

Layouts Automáticos para Mapas Conceituais - Um serviço integrado a uma plataforma de serviços web

Raphael Nogueira de Castro
Univ. Federal Espírito Santo
Av. Fernando Ferrari, 514
UFES-DI
+55 27 40092061
raphaelncaastro@gmail.com

Wagner Andrade Perin
Univ. Federal Espírito Santo
Av. Fernando Ferrari, 514
UFES-DI
+55 27 40092061
wagnerperin@gmail.com

Davidson Cury
Univ. Federal Espírito Santo
Av. Fernando Ferrari, 514
UFES-DI
+55 27 40092817
dedecury@gmail.com

ABSTRACT

Concept maps are graphical tools for representing and organizing knowledge in two dimensions, using distributed concepts in such a way that the relations established between them are evident. To be evident, these relationships depend on aesthetic aspects which, in turn, have a strong influence on cognitive processes. This article explores the automatic generation of concept maps layouts_ highlighting the findings and learning collected in the course of their development. The results of implementations are also presented and analyzed, resulting in suggestions for further improvements.

RESUMEN

Los mapas conceptuales son herramientas gráficas para representar y organizar el conocimiento en dos dimensiones, utilizando conceptos distribuidos de tal manera que las relaciones que se establecen entre ellas son evidentes. Para ser claros, estas relaciones dependen de aspectos estéticos que, a su vez, tienen una fuerte influencia en los procesos cognitivos. Este artículo explora la generación automática de los diseños de mapas conceptuales, destacando los resultados y aprendizajes recogidos en el curso de su desarrollo. Los resultados de las implementaciones también se presentan y analizan, dando lugar a sugerencias para nuevos trabajos.

Categories and Subject Descriptors

K.3.1 [Computers and Education]: Computer Uses in Education.

General Terms

Algorithms.

Keywords

Concept maps, web service platform.

1. INTRODUÇÃO

Os mapas conceituais são ferramentas gráficas para representar e organizar o conhecimento em duas dimensões, utilizando conceitos distribuídos de tal forma que as relações estabelecidas entre eles estejam evidentes [1]. Eles têm sido utilizados nas mais diversas atividades pedagógicas que compõem o ensino-aprendizagem. De fato, suas aplicações vêm sendo exploradas e têm apresentado vantagens tanto no planejamento de atividades pedagógicas [2] como no acompanhamento e na avaliação da aprendizagem [3]. De igual maneira, eles têm sido usados como ferramenta para organização do conhecimento [4], seja qual for a área de conhecimento da aplicação. Mais recentemente começaram a ser utilizados como ferramentas para representação

de conhecimento de tal maneira que possam ser interpretados, ou processados, computacionalmente [5, 6].

Estamos também interessados nos aspectos estéticos dos mapas a serem gerados, uma vez que a beleza tem influência nos processos cognitivos. Jean Piaget afirma que todo o conhecimento humano tem seu ponto de partida na coordenação sensorio-motora, onde a criança, por meio da função motora e dos sentidos, explora o mundo circundante. De acordo com Piaget, estar presente no mundo envolve constantes impulsos sensoriais, onde as experiências adquiridas pelas interações sensoriais com o mundo objetivo, cria esquemas internos, via processos de adaptação, que representam o conhecimento que se tem do mundo. Se todo o nosso conhecimento do mundo é basicamente devido à experiência sensorial e de processos de reconhecimento, uma interpretação do estético como reconhecimento sensorial traria, como consequência, que todo o conhecimento é, portanto, estético [7].

Outro conceito importante que nos mobiliza é a legibilidade. Segundo Nielsen [8], um dos requisitos de usabilidade fundamentais para interfaces de sistemas computacionais é a eficiência de uso. Para ele, um sistema computacional deve ser ágil no que diz respeito ao uso de suas funções por parte dos usuários, pois isso impacta diretamente na rapidez do desenvolvimento de suas tarefas. Eficiência de uso também é aspecto destacado pela ISO 9241-11 [9] que padroniza aspectos ergonômicos de interação humano-computador. E eficiência está bastante ligada à legibilidade.

Além disso, a importância da geração automática de layouts pode ser comprovada por analisar ferramentas para edição de grafos, mapas e ontologias, tais como Graphviz [10], Pcknot [11] e WebDot [12] CmapTools [13] e Protégé [14]. No entanto, analisando a Figura 1 é possível perceber o desconforto causado ao usuário quando da visualização de um mapa gerado. Cobia ao usuário a responsabilidade de reposicionar cada elemento presente no mapa de forma a permitir uma visualização adequada dele.

A fim de facilitar a compreensão do problema explorado, as pesquisas realizadas e os resultados obtidos com as implementações, este artigo está organizado em mais três seções: a Seção 2 (Layouts para Mapas Conceituais) procura identificar, discutir e analisar os diferentes layouts para mapas conceituais bem como as abordagens algorítmicas existentes para suas implementações; a Seção 3 (Algoritmos Implementados) apresenta as soluções desenvolvidas e os resultados obtidos; finalmente, a Seção 4 (Conclusões e Trabalhos Futuros) discute os resultados obtidos, as limitações identificadas e os possíveis trabalhos futuros para o projeto.

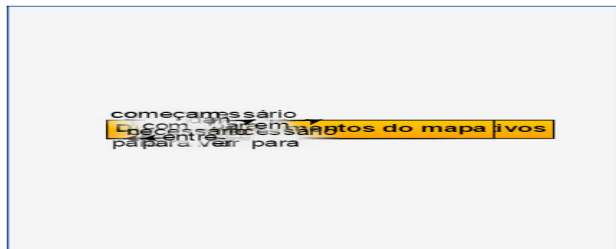


Figura 1 - Apresentação de um Mapa Conceitual sem coordenadas.

2. LAYOUT PARA MAPAS CONCEITUAIS

Novak [26] afirma que uma das características dos mapas conceituais é que os conceitos são representados de maneira hierárquica, com os conceitos mais inclusivos e gerais no topo e os mais específicos e menos gerais dispostos hierarquicamente abaixo. Entretanto, embora a hierarquia seja adotada, na maior parte das vezes, como layout estrutural de mapas conceituais, não existem regras fixas para a construção de mapas [27].

Concordamos com Tavares [20] quando afirma que o conhecimento sobre um determinado domínio pode ser organizado de diferentes formas, estando estas formas diretamente relacionadas à estrutura cognitiva do autor, ou seja, à forma como o conhecimento é internalizado por ele. Assim, Tavares classificou as seguintes formas gráficas de representação para mapas conceituais: **1. Layout Hierárquico:** Os conceitos são posicionados em ordem de importância, destacando os conceitos mais inclusivos (Figura 2). No layout hierárquico são traçadas linhas horizontais formando camadas. Os vértices são posicionados nessas camadas. As arestas somente ligam dois vértices pertencentes a camadas diferentes. Define-se aresta curta aquela que conecta dois vértices de camadas consecutivas; caso contrário, a aresta é definida como longa [24]. **2. Layout Teia de Aranha:** O conceito central e mais relevante do domínio é posicionado no centro do mapa com os demais ao seu redor (Figura 3).

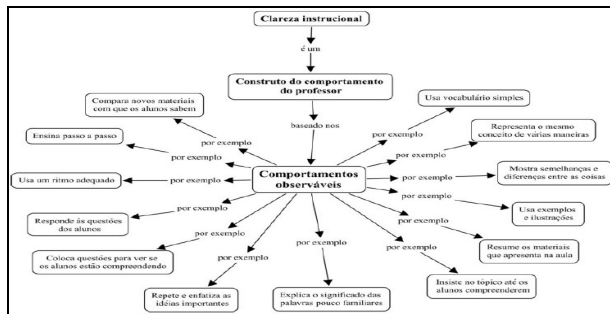


Figure 3. Exemplo de layout teia de aranha.

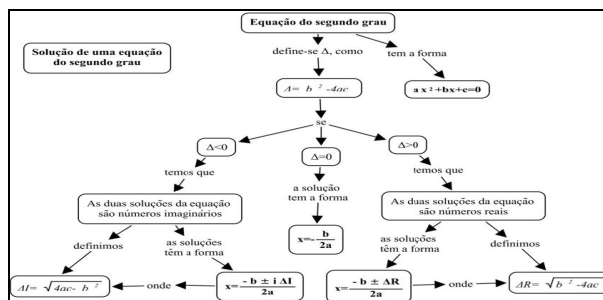


Figure 4. Exemplo de layout fluxograma.

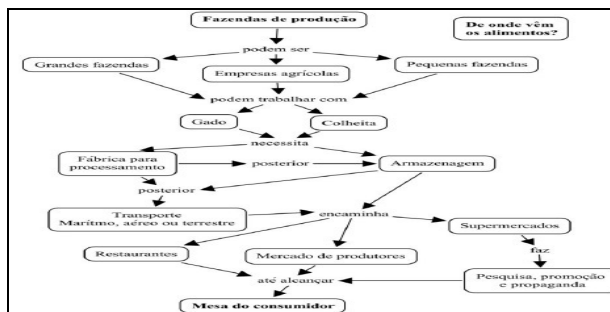


Figure 5. Exemplo de layout sistema.

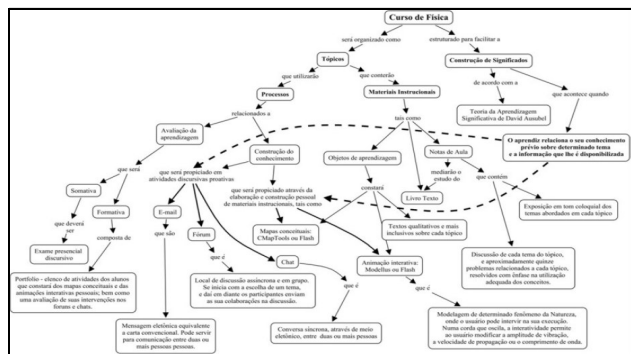


Figure 2. Exemplo de layout hierárquico.

3. Layout Fluxograma: Os conceitos são organizados de maneira linear e lógica, determinando um ponto inicial e final (Figura 4). Ele organiza a informação de uma maneira linear. Ele é utilizado para mostrar passo a passo determinado procedimento, e normalmente inclui um ponto inicial e outro ponto final. **4. Layout Sistema:** O layout é semelhante ao fluxograma, contemplando várias relações entre conceitos e impondo conceitos de entrada e saída (Figura 5).

Amoretti [19] também realiza uma classificação estrutural de mapas conceituais. No entanto, suas classificações, em essência, muito se assemelham aos propostos por Tavares. Realizando um cruzamento entre as definições de Novak [20] e Tavares percebemos que Novak faz referência aos layouts classificados por Tavares como Hierárquico e Teia de Aranha. Nesta pesquisa, selecionamos ambos layouts como clássicos e passíveis de implementação na ferramenta proposta. Na seção seguinte apresentaremos suas implementações.

3. CRITÉRIOS DE LEGIBILIDADE E ALGORITMOS DE IMPLEMENTAÇÃO

A implementação de um algoritmo para *layout* automático de mapas conceituais consiste na criação de uma rotina que determine a posição de cada um dos elementos presentes no mapa. Uma rotina simples poderia definir de forma aleatória o posicionamento dos elementos. Como vimos na seção anterior, no entanto, essa solução apresentaria resultados inadequados em se tratando de mapas conceituais. É necessário, portanto, a definição de critérios que determinem o posicionamento dos elementos que constituem o mapa conceitual. Nesta seção serão discutidos os critérios de visibilidade e o comportamento algorítmico para

implementação dos *layouts* hierárquico e teia de aranha. Nas buscas pelos algoritmos de geração automática de *layouts* consideramos os mapas como parentes próximos dos grafos, como feito por Lamas [21].

3.1 O Layout Hierárquico

De acordo com Ismael [22], os critérios estéticos de legibilidade para o *layout* hierárquico são: 1) Deve-se evitar o cruzamento de arestas. 2) Arestas com sentido ascendente devem ser evitadas. 3) Arestas longas devem ser evitadas quando possível. 4) Os vértices devem ser distribuídos uniformemente sobre suas camadas. 5) Os “falsos vértices” (utilizados pelo algoritmo) devem ser alinhados verticalmente. 6) A relação de aspecto da área de desenho deve ser razoável. 7) A área de desenho deve ser minimizada.

O algoritmo de Sugiyama *et al.* [23] é uma abordagem clássica para desenhos de grafos em layout hierárquico. Esse algoritmo pode ser resumido em quatro etapas principais que tornam seu desenvolvimento modular, já que cada etapa soluciona um problema específico e podem ser implementadas separadamente. Essas etapas são: **Etapa 1 (Remoção Ciclo):** É uma etapa opcional que consiste em inverter a direção de algumas arestas a fim de obter um grafo sem ciclos. **Etapa 2 (Atribuição de Camadas):** Consiste da atribuição de uma camada para cada vértice do grafo. **Etapa 3 (Minimização de Intercessão):** Consiste em determinar a ordem dos vértices nas camadas a fim de obter o menor número de interseção entre arestas. **Etapa 4 (Atribuição de coordenadas horizontais):** Consiste da atribuição do valor da coordenada horizontal dos vértices de cada camada.

3.2 O Layout Teia de Aranha

Dentre os critérios estéticos gerais de legibilidade para grafos apresentados por Ismael [22], identificamos como adequados para o *layout* teia de aranha: 1) Deve-se evitar o cruzamento de arestas. 2) A relação de aspecto da área de desenho deve ser razoável. 3) A área de desenho deve ser minimizada. 4) O vértice (conceito) central deve ser o que possuir mais arestas conectadas. 5) Os vértices devem ser distribuídos uniformemente sobre círculos ao redor do vértice central. 6) Único: o algoritmo deve retornar um único layout para cada grafo. 7) A arestas seguirão apenas uma direção (retas).

Para implementar este tipo de layout, desenvolvemos uma abordagem que consiste das seguintes etapas:

Etapa 1 (Ordenar Vértices): Consiste em ordenar decrescentemente os vértices pelo número de arestas que possui. **Etapa 2 (Atribuição de Círculos):** O vértice com mais arestas é eleito como central e todos os outros serão posicionados em círculos em torno do vértice central. **Etapa 3 (Atribuição de Ângulo e Posição):** Esta etapa recebe um grafo com os vértices divididos em um conjunto finito de círculos.

4. DA TECNOLOGIA USADA

O editor de mapas de nossa plataforma é uma página web. Seu conteúdo, portanto, é um código HTML. Para gerar o editor contido nesta página, usamos a biblioteca GoJS (<http://gojs.net/latest/index.html>). Para melhor entendimento, construímos o diagrama de classes para as classes utilizadas do GoJS pelo editor, mostrado na Figura 6. É possível perceber que para implementar um algoritmo para geração de *layout* utilizando a biblioteca do GoJS basta criar um código que determine qual valor assumirá o atributo *loc* de cada Node.

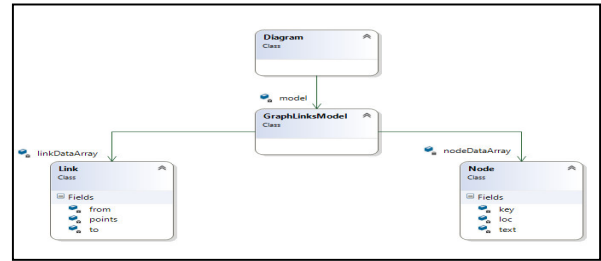


Figura 6. Diagrama de classes para o GoJS.

4.1 Adaptação do Algoritmo de Sugiyama

A implementação teve como base o algoritmo de Sugiyama. Quatro métodos principais foram derivados deste algoritmo, sendo estes denominados: GerarCamadas, InserirDummy, MinimizarIntersecao e DesfazerDummy. Entretanto, além destes métodos, foram implementados outros dois de menor complexidade algorítmica: TornarNodeMaisLinksPrimeiro e ZerarLink. A ordem de execução dos métodos é mostrada a seguir: 1) TornarNodeMaisLinksPrimeiro; 2) ZerarLink; 3) GerarCamadas; 4) InserirDummy; 5) MinimizarIntersecao; 6) DesfazerDummy. A seguir serão detalhados os métodos principais.

4.1.1 Método GerarCamadas

Este método tem por finalidade atribuir a cada Node uma camada.

O primeiro passo deste método consiste em atribuir o primeiro Node encontrado à camada zero. Uma vez que os Nodes foram previamente ordenados decrescentemente pelo número de Links, pelo método TornarNodeMaisLinksPrimeiro, o Node com mais ligações se posicionará no topo do mapa. Ao realizar este passo, implicitamente, este Node será incluído no grupo de Nodes com camada já calculada.

O passo seguinte consiste em percorrer os Nodes restantes calculando a camada de cada. Assume-se que o nodeDataArray tenha *n* Nodes e *i* seja a ordem dos Nodes restantes de tal forma que $2 < i \leq n$. Para determinar a camada que o Node de ordem *i* pertence, procura-se o Node já incluído no grupo de Nodes com camada já calculada e de maior camada ligado ao Node de ordem *i*. Caso exista, atribui-se o valor da camada do Node encontrado incrementado de 1 à camada do Node de ordem *i*. Caso contrário o Node *i* irá para camada zero. Para melhor ilustrar o funcionamento considere a Figura 7 a seguir.

No exemplo da Figura 7, o resultado da ordenação realizada pelo método TornarNodeMaisLinksPrimeiro será: Node 3, Node 2, Node 4, Node 1, Node 5, Node 6. Portanto, no primeiro passo, determina-se que o Node 3 pertence à camada zero. Implicitamente, o Node 3 também é incluído no grupo de Nodes com camada já calculada. Percorre-se então os restantes dos Nodes. O próximo elemento do *array* será o Node 2. Neste momento apenas um Node faz parte do grupo de Nodes com camada já calculada, o Node 3. O Node 2 é ligado ao Node 3 e portanto, será atribuído à ele a camada 1, uma vez que $0+1 = 1$. Implicitamente, o Node 2 também é incluído no grupo de Nodes com camada já calculada. O próximo elemento do *array* será o Node 4. O Node 4 possui ligação com o Node 3 e 2 de camadas 0 e 1, respectivamente. Portanto, o Node de maior camada já calculada até este passo, e ligado ao Node 4, é o Node 2. Logo, será atribuída a camada 2 para o Node 4. Para o restante dos Nodes (1, 5 e 6) será atribuída a camada 1, já que são apenas ligados ao Node 3 de camada zero. O resultado final será: camada zero (Node 3); camada 1 (Nodes 2, 1, 5, 6); camada 2 (Node 4).

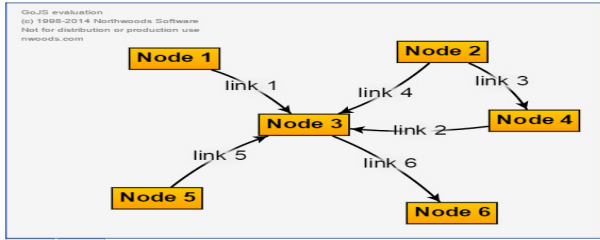


Figura 7 - Exemplo de Mapa.

4.1.2 Método InsereDummy

Este método substitui todas as arestas longas, isto é, Links que ligam camadas não adjacentes, por falsos Links curtos e falsos Nodes. Estes falsos Links e Nodes são também chamados de dummy pelo algoritmo de Sugiyama.

No exemplo da Figura 7, o link 2 conecta os Nodes 3 e 4 de camadas zero e 2, respectivamente. Este link será substituído por dois dummy Links e por um dummy Node. O dummy Node será posicionado na camada 1 e os dois dummy Links serão responsáveis por conectar o Node 3 ao dummy Node e o dummy Node ao Node 4. Estes dummy Nodes e Links são temporários e, portanto, não são visualizados pelo usuário. A figura 8 ilustra esta substituição temporária.

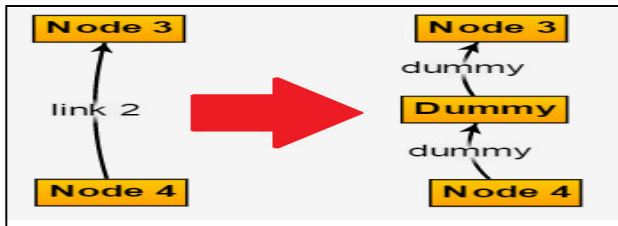


Figura 8. Substituição de Links longos por dummy.

4.1.3 Método MinimizaIntersecao

A função principal deste método é determinar a posição horizontal dos Nodes de forma que exista o menor número de cruzamento de arestas possíveis. No passo inicial determina-se a posição horizontal dos Nodes da camada zero. O primeiro node encontrado nessa camada terá coordenada horizontal de valor 0. As posições horizontais dos próximos Nodes da camada zero são calculadas deslocando o Node por uma distância igual ao produto da quantidade de letras do Node anterior adjacente por uma constante. Esta última forma de cálculo não foi escolhida, pois não foi encontrado até o momento uma função que retorne o tamanho real ocupado pelo Node.

A coordenada horizontal dos nodes das próximas camadas está diretamente relacionada com a posição dos nodes da camada superior. A forma de cálculo escolhida foi a do Baricentro. Portanto, a posição de um node de camada c será a média da posição dos nodes conectados da camada c -1. Por exemplo, suponhamos que o Node X esteja na camada dois e que esteja ligado a três outros nodes (Y, W, Z), da camada 1 de coordenadas horizontais 0, 40 e 140, respectivamente. A média destes valores será 60, a nova coordenada horizontal do node X. A Figura 9 ilustra este exemplo.

Contudo, ao calcular o valor da coordenada horizontal pelo Baricentro, pode-se gerar, para Nodes de mesma camada, valores de coordenada horizontal iguais ou muito próximos. Isto resultaria na sobreposição dos elementos.

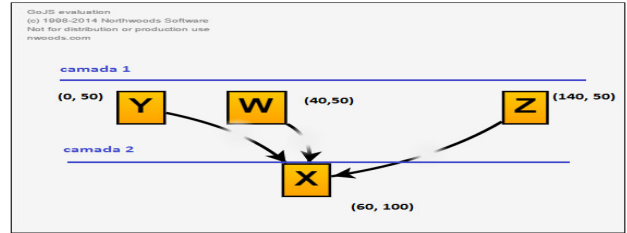


Figura 9. Cálculo da coordenada horizontal pelo Baricentro.

Para resolver o problema foi implementado um método adicional, denominado de AjustePosicao, que verifica se o valor gerado pelo cálculo do Baricentro resultará em sobreposição, e se for o caso, ajusta o valor deslocando para direita ou para esquerda. Para ilustrar como funciona este método veja a Figura 10 que representa um mapa conceitual simples.

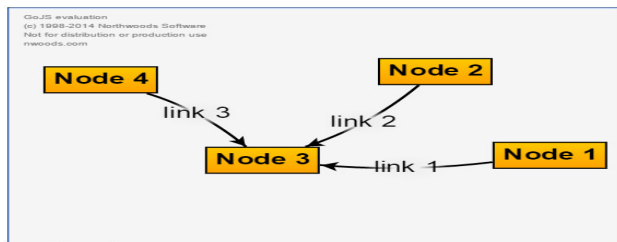


Figura 10. Cálculo da coordenada horizontal pelo Baricentro.

Na execução da ordenação pelo número de links o resultado é (3,2,4,1). O Node 3 pertencerá ao topo (camada zero). Os outros Nodes (2, 4, 1) ficarão na camada 1. Então, o Node 3 assumirá posição final (0, 0). O próximo elemento, Node 2, ligado apenas ao node 3, devido ao cálculo do Baricentro, assumirá a mesma coordenada horizontal, zero. Pelo mesmo raciocínio, os Nodes 4 e 1 também assumiriam coordenada horizontal igual a zero. Entretanto, devido ao método AjustePosicao este são deslocados um para direita, outro para esquerda. O valor desse ajuste é dado por uma constante. A Figura 11 ilustra a execução do método.

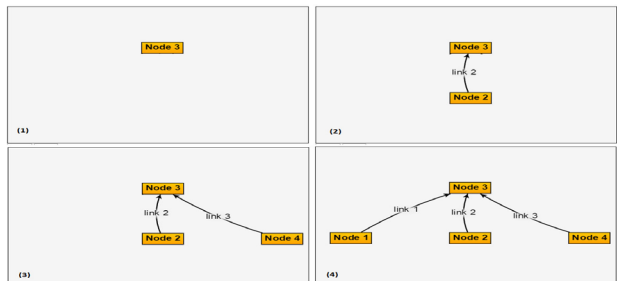


Figura 11. Cálculo da coordenada horizontal pelo Baricentro.

4.1.3 Método DesfazDummy

O método DesfazDummy substitui os dummy nodes e dummy links pelos links originais.

4.2 Algoritmo para o Layout Teia de Aranha

Ao contrário da implementação para layout hierárquico, a implementação do layout Teia de Aranha teve nossa solução. A implementação do algoritmo foi dividida em três métodos principais: TornarNodeMaisLinksPrimeiro, GerarRaios e GerarPosicao que correspondem respectivamente as etapas

Ordenar Vértices, Atribuição de Círculos e Atribuição de ângulo e Posição.

4.2.1 Método Gerar Raios

O método Gerar Raios é fiel à descrição da etapa Atribuição de Círculos dada na Seção 5.2. Ele é muito similar ao método Gerar Camadas do Algoritmo Hierárquico. Uma diferença é que no algoritmo Teia de Aranha apenas um Node será considerado central.

4.2.2 Método Gerar Posicao

O método Gerar Posicao é fiel à descrição dada na etapa Atribuição de Ângulo e Posição da Seção 5.2. O Node central assumirá a posição de coordenada (0,0). Os nodes do próximo círculo serão distribuídos uniformemente em um raio de valor definido por uma constante (R). Para gerar as coordenadas finais do primeiro círculo é utilizada a fórmula apresentada na Figura 12.

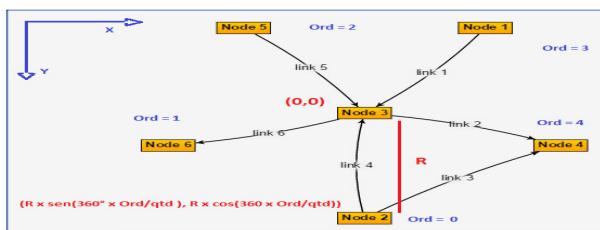


Figura 12. Ângulo e posição final em função da ordem do vértice.

Observações: Ord representa a ordem do Node na camada; qtd é a quantidade de Nodes na camada; R é uma constante que determina o raio do círculo; os valores do eixo Y crescem para baixo; $(R \times \text{sen}(360^\circ \times \text{Ord}/\text{qtd}), R \times \text{cos}(360^\circ \times \text{Ord}/\text{qtd}))$ é a fórmula para calcular a posição final dos Nodes do primeiro círculo. Para os próximos círculos o cálculo do ângulo assemelha-se ao do Baricentro do Algoritmo Hierárquico. O ângulo do node será a média resultante dos ângulos de todos os nodes ligados da camada anterior. A Figura 13 ilustra este exemplo.

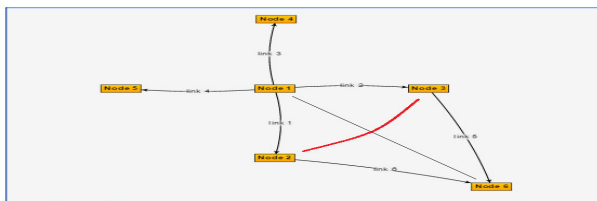


Figura 13. Cálculo do ângulo médio.

4.3 Resultados Obtidos

Apresentaremos os resultados das implementações dos algoritmos para geração automática dos layouts Hierárquico e Teia de Aranha discutidos na seção anterior. Para testes de efetividade destas funcionalidades, elaboramos um mapa conceitual utilizando o editor de mapas da nossa plataforma.

A Figura 1, já exibida na Seção 1, apresentou o estado inicial do mapa utilizado nos testes. As Figuras 14 e 15 apresentam os resultados gerados pela aplicação das abordagens discutidas nas Seções 3.1 e 3.2, respectivamente. Como é possível observar, ambos algoritmos implementados apresentaram resultados satisfatórios em seu comportamento no que diz respeito a facilitar a leitura de mapas.

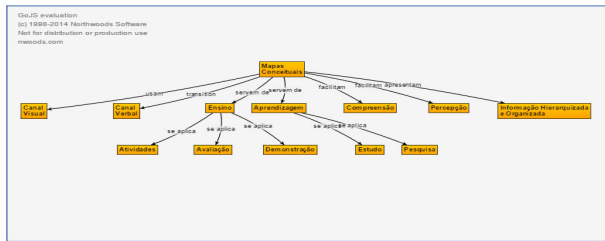


Figure 14. Mapa Gerado Automaticamente (Hierárquico).



Figure 17. Mapa Gerado Automaticamente (Teia de Aranha).

5. ALGUMAS CONCLUSÕES

Este artigo explorou uma limitação apresentada pelo editor de mapas da nossa plataforma. Seu objetivo foi implementar algoritmos para geração automática de layouts para os mapas conceituais. Para isso, foi realizada uma pesquisa bibliográfica que apresentou as abordagens computacionais com essa finalidade. Apresentamos, discutimos e implementamos duas das abordagens investigadas.

Além dos testes apresentados na Seção 4, realizamos outros testes para comprovar a efetividade dos algoritmos apresentados. Percebemos que para mapas conceituais de grande porte, com relações que ultrapassam diversas camadas, ambos geradores automáticos ainda apresentam cruzamento de arestas, como pode ser visto na Figura 18.

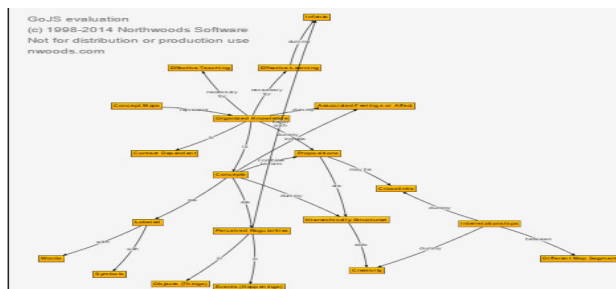


Figure 18. Mapa com cruzamento de arestas (Teia de Aranha).

Apesar de considerarmos a intervenção humana após utilização destas funções fundamental para o processo de personalização do mapa conceitual, os trabalhos futuros deste projeto incluem investigar os aprimoramentos que os algoritmos necessitam a fim de minimizar os cruzamentos de arestas. Não descartamos também a implementação de outras abordagens computacionais para geração automática de mapas conceituais, incluindo o de Fluxograma e de Sistema, discutidos na Seção 3 deste artigo.

AGRADECIMENTOS

Nossos agradecimentos à nossa mestranda Camila Zacche por sua ajuda importante na sugestão, compreensão e implementação de alguns algoritmos.

REFERÊNCIAS

- [1] Novak, Joseph D.; Gowin, D. Bob. 1984. Aprender a Aprender. 1. ed. Lisboa: Plátano Edições Técnicas.
- [2] Hartmann, A. M., Werlang, R. B., Carminatti, M., Balladares, A. L., Wagner, C., Suart, J. B. S. Jr. 2011. O uso de mapas conceituais no planejamento de projetos investigativos para Feiras de Ciências. In: Anais do VIII ENPEC. Campinas – SP, Brazil.
- [3] González, F.M., Moron, C., & Novak, J.D. 2001. Errores Conceptuales: Diagnosis, Tratamiento y Reflexiones. Pamplona, Spain: Ediciones Eunate.
- [4] Moon B. M., Hoffman R. R., Eskridge T.C., Coffey J. W. 2011. Skills in applied concept mapping. In: Moon B. M., Hoffman R. R., Novak J. D., Cañas A. J. (eds) Applied concept mapping: Capturing, analyzing, and organizing knowledge. CRC Press, Boca Raton, FL, pp 23-46.
- [5] Cañas, A. J., J. D. Novak, 2008. Facilitating the Adoption of Concept Mapping Using CmapTools to Enhance Meaningful Learning. In: Knowledge Cartography: Software Tools and Mapping Techniques, ed. A. L. P. Okada et al., Springer Verlag.
- [6] Cañas, A. J., Hill, G., Carff, R., Suri, N., Lott, J., Eskridge, T., Gómez, G., Arroyo, M., Carvajal, R. 2004. "CmapTools: A Knowledge Modeling and Sharing Environment". In: Concept Maps: Theory, Methodology, Technology, Proceedings of the First International Conference on Concept Mapping, A.J. Cañas, J.D. Novak, and F.M. González, Editors , Universidad Pública de Navarra: Pamplona, Spain. p. 125-133.
- [7] Austring, B.D., Sørensen, M. 2012. A Scandinavian View on the Aesthetics as a Learning Media. Journal of Modern Education Review, ISSN 2155-7993, USA.
- [8] Nielsen, J. 1993. Usability Engineering. Academic Press, Cambridge, MA.
- [9] ISO 9241-11, 1998. Ergonomic Requirements for office work with visual display terminals. Genève, v. 11, 5 ed..
- [10] Richardson, R., Fox, E. A. 2007. Using concept maps in NDLTD as a cross-language summarization tool for computing-related ETDs. In Proceedings of 10th International Symposium on Electronic Theses and Dissertations, Uppsala, Sweden (pp. 1-8).
- [11] Clariana, R. B., Koul, R. 2004. A computer-based approach for translating text into concept map-like representations. In Proceedings of the first international conference on concept mapping (pp. 14-17).
- [12] Chen, N. S., Wei, C. W., Chen, H. J. 2008. Mining e-Learning domain concept map from academic articles. Computers & Education, 50(3), 1009-1021.
- [13] Cañas, A. J., K. M. Ford, J. Coffey, T. Reichherzer, R. Carff, D. Shamma, M. Breedy, 2000. Herramientas para Construir y Compartir Modelos de Conocimiento basados en Mapas Conceptuales, Revista de Informática Educativa, Colombia, 13(2), pp. 145-158.
- [14] Protege 2000. The Protege Project. <http://protege.stanford.edu>.
- [15] Kowata, J. H.; Cury, D.; Boeres, M. C. S. 2009. Caracterização das Abordagens para Construção (Semi) Automática de Mapas Conceituais. In: XX Simpósio Brasileiro de Informática na Educação, 2009, Florianópolis, SC. Anais do XX Simpósio Brasileiro de Informática na Educação (SBIE).
- [16] Kowata, J. H.; Cury, D.; Boeres, M. C. S. 2010. Uma abordagem computacional para construção de mapas conceituais a partir de textos em língua portuguesa. In: XXI Simpósio Brasileiro de Informática na Educação, 2010, João Pessoa, PB. Anais do XXI Simpósio Brasileiro de Informática na Educação (SBIE).
- [17] Kowata, J. H. ; Cury, D. ; Boeres, M. C. S. 2011. Construindo de Mapas Conceituais a partir de Textos uma abordagem computacional aplicada à Língua Portuguesa do Brasil. In: XXII Simpósio Brasileiro de Informática na Educação, 2011, Aracaju, Sergipe. Anais do XXII Simpósio Brasileiro de Informática na Educação.
- [18] Starr, R. R. (2009). Uso de mapas conceituais como suporte à aquisição autônoma de conhecimento. Master Dissertation, Instituto Tecnológico da Aeronáutica (ITA), Sao Paulo, Brazil. Tavares, Romero. 2007. "Construindo mapas conceituais." Ciências & Cognição 12: 72-85.
- [19] Amoretti, M, M. Picone, S. Busanelli, M. Amoretti, F. Zanichelli, and G. Ferrari (2015.) Advanced Technologies for Intelligent Transportation Systems, Springer, Germany.
- [20] Novak, J. D.; Alberto J. Cañas. "The theory underlying concept maps and how to construct and use them." Florida Institute for Human and Machine Cognition Pensacola FL, [www.ihmc.us.\[http://cmap.ihmc.us/Publications/ResearchPapers/TheoryCmaps/TheoryUnderlyingConceptMaps.Htm\]](http://www.ihmc.us/Publications/ResearchPapers/TheoryCmaps/TheoryUnderlyingConceptMaps.Htm) 284 (2008).
- [21] Lamas, F. S. L., Boeres, M.C., Cury, D. Um ambiente para comparação de Mapas Conceituais utilizando correspondência de grafos. VIII Ciclo de Palestras Sobre Novas Tecnologias na Educação: Saber criar, saber usar. RS - Brasil: Anais do Evento. 2006. p. 1-10.
- [22] Ismaeel, Alaa A. K. Dynamic Hierarchical Graph Drawing. Dissertation, Karlsruhe Institut für Technologie (KIT), 2012.
- [23] Sugiyama, K.; Tagawa, S.; Toda, M. (1981). "Methods for visual understanding of hierarchical system structures", IEEE Transactions on Systems, Man, and Cybernetics, SMC-11 (2): 109–125.
- [24] Novak, J. D. (1977) A Theory of Education. Ithaca, N.Y.: Cornell University Press.
- [25] Moreira, M. A. "O mapa conceitual como instrumento de avaliação da aprendizagem." Educação e Seleção 10 (2013): 17-34.