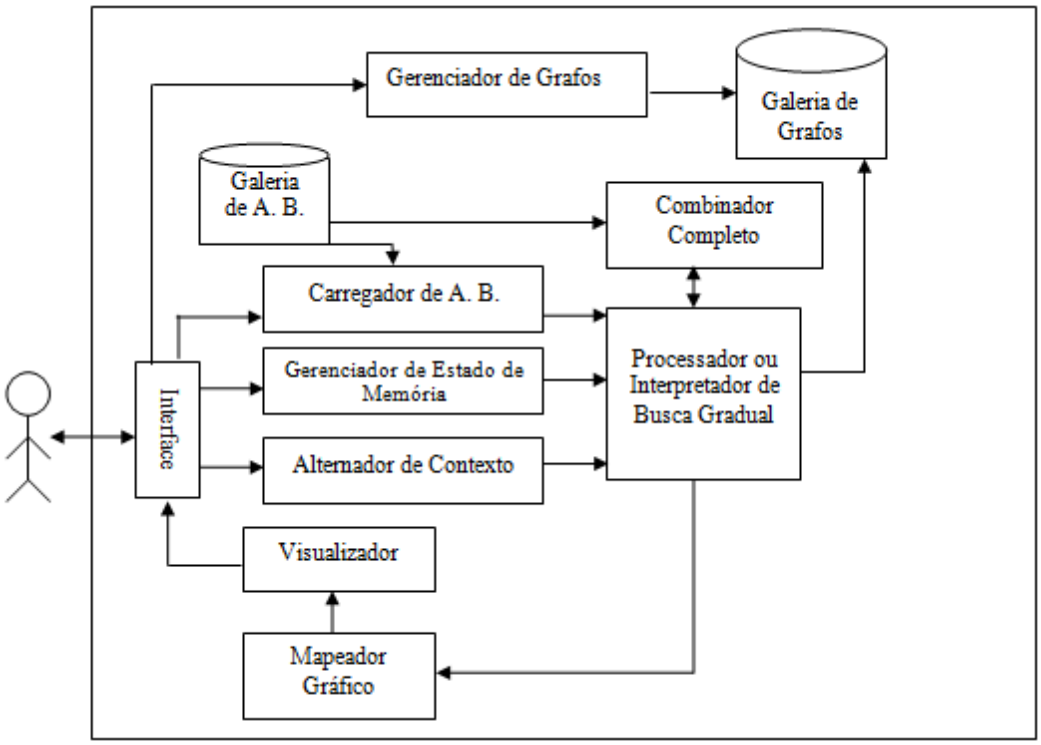


Figura 4.1: Arquitetura funcional do sistema

4.1.1 Gerenciador de grafos

O domínio de sub-problemas implementado na VIMAP como experimento inicial se baseou em grafos para expressar a transição entre estados da busca. Já o espaço da busca em si foi limitado às árvores (casos especiais de grafos). Portanto foi elaborado um módulo de autoria que permite uma interação entre o aprendiz/professor com a ferramenta. O professor pode cadastrar o grafo de transição de estados que deseja utilizar para demonstrar um ou mais exemplos de busca aos alunos, de acordo com a explicação dos conceitos em sala. Já para o aprendiz, quando for utilizar a ferramenta, ele pode realizar comparações entre o comportamento dos algoritmos de busca no grafo ou do mesmo algoritmo em diferentes grafos cadastrados.

Portanto, ao se cadastrar um grafo, cada nodo terá o valor heurístico h informado neste momento, tendo em vista que o trabalho não tem como objetivo central o desen-

volvimento de funções heurísticas. Além do valor heurístico, deverão ser cadastrados os nodos ascendentes (nodos pais) e o valor do custo g entre eles.

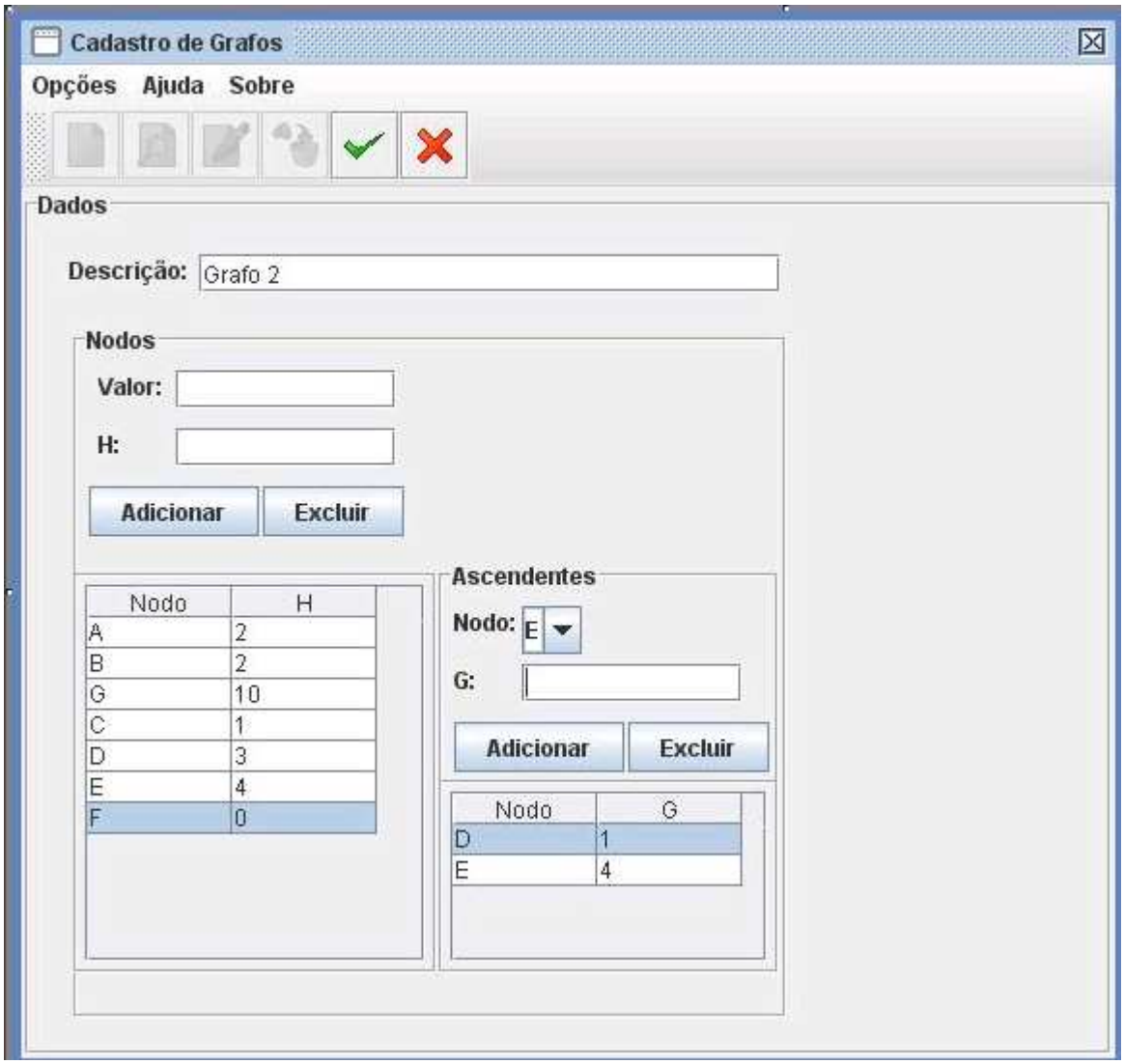
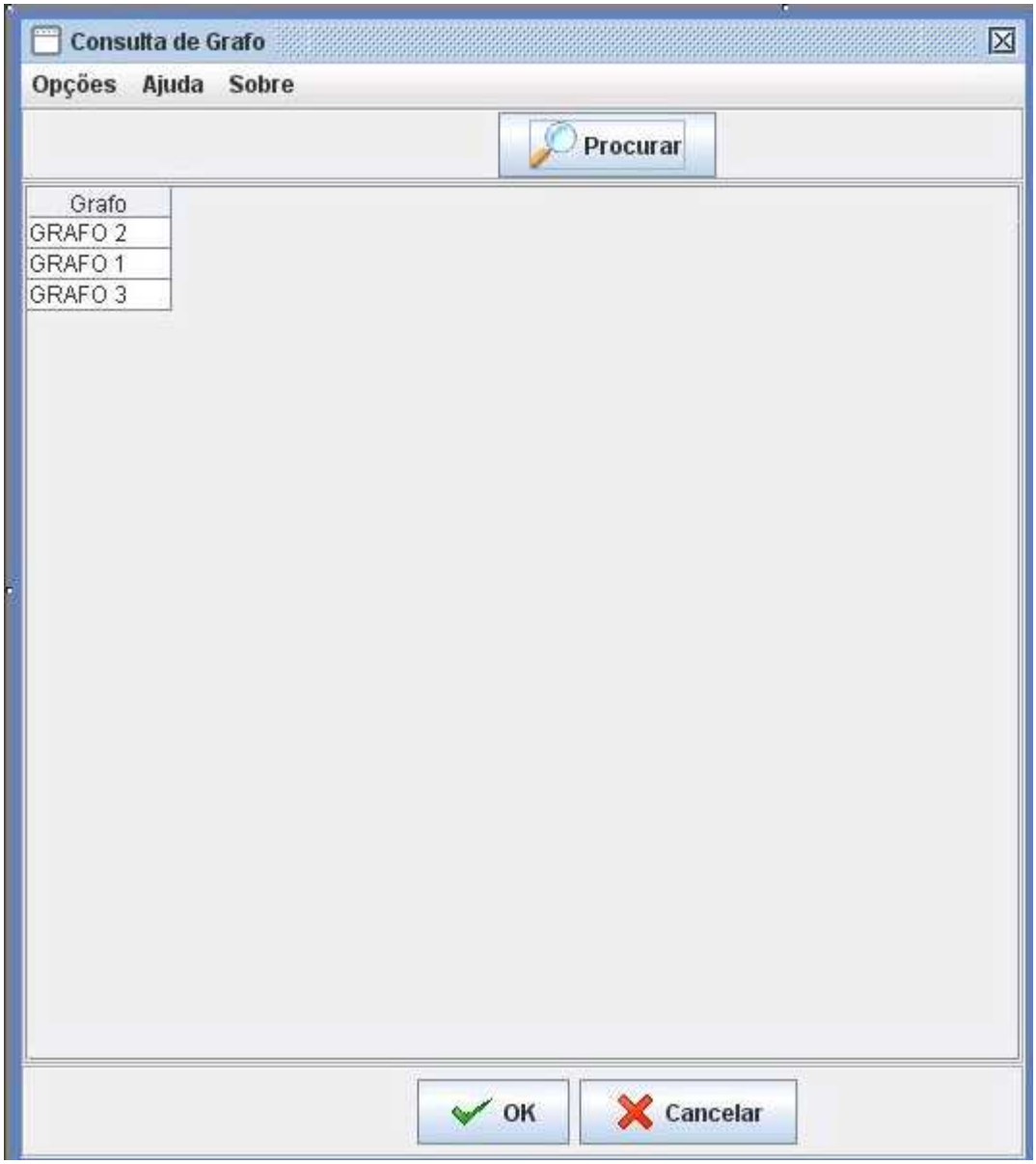


Figura 4.2: Interface de cadastro dos grafos

A figura 4.2 mostra a interface de cadastro desse módulo. Além da interface de cadastro tem a tela de consulta que é mostrada na figura 4.3, que permite ao aprendiz/professor selecionar o grafo desejado e alterá-lo. Pode inserir novos nodos ou excluir aos nodos existentes.

4.1.2 Galeria de grafos

Ao cadastrar um grafo, este fica armazenado na galeria a qual poderá ter a quantidade de grafos desejada. Pois assim o professor/aprendiz irá cadastrar um ou mais grafos para



[70]

Figura 4.3: Interface de consulta dos grafos

realizar a visualização do uso de memória na execução do algoritmo.

Para realizar a persistência[3] foi utilizado o conceito de prevalência de objetos. Na prevalência o armazenamento principal dos objetos é feito na memória principal do computador, a qual é um meio físico volátil. No entanto, a prevalência garante a persistência dos objetos após várias execuções do sistema através da utilização dos mecanismos de *log*

[3]Segundo Carvalho[36] é a propriedade em que um objeto continua a existir, mesmo quando a aplicação que o criou terminar a execução

de transação[4] e de backup[5]. Desta forma, garante-se que, mesmo se o sistema for desligado ou parar de executar, nenhum objeto armazenado em memória será perdido, pois os dois mecanismos salvam o estado de um ou mais objetos em um meio físico não-volátil (disco), podendo restaurar o seu valor caso alguma falha ocorra.

As vantagens em utilizar esse tipo de armazenamento são:

- Independência de outros aplicativos, tais como, banco de dados, para execução do ferramenta, sendo necessário somente a JVM[6] para sua execução;
- Representação única da solução, pois em um sistema prevalente a solução é modelada e mantida em um único modelo de dados, o da própria aplicação;
- A prevalência de objetos é muito simples de usar e não sobrecarrega o programador;
- O custo da licença de SGBD supera o custo de adquirir memória RAM.

No desenvolvimento da ferramenta foi utilizado o *Prevayler* , o qual é a implementação do conceito de prevalência utilizando a linguagem de programação java. Esta foi utilizada no desenvolvimento da ferramenta.

Para representar os objetos persistentes da ferramenta VIMAP foram necessárias o desenvolvimento de três classes, apresentadas no diagrama a seguir. O diagrama de classe parcial de acordo com a figura 4.4.

4.1.3 Carregador de algoritmo de busca

Este módulo faz a comunicação entre a interface e a galeria de algoritmos de busca heurística. Após o aprendiz selecionar as opções iniciais por meio da interface, o módulo verifica e busca na galeria carregando-o na memória junto com suas informações necessárias.

[4]O log de transação é implementado pela utilização de classes que representam as alterações em objetos, serializando e salvando-as em um meio físico não-volátil.

[5]a forma de backup utilizada é o *snapshot*, responsável por salvar o estado de todos os objetos da memória principal para o disco, tirando uma espécie de 'fotografia' do estado dos objetos naquele dado momento.

[6]Java Virtual Machine ou seja Maquina Virtual Java

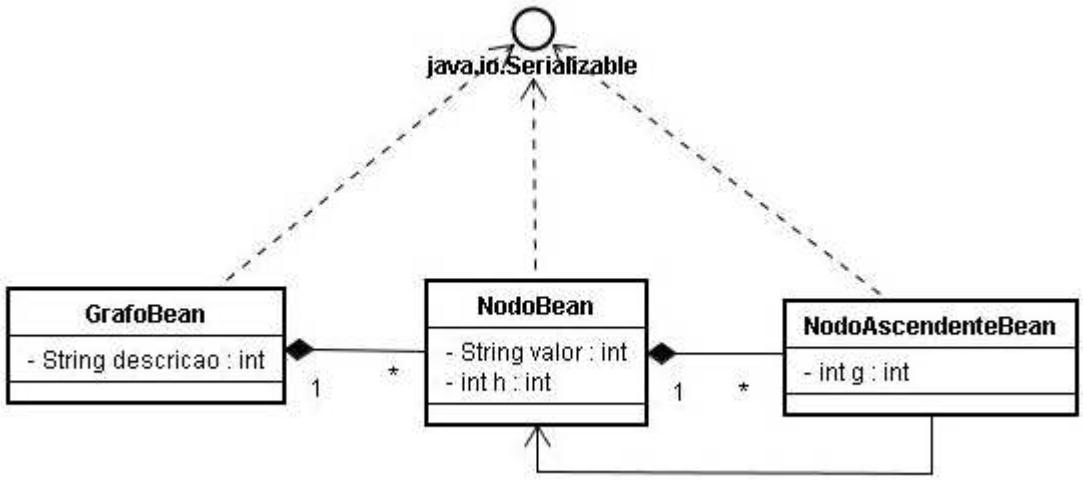


Figura 4.4: Diagrama de classe parcial

Para carregar um A.B.H. como o A*, IDA*, SMA* ou Best First, são necessários os seguintes elementos de acompanhamento para que o algoritmo encontre um plano como solução do problema:

- Ei (estado inicial) - nodo de onde partirá a busca a partir do estado com o valor que o aprendiz selecionar por meio da interface;
- Ef (estado final) - nodo que representa a terminação da trajetória (ou plano) e que, semelhante ao Ei, tem seu valor de estado selecionado pelo aprendiz;
- Regras de transformações atômicas - todas as transformações possíveis que um estado pode sofrer para gerar os descendentes.
- Função heurística - função que dá a estimativa de custo entre um determinado estado corrente (Ec) e o estado solução (Ef).

Já para um A.B.H. como os das classes de Satisfação de Restrições, SSS* ou *Poda* α–β, onde solucionar um problema se traduz em encontrar um estado final (Ef) que atende a certas condições, os seguintes elementos de acompanhamento devem ser carregados junto com o algoritmo:

- Ei (estado inicial) - apenas ele como estado pois os vários possíveis estados finais (Ef) que constam no espaço de busca são considerados solução de um problema;

- Regras de transformações atômicas;
- Função heurística.

Uma arquitetura verdadeiramente genérica precisa lidar com tais diferenças. Todavia, o carregador implementado no protótipo atual engloba apenas os algoritmos de busca capazes de encontrar um plano ou trajetória como solução de problema. O protótipo se encontra em um estágio ainda um tanto quanto primitivo porque sua estrutura está muito próxima do próprio algoritmo $A*$ e do gerenciador de estados de memória. Entretanto ela atende à proposta desse trabalho de atingir a visualização apenas do espaço de busca alocado por árvores OU.

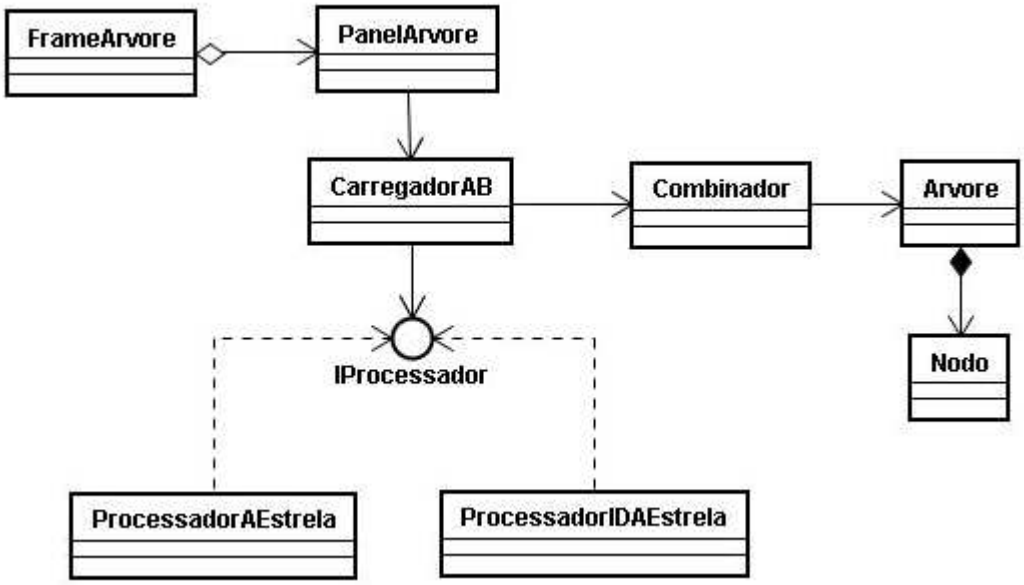


Figura 4.5: Diagrama geral de classes

No diagrama Geral de Classe da figura 4.5 da ferramenta de ensino, pode-se observar e entender melhor as explicação de cada módulo.

4.1.4 Galeria de algoritmos de busca

Outra galeria é a base de algoritmos de busca heurística, que corresponde ao conjunto de classes que em Java são conhecidos como *Beans*. Algumas destas classes foram projetadas para serem utilizadas por outros algoritmos de busca. Portanto, isso permitirá ao programador implementar um novo algoritmo de forma simples.

No protótipo foram implementados dois algoritmos de busca em grafos OU, são eles:

- $A*$ - o qual foi detalhado no capitulo 3.
- $IDA*$ - um algoritmo baseado no A* com diferença no aspecto de uso da memória. Pois este algoritmo desaloca memória e a quantidade de alocações é limitada por um valor de f que, aumentando de forma iterativa, ao ser executado, é possível visualizar a diferença entre os algoritmos.

Foram selecionados esses dois algoritmos de busca heurística porque são bastante conhecidos pela comunidade de Ciências da Computação e são excelentes exemplos didáticos. A galeria de exemplos pode ser ampliada para a quantidade de algoritmos desejada. Entretanto é necessário um conhecimento mais aprofundado de programação e da linguagem Java para implementá-los. A VIMAP ainda não tem uma ferramenta de autoria para o aprendiz/professor editar um novo algoritmo sem estes conhecimentos específicos.

4.1.5 Gerenciador de estado de memória

Pode ser considerado o módulo mais importante, porque trata o foco principal do trabalho. Pois este é responsável por gerenciar e controlar o estado de cada nodo da árvore. Controlando internamente os estados do nodo: se está alocado, se está na memória, se foi desalocado, ou realocado.

Na implementação está bastante ligado ao processador de busca gradual e ao próprio algoritmo. Para isto foi necessário implementar um atributo na classe nodo, e por meio deste é possível controlar o estado de memória do nodo, e assim durante a execução do algoritmo altera-se o valor deste atributo a medida que o nodo muda de estado.

Além do atributo mencionado, foram necessários vários métodos, de acordo com o algoritmo, para gerenciar essas mudanças. Estes métodos foram definidos na interface IProcessador, e implementados nas classe específicas de acordo com cada algoritmo. E todo este arcabouço também poderá ser preparado para problemas que encontram o estado solução ou invés da trajetória.

4.1.6 Processador de busca gradual

Definido o contexto (o grafo, o estado inicial e o estado final), o sistema possibilita ao aprendiz duas formas para visualização da execução do algoritmo, sendo que o aprendiz pode solicitar a execução automática do sistema, onde é efetuada uma iteração a cada 2 segundos ou a execução passo-a-passo, sendo que nesta opção o aprendiz tem a possibilidade de voltar caso não tenha entendido a decisão tomada pelo algoritmo. Na opção de execução automática o sistema possibilita ao aprendiz parar a execução automática a qualquer momento e continuar executando passo-a-passo, o mesmo ocorre na execução passo-a-passo possibilitando que seja solicitado a execução automática a qualquer momento.

O módulo responsável por esta funcionalidade no sistema é o processador de busca gradual, o qual tem como responsabilidade acionar a execução do algoritmo no contexto previamente definido pelo aprendiz e manter os dados necessários para transmitir ao mapeador gráfico, disponibilizando as informações necessárias à exibição de cada passo na interface de visualização da árvore.

Foi implementada uma Interface[7] com a definição dos métodos que serão comuns aos algoritmos. Além dessa Interface foram implementadas as classes propriamente ditas, sendo implementada uma classe para cada A.B.H. Esta funcionalidade está muito ligada ao gerenciador de estado de memória, que foi implementado nestas classes. Os dois métodos de grande importância da Interface que são executados a cada ciclo do algoritmo:

- public void proximo() - executa o próximo passo do algoritmo selecionado;
- public void retornar() - volta um passo da execução;

Ao processador também cabe a tarefa de controlar os nodos folhas para poder enviar ao processador mapeador a ordem do melhor nodo com menor valor de f ao nodo com maior valor de f. Esta ordem será exibida ao aprendiz durante a execução e a cada ciclo de execução a ordem é alterada dinâmicamente à medida que os nodos são expandidos.

[7]Interface em Java é uma arquivo que contém a assinatura dos métodos que devem ser implementados pela classe que implementa a interface, sendo este uma das formas de implementação de polimorfismo em java

No momento a ferramenta abrange os algoritmos que encontram um plano como solução de problema. Para tratar os algoritmos de busca heurística que encontram um estado final como solução será necessário um tratamento mais específico nos processos internos de busca gradual.

4.1.7 Alternador de contexto

Este módulo permite ao aprendiz mudar o contexto da busca, de modo que ele possa simular várias execuções do algoritmo de busca. É possível alterar o estado inicial, o estado final, o grafo e o algoritmo de busca heurística. As várias execuções do algoritmo em contextos diferentes possibilitam ao aprendiz um melhor entendimento do funcionamento do algoritmo.

Após alterar o contexto de execução do algoritmo, ou seja, qualquer uma das opções já citadas, a ferramenta aciona o processador de busca gradual que, por sua vez, interage com o combinador que monta previamente, por completo, a nova árvore de busca antes mesmo de ela ser explorada pelo aprendiz (ver detalhes na Subseção "Tela de Execução"). Isso se faz necessário pois a imagem inicial da árvore completa precisa ser desenhada em cor pálida depois ser superposta gradualmente com cores mais vivas.

4.1.8 Mapeador gráfico

O mapeador recebe os fragmentos dos estados de memória do processador de busca gradual e monta quais as representações gráficas serão utilizadas conforme foram citadas no capítulo 3. Por exemplo, qual a cor que irá representar o caminho solução, quais nodos foram carregados na memória, etc.

O mapeador gráfico praticamente recebe um nodo como parâmetro, verifica seus estados de memória que já foram alterados pelo gerenciador de estado de memória e formata as suas opções gráficas. Os informações gráficas que são alteradas são:

- Verifica a situação do nodo e coloca a cor definida para aquele estado;
- Desenha o nodo de acordo com os parâmetros, altura, largura, posição (x,y);

- Desenha o nome do nodo;
- Desenha o valor de f;
- Desenha a aresta do nodo até o nodo ascendente;
- Desenha ao lado dos nodos folhas a ordem de acordo com o melhor nodo;

4.1.9 Visualizador

Este módulo recebe as informações definidas pelo mapeador gráfico e gera a interface de visualização da árvore de memória, ou seja, a interface do passo que o aprendiz deseja acompanhar.

Foi implementada uma classe especifica para exibir os elementos ao aprendiz chamada PanelArvore.java, basicamente a implementação utilizou os pacotes *swing* e *awt* atualmente incorporado à linguagem Java. Do pacote awt foi utilizado os elementos de cores, fontes, gráfico e eventos do *mouse* e do pacote swing os elementos JPanel e JOptionPane.

A classe PanelArvore interage diretamente com o mapeador gráfico para exibir cada avanço na execução. O método *paint* é de grande importância pois este é acionado no momento que o mapeador gráfico modifica as informações que devem ser alteradas na árvore visual. Este método é simples pois faz parte da classe JPanel do java, difere-se do método da classe pai porque este recebe informações especificas do mapeador gráfico a serem atualizadas.

4.1.10 Combinador completo

O combinador é um recurso de suma importância para a ferramenta dimensionar o tamanho da árvore, gerenciar a alocação de memória e sua representação. Para permitir ao aprendiz visualizar os ramos desta que ainda não foram percorridos será necessária uma execução desse algoritmo de forma oculta.

O classe Combinador possui três métodos para gerenciar suas atividades. Foram implementados dois métodos gerarArvore, caracterizando uma sobrecarga de função[8], e um

[8]mais de um método com o mesmo nome, porém o que o que os diferencia são os parâmetros

método gerar descendentes de acordo com as regras de transformação.

Portanto, para o combinador gerar a árvore inicial são executadas as seguintes funções:

- Aciona o processador de busca gradual que recupera o grafo cadastrado na galeria de grafos;
- Calcula a profundidade do estado inicial e do estado final, e a diferença de profundidade entre os dois estados será a profundidade da árvore, caso a profundidade seja muito grande é utilizado um valor fixo;
- Executa-se uma busca em profundidade a partir do nodo raiz gerando seus descendentes até a profundidade definida.

Assim, dependendo do tamanho do grafo o combinador monta a árvore completa ou parte dela, porem não se preocupa se o estado solução está na árvore. De acordo com os testes realizados não nos preocupamos com grafos muito grandes. Realizamos as buscas em grafos do tamanho dos que estão nos livros didáticos típicos.

4.1.11 Interface

A interface possibilita ao aprendiz interagir com a ferramenta, permitindo que essa receba informações, acione a execução de tarefas e exiba informações processadas. Além disso, possibilita que o aprendiz execute as ações de investigação desejadas, tais como: retornar um passo na execução do algoritmo, avançar um passo e, ainda, obter informações ao clicar sobre o nodo, pois esse *input* abrirá uma janela *pop up* com dados do nodo.

4.1.11.1 Menu de opções

A tela inicial do protótipo tem uma apresentação da ferramenta, ou seja o nome e função. Portanto, possui uma barra de menu que possibilita a execução de determinadas ações, o qual possibilita a execução da tarefa desejada, como cadastrar grafos e executar a busca, mas só poderá executar a busca se existir um grafo cadastrado.

Conforme a figura 4.6 a barra de menu também possuí alguns dos itens de interação entre o aprendiz e a ferramenta como é exibido na figura. Para não deixar a tela carregada

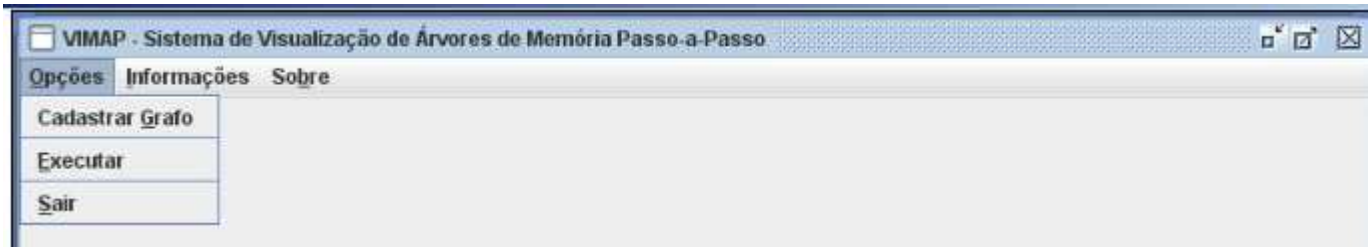


Figura 4.6: Barra de menu do sistema

de opções os menus são excelentes recursos para as operações mais genéricas. Os itens e subitens de menu são:

- Opções
  - Cadastrar Grafo
  - Executar
  - Sair
- Informações
- Sobre

4.1.11.2 Tela de execução

Nesta tela o aprendiz vai observar a execução do algoritmo de acordo com o estado de memória e como a árvore gráfica vai colorindo seu conteúdo. A figura 4.7 mostra como a árvore é gerada inicialmente, toda cinza indicando que o algoritmo não iniciou a execução e os nodos em cinza não estão na memória instantânea neste momento.

Logo a execução pode basicamente acontecer de duas formas: automática a qual o aprendiz apenas acompanha passivo a execução do algoritmo e este a cada 2 segundos avança um passo. Possui também a execução manual onde é necessário um clique do aprendiz para o executar o próximo passo ou retornar ao anterior. Ainda é possível ao aprendiz alternar durante a execução entre o modo automático e manual.

Cada nodo é sensível ao mouse, quando o aprendiz clica sobre o mesmo a ferramenta VIMAP abre uma janela de mensagem com as informações do nodo como mostra a figura

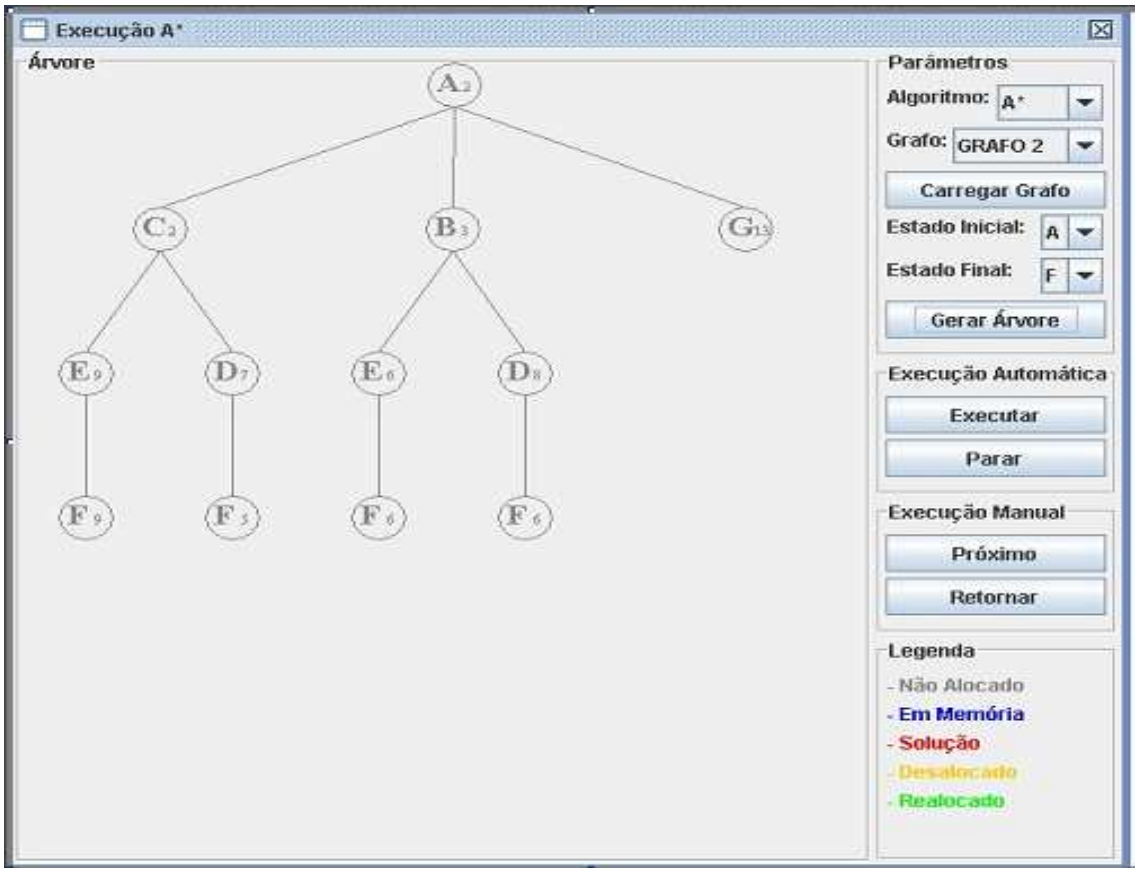


Figura 4.7: Visão inicial da árvore

4.8. Tais informações são importantes ao aprendiz para conferir as informações como o valor do custo g do nodo até o seu ascendente ou o valor estimado h até o estado final.

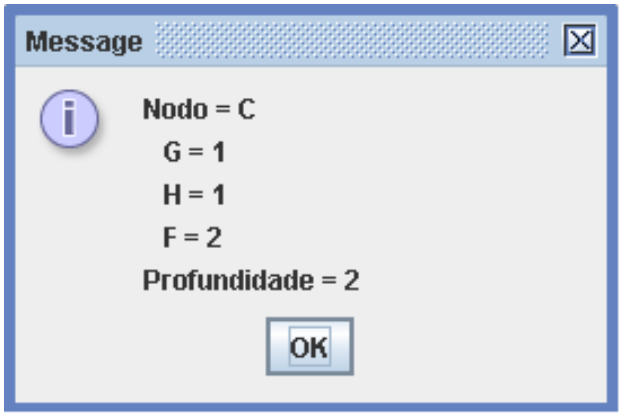


Figura 4.8: Informações do nodo

4.1.11.3 Botões de interação

Nesta barra estão alguns dos itens de interação entre o aprendiz e a ferramenta como é exibido na figura 4.9. Cada botão aciona uma tarefa de acordo com a seguência da figura:

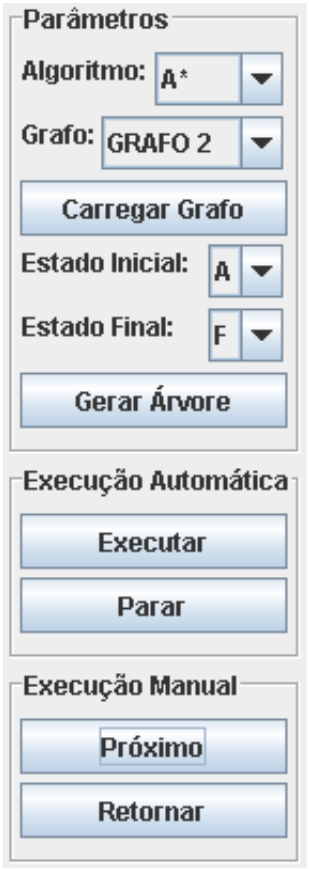


Figura 4.9: Elementos de interação

- Algoritmo - selecionar o algoritmo que deseja observar;
- Grafo - selecionar o Grafo, que foi previamente cadastrado;
- Carregar Grafo - carregar o grafo selecionado na memória carregando os nodos do grafo nas opções Estado Inicial e Estado Final;
- Estado Inicial - selecionar o estado inicial, um nodo do Grafo selecionado;
- Estado Final - selecionar o estado final;
- Gerar Árvore - exibir a árvore na cor cinza, nesse momento não tem nodo na memória instantânea pois não foi iniciada a execução do algoritmo;

- Execução Automática

  Executar - acionar o execução automática do algoritmo com tempo de 2 segundos a cada passo;

  Parar - parar a execução automática do algoritmo;

- Execução Manual

  - Próximo - avançar a execução um passo;

  - Retornar - retornar ao passo anterior.

CAPÍTULO 5

DISCUSSÃO SOBRE OS RESULTADOS

Nesse capítulo estão apresentados os resultados que se espera atingir com a utilização da ferramenta e ainda algumas vantagens e limitações da mesma.

5.1 Efetividade dos princípios e desenvolvimento de perícia

Embora o conteúdo de algoritmos de busca heurística seja ensinado em uma sala de aula, também pode ser encontrado facilmente em um livro de I.A. Porém, esse conteúdo não é tão simples de ser compreendido. Assim, muitas vezes o conhecimento adquirido é apenas decorado e não entendido de forma correta para ser aplicado em solução de problemas reais.

O sistema proposto permitirá que os aprendizes visualizem o funcionamento dos algoritmos conforme os princípios aprendidos em sala ou pelos livros. Ele também deverá facilitar o desenvolvimento de perícias de resolução de problemas complexos.

5.2 Facilidade de auto-estudo

Como já comentado anteriormente, a interface permite que o aprendiz realize a aprendizagem por investigação, pois no momento de suas dúvidas ele pode acompanhar passo-a-passo a execução de cada ciclo do algoritmo e ainda diversas maneiras de interagir com a ferramenta. Assim, o aluno exercita o conhecimento teórico utilizando-o na prática. Como incremento ao estudo dirigido dentro de sala de aula, a ferramenta possibilita a um aprendiz interessado conduzir um estudo por conta própria, de acordo com a sua necessidade, tendendo a dirimir as dúvidas.

Contudo, para garantir essa facilidade de auto-estudo, uma caracteristica importante da interface é ser intuitiva, de forma que o aprendiz não necessite ter conhecimentos

avançados de Informática para utilizar a ferramenta. A idéia é de que o aprendiz se preocupe apenas em aprender algoritmos de busca heurística. Pode-se observar os rótulos (*labels*) dos botões com palavras que deixam claro ao aprendiz a funcionalidade daquele elemento de interação.

5.3 Utilizar diferentes conceitos pedagógicos

Espera-se, com a utilização da VIMAP, que possa ser de grande proveito e maior interesse tanto de aprendizes como de professores o uso de diferentes métodos pedagógicos e ferramentas computacionais de apoio ao ensino. A utilização desses métodos poderá tornar possível a identificação de novos conceitos pedagógicos e confirmação de conceitos existentes que seguem uma abordagem construtivista, ou mesmo construcionista, como a da visão de S. Papert, o qual apregoa diversas vantagens por meio do computador.

5.4 Algumas limitações

Pode-se levantar algumas limitações do trabalho:

- Para se implementar outros algoritmos é necessário ter conhecimento de linguagem de programação Java, fator este que limita o uso da ferramenta pois não permite a um aprendiz ou professor que não detenha este conhecimento expandir a galeria de algoritmos de busca heurística;
- A ferramenta está preparada para tratar problemas que encontram trajetórias como solução, apesar do trabalho ja deixar explícito isso no inicio, portanto, para quem vai utilizar a ferramenta seria interessante envolver problemas que encontram um estado final.

5.5 Vantagens

Para o aprendiz as vantagens no processo de aprendizagem são várias. Pode-se destacar a facilidade de se colocar em prática o conhecimento adquirido com a execução do algoritmo.

Como a visão é um dos sentidos mais importantes do ser humano, a visualização da alocação de memória representada na estrutura de árvore irá proporcionar aos aprendizes entender o funcionamento dos algoritmos de busca por meio da percepção para a redução do esforço cognitivo.

Além da vantagem de se constituir em um veículo a mais de aprendizagem, a ferramenta oferece a vantagem técnica pois foi implementada na linguagem de programação java, sendo portanto independente de sistema operacional.

CAPÍTULO 6

CONCLUSÃO

6.1 Reafirmação da contribuição do trabalho

Este documento apresentou uma abordagem de utilização de representações visuais e elementos de interação para apoiar a aquisição de conhecimento pericial em algoritmos de busca heurística. Nesse contexto em que está inserida a ferramenta, nota-se a carência de sistemas tutores que se adaptam às particularidades de cada aprendiz. A partir desta constatação, este trabalho aprofunda de maneira sólida ao ensino de algoritmos de busca heurística por meio dos métodos de aprendizagem por visualização e por investigação.

Em particular, enfocou-se a resenha nos aspectos referentes à autoria de conteúdos inteligentes, consistência de ensino em sistemas tutores inteligentes, ensino de conceitos visuais, visualização computacional e múltiplas representações.

Sobre os conceitos utilizados na solução do problema, apresentou-se o potencial de micro-mundos de exploração, representação de árvore de memória, representações visuais e elementos de interação. Isso foi feito com os devidos critérios de utilização das representações visuais para que domínio fosse refletido de forma realista.

Após, descreveu-se cada módulo da arquitetura que dá suporte à abordagem deste trabalho. Durante a explicação desses módulos, existem detalhes de implementação da ferramenta. Com a construção da ferramenta de apoio ao aprendizado de algoritmos de busca heurística, por meio da visualização das árvores de memória, espera-se possibilitar ao aprendiz a passagem do nível de conhecimento dos princípios ao nível de perícia no que se refere às características de algoritmos de busca heurística. Os elementos de interação facilitam a adaptação do aprendiz com o ensino pois este pode acompanhar a execução de cada passo no tempo desejado.

Tem-se ciência de que os objetivos cumpridos têm uma contribuição relevante e que o presente trabalho oferece grande potencial, não obstante as limitações relatadas, para

enriquecer o ensino do domínio em questão. Contudo, tem-se ainda a necessidade de aplicar as conclusões de tais pesquisas no sentido de estabelecer um arcabouço por experiência que propicie a construção de tutores que se adaptam às necessidades individuais do aprendiz, e que esta dissertação seja apenas um ponto de partida neste estudo. Deste modo, pode-se explorar uma perspectiva possível de contribuição à área de ferramentas voltadas à educação.

6.2 Trabalhos futuros

O conhecimento gerado, com a efetivação do projeto proposto, contribuirá à continuidade da pesquisa referente à utilização da tecnologia desenvolvida pela Informática na educação, em vários pontos, a saber:

- O mais interessante deles seria a construção de um framework (arcabouço de visualização) para ser utilizado por outros domínios além do domínio de busca em grafos, de maneira que o aprendiz tenha a possibilidade de adaptar as representações visuais de acordo com o problema em questão. Além das representações visuais os módulos poderiam trabalhar com algoritmos de busca heurística cujo o objetivo é de encontrar o estado final, além de encontrar um plano.

- Utilização de múltiplas representações externas. Isso facilitará o aprendizado, pois será possível disponibilizar ao aprendiz, ao mesmo tempo, tanto uma janela que apresente a árvore de memória quanto outra onde possa ser vista a ação acontecer de forma gráfica, ou seja uma transformação atômica sendo executada. Outra representação poderia ser o próprio código fonte com destaques na linha executada naquele instante em conjunto com a representação em árvore. Podemos comparar algo desse tipo aos depuradores mas não se encontra nada referente à educação.

- Construção de ferramentas de autoria para a elaboração de instrumentos de ensino baseados em visualização que proporcione ao tutor humano criar sua ferramenta de ensino.

- Ampliação da galeria de algoritmos de busca da ferramenta para a visualização dos estados de memória na estrutura de gráfos.

- Utilização outras técnicas de visualização como por exemplo: representações 3D.

BIBLIOGRAFIA

[1] CAVA C. M. D. S. Freitas O. M. Chubachi P. R. G. Luzzardi R. A. Introdução à visualização de informações. *RITA*, VIII(2), 2001.

[2] DIRENE A. Designing intelligent systems for teaching visual concepts. *International Journal of Artificial Intelligence in Education*, 8:44–77, 1997.

[3] DIRENE A., SCOTT D. Identifying the component features of expertise in domains of complex visual recognition. *Information Technology Research Institute Technical Report Series. Univ. of Brighton, Lewes Road, Brighton, UK*, 2001.

[4] JUELL P. Shanmugasundaram V. Denton A. Effectiveness of visualizations for student use. *ED-MEDIA 2003: World Conference on Educational Multimedia, Hypermedia and Telecommunications*, Honolulu, Hawaii:23–28, 2003.

[5] ZHANG J. Norman D. A. Representations in distributed cognitive tasks. *Cognitive Science*, 18:87–122, 1994.

[6] SILVA A. L. da. Modelagem e criação de interface para uma proposta de ferramenta de autoria para a elaboração de conteúdos digitais. *tese*, 2004.

[7] LESGOLD A. M. Acquiring expertise. *Em Anderson, J. R. and Kosslyn, S. M., Tutorials in Learning and Memory: Essays in Honor of Gordon Bower*, W. H. Freeman:31–60, 1984.

[8] PIMENTEL A. R., DIRENE A. *Medidas cognitivas no ensino de programação de computadores com sistemas tutores inteligentes.* Tese de Doutorado, UFPR, março de 1998.

[9] DU BOULAY B., MIZOUGUCHI R. The simquest authoring system for simulation-based discovery learning. *Artificial Intelligence in Education*, IOS Pres:79–86, 1997.

[10] SHNEIDERMAN B. The eyes have it: a task by data type taxonomy for information visualizations. *Proceedings of IEEE Symposium on Visual Languages, Boulder, CO*, 3(6):336–343, 1996.

[11] WHITE B. Thinkertools: Causal models, conceptual change and science education. *Cognition and Instruction*, 10(1):1–100, 1993.

[12] WINN B. The psychology of illustration, capítulo charts, graphs and diagrams in educational materials. *Springer-Verlag*, páginas 152–198, 1987.

[13] LUZZI F. Ferreira R.Z. Senna R.C. L.M. Martins Giraffa R. Melo Bastos. Assistente inteligente para suporte ao ensino de química orgânica. *IV Congresso RIBIE, Brasilia*, 1998.

[14] JUELL P. Shanmugasundaram V. Hill C. Knowledge building using visualizations. *The 11th Annual Conference on Innovation and Technology in Computer Science Education. Bologna, Italy*, ITICSE-06:26–28, 2006.

[15] MORAES C. R. Estrutura de dados e algoritmos. *Berkeley - São Paulo*, 2001.

[16] PIMM D. Symbols and meanings in school mathematics. *London: Routledge*, 1995.

[17] PLASS J. L. Chun D. M. Mayer R. E. Leutner D. Supporting visual and verbal learning preferences in a second language multimedia learning environment. *Journal of Educational Psychology*, 90(1):25–36, 1998.

[18] SLEEMAN D. H. Pixie: a shell for developing intelligent tutoring systems. in lawler, r. e yazdani, m., editors. *AI and Education: Learning Environments and Intelligent Tutoring Systems. Ablex Publishing*, 1987.

[19] SHARPLES M. Jeffery N. Teather D. Teather B. du Boulay G. A socio cognitive engineering approach to development of a knowledge-based training system for neuroradiology. *In du Boulay, B. and Mizoguchi, R., editors, Artificial Intelligence in Education*, IOS Press:402–409, 1997.

[20] NASCIMENTO E. *Representação e Ferramentas para Apoiar a Programação de Dispositivos*. Tese de Doutorado, UFPR, março de 2000.

[21] WENGER E. Artificial intelligence and tutoring systems : Computacional and cognitive appoches to the communication of knowledge. Morgan Kaufmann Publishers, Los Altos, CA, 1987.

[22] LESGOLD A. M. Rubinson H. Feltovich P. Glaser R. Klopfer D. e Wang Y. Expertise in a complex skill: Diagnosing x-ray pictures. *Em Chi, M., Glasser, R. e Farr, M., editors, The Nature of Expertise*, Lawrence Erlbaum, 1989.

[23] SANTOS G., Direne A. e A.L.P. Guedes. Autoria e interpretação tutorial de soluções alternativas para promover o ensino de programação de computadores. *In XIV SBIE - Simpósio Brasileiro de Informática na Educação*, páginas 623–632, 2003.

[24] NWANA H. S. Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, 4:251–277, 1990.

[25] CHANDLER P. Sweller J. The split-attention effect as a factor in the design of instruction. *British Journal of Educational Psychology*, 62(2):233–246, 1992.

[26] KALYGUGA S. Chandler P. Sweller J. Levels of expertise and instructional design. *Human Factors*, 40(1):1–17, 1998.

[27] ANDERSON J. R. Acquisition of cognitive skill. *Psychological Review*, 89:369–403, 1982.

[28] SANTAELLA L., NÖTH W. Imagem: cognição, semiótica, mídia. *São Paulo: Iluminuras*, 2001.

[29] SHARPLES M. Computer-based tutoring of visual concepts: from novice to expert. *Journal of Computer Assisted Learning*, 7:123–132, 1991.

[30] YERUSHALMY M. Student perceptions of aspects of algebraic function using multiple representation software. *Journal of Computer Assisted Learning*, 7:42–57, 1991.

[31] CARMO M. B. Visualização de informação modelo integrado para o tratamento de filtragem e múltiplas representações. *tese*, 2003.

[32] MAJOR N. Using coca to build na intelligent tutoring system in simple algebra. *Intelligent Tutoring Media*, 2(3/4):159–169, 1991.

[33] MAJOR N. Redeem: Exploting symbiosis between psychology and authoring environments. *International Journal of Artificial Intelligence in Education*, 8:317–340, 1997.

[34] JUELL P. The visual program project. *32nd Annual Midwest Instruction and Computing Symposium*, University of Wisconsin - La Crosse, WI, 1999.

[35] NORVIG P. Russell S. Artificial intelligence: A modern approach. *Prentice Hall*, 1995.

[36] CARVALHO P.M. Prevalência - persistência tranparente de objetos. *CEFET*, Medianeira/PR:6–34, 2005.

[37] TABACHNECK H. J. M. Koedinger K. R. e Nathan M. J. Towards a theoretical account of strategy use and sense making in mathematical problem solving. *6th Annual Conference of the Cognitive Science Society - Hillsdale, NJ: LEA*, 1:836–841, 1994.

[38] LOWE R. K. Effects of interactive animation in complex visual learning. *In S. Vosniadou, K. Matsagouras, K. Maridaki-Kassotaki, and S. Kotsanis (Ed.) 7th European Conference for Research on Learning and Instruction Athens: Gutenberg University Publications*, páginas 181–182, 1997.

[39] MINGHIM R Oliveira M. C. F. de. Uma introdução à visualização computacional. *XVI JAI - Jornada de Atualização em Informática - Brasília-DF*, Prentice Hall:85–127, 1997.

[40] ROCHA H.V. da ROMANI L.A.S. O uso de técnicas de visualização de informação como subsidio à formação de comunidades de aprendizagem em ead. *WORK-*

SHOP SOBRE FATORES HUMANOS EM SISTEMAS COMPUTACIONAIS, Florianópolis. Anais UFSC/SBC(4):169–182, 2001.

[41] AINSWORTH S. Designing effective multi-representational learning environments. *Relatório Técnico 58, ESRC Centre for Research in Development, Instruction and Training University of Nottingham*, março, 1999.

[42] AINSWORTH S. The functions of multiple representations. *Computers and Education*, 33(2-3):131–152, 1999.

[43] AINSWORTH S. Deft: A conceptual framework for considering learning with multiple representations. *Learning and Instruction*, em publicação, 2006.

[44] HOROWITZ E. Sahni S. Rajasekaran S. Computer algorithms. *Computer Society Press*, 1998.

[45] PALMER S. E. Fundamental aspects of cognitive representation. *Cognition and categorization - Hillsdale, NJ: LEA*, páginas 259–303, 1978.

[46] AINSWORTH S. Labeke N. V. Using a multi representational design framework to develop and evaluate a dynamic simulation environment. *Dynamic Information and Visualisation Workshop*, julho, 2002.

[47] BLESSING S.B. A programming by demonstration authoring tool for model-tracing tutors. *International Journal of Artificial Intelligence in Education*, 8:233–261, 1997.

[48] LARKIN J. H. Simon H. A. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99, 1987.

[49] MURRAY T. Authoring knowledge based tutors: Tools for content, instructional strategy, student model and interface design. *Journal of the Learning Sciences*, 7(1):5–64, 1998.

[50] MURRAY T. Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10:98–129, 1999.

ANDRÉIA OLIVEIRA LIZARDO

FERRAMENTAS DE APOIO AO APRENDIZADO DE ALGORITMOS DE BUSCA HEURÍSTICA POR MEIO DA VISUALIZAÇÃO DAS ÁRVORES DE MEMÓRIA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Alexandre Ibrahim Direne

CURITIBA

2007