Um Time Assíncrono para Minimizar o Número de Cruzamentos de Arestas em Desenho Linear de Grafos

Letícia R. Bueno[1], **C. F. Xavier de Mendonça Neto**[2],
Ademir A. Constantino[3], **Marco A. L. Barbosa**[3]
[1]COPPE Sistemas e Computação, Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

[2]Escola de Artes, Ciências e Humanidades (EACH) – Universidade de São Paulo (USP)
São Paulo – SP – Brasil

[3]Departamento de Informática (DIN) – Universidade Estadual de Maringá (UEM)
Maringá – PR – Brasil

letbueno@cos.ufrj.br,cfxavier@usp.br,{ademir,malbarbo}@din.uem.br

Resumo. *Um Time Assíncrono é apresentado para minimizar o Número de Cruzamentos em Desenho Linear (LCNP - "Linear Crossing Number Problem"). O LCNP consiste no problema de encontrar um desenho de um grafo com o menor número de cruzamentos de arestas onde os vértices de um grafo são posicionados em uma linha reta horizontal no plano e cada aresta é desenhada como um semicírculo em uma das duas páginas. O problema de decisão associado ao LCNP é NP-Completo.*
Palavras-chave: número de cruzamentos, times assíncronos, meta-heurística.
Área de Classificação: otimização combinatória.

Abstract. *An Asynchronous Team is presented to minimize the LCNP - Linear Crossing Number Problem. The LCNP consists in the problem of finding the drawing of a graph with minimum number of edge crossings where the vertices are placed on a straight horizontal line in the plane and each edge is drawn as an circular arc in one of the two pages. The LCNP related decision problem is known to be NP-Complete.*
Keywords: crossing number, asynchronous teams, meta-heuristic.

1. Introdução

A planarização de grafos tem importantes aplicações em várias áreas do conhecimento: desenho de grafos, sistemas de visualização gráfica [di Batista et al. 1994], projeto de placas de circuitos impressos e projeto de circuitos VLSI [Chung et al. 1987]. Além disto, há na literatura uma vasta discussão sobre a minimização do número de cruzamentos de arestas por meio de uma representação em livros que, segundo [Shahrokhi et al. 1997], resultam em métodos efetivos e computacionalmente simples, fornecendo desenhos de grafos próximos do ótimo.

Numa representação em livros (ou desenho linear), os vértices de um grafo são dispostos em uma linha reta horizontal no plano, chamada *espiral*. Podemos dizer que a espiral divide o plano em duas metades chamadas *páginas*. Estas páginas também podem ser vistas como duas páginas de um livro aberto unidas pela espiral. As arestas do grafo são desenhadas como semicírculos em uma das páginas de tal modo que se duas arestas em uma mesma página se cruzam, se cruzam uma única vez e em seu interior. Assim, duas arestas em uma mesma página se encontram em um de seus extremos somente quando são incidentes em um mesmo vértice. Um problema conhecido por *problema de minimização do número de cruzamentos em desenho linear* (LCNP - *"linear crossing number problem"*, em inglês) consiste em minimizar o número de cruzamentos de arestas reposicionando

os vértices na espiral ou mudando arestas de página. O problema de decisão associado ao LCNP é NP-Completo [Chung et al. 1987].

Um dos primeiros a abordar o problema do LCNP foi Nicholson [Nicholson 1968]. O algoritmo apresentado neste trabalho é bastante primitivo com resultados limitados, pois consiste de um algoritmo guloso de tempo $O(n^3)$. Entretanto, o autor apresenta uma demonstração de que qualquer grafo pode ser representado em desenho linear com número mínimo de cruzamentos.

Uma versão restrita do LCNP, a qual chamaremos de *LCNP-fixo*, consiste no LCNP com a posição fixa dos vértices na espiral. Um dos primeiros a estudar esta versão restrita do LCNP foi Cimikowski [Cimikowski e Shope 1996]. Entretanto, apesar da restrição, o problema de decisão associado também é NP-Difícil [Masuda et al. 1990].

Cimikowski em [Cimikowski 2002] mostra que entre diversas heurísticas para o LCNP-fixo, a rede neural proposta em [Cimikowski e Shope 1996] é, sem dúvida, a melhor heurística, alcançando quase sempre a solução ótima. Em ambos os trabalhos [Cimikowski e Shope 1996] e [Cimikowski 2002], a ordem dos vértices na espiral é pré-determinada pela ordem dos vértices em um ciclo hamiltoniano; quando o grafo não possui um ciclo hamiltoniano, esta ordem é aleatória.

Neste trabalho, propomos um Time Assíncrono para o LCNP, buscando a melhor ordem dos vértices na espiral e o melhor posicionamento das arestas nas duas páginas.

O restante deste trabalho está organizado como segue. Na seção 2 apresentamos algumas definições preliminares e a meta-heurística Times Assíncronos. Na seção 3 damos a descrição de nosso algoritmo. Na seção 4 apresentamos alguns resultados experimentais. Concluímos nosso trabalho na seção 5 mostrando algumas vantagens de nossa abordagem e indicando possíveis trabalhos futuros.

2. Preliminares

2.1. Terminologia

Neste trabalho utilizamos a definição de *grafo* conforme o texto-padrão [Bondy e Murty 1976] como sendo grafo finito, simples e não-orientado. Utilizamos também o mesmo texto para as definições de vizinhança e adjacência de um vértice, caminho, trilha, percurso, subgrafo e conexidade.

Seja $G = (V, E)$ um grafo com $n = |V|$ e $m = |E|$. Um *desenho simples* $D(G)$ é um desenho de G no plano tal que nenhuma aresta cruza a si mesma, arestas incidentes em um mesmo vértice não se cruzam, caso duas arestas se cruzem, cruzam-se uma única vez, somente duas arestas se cruzam em um mesmo ponto e nenhuma aresta passa por um vértice se não for incidente no mesmo.

Em um *desenho linear* de G, os vértices de G são dispostos em uma espiral (linha reta), suas arestas são desenhadas como semicírculos em uma das duas páginas, onde toda aresta está inteiramente contida em uma das duas páginas. Ao longo deste trabalho, quando nos referirmos a desenho linear, subentendemos desenho linear simples.

Um ciclo C de G é *ciclo hamiltoniano* se C é um subgrafo conexo de G que contém todos os vértices de G, onde todos os vértices de C tem precisamente 2 vizinhos em C. Chamamos de *ordem de visitação* a ordem de visitação dos vértices em C no percurso entre dois vértices adjacentes u e v, sem passar pela aresta $\{u, v\}$. Chamaremos a ordem dos vértices no caminho de u para v de *ordem hamiltoniana*. A Figura 1 mostra um desenho linear do grafo $K_{3,3}$ em que a ordem dos vértices na espiral é determinada pela ordem hamiltoniana.

Um desenho D de um grafo G é *ótimo* se não existe um desenho de G com menos cruzamentos que D. O número de cruzamentos de um desenho ótimo é chamado na literatura de *número de cruzamentos* de G (“*crossing number*”, em inglês) e é denotado por $cr(G)$. Segue do resultado

(a) Grafo $K_{3,3}$ (b) Desenho linear do $K_{3,3}$

Figura 1: Desenho linear de um grafo

de [Nicholson 1968] que o número de cruzamentos do desenho linear ótimo é igual ao número de cruzamentos do grafo. Um grafo G é *planar* se apresenta $cr(G) = 0$. O problema de decisão associado a determinar o número de cruzamentos de um grafo é NP-Completo [Garey e Johnson 1983].

Neste trabalho, representamos um desenho linear por uma matriz binária A de dimensão $n \times n$ onde, para cada aresta $e = \{i, j\}$ e $i < j$, e está na página superior se $A[i, j] = 1$ e $A[j, i] = 0$ e está na página inferior se $A[i, j] = 0$ e $A[j, i] = 1$.

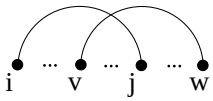


Figura 2: Condição de cruzamentos de arestas

Assim, um cruzamento entre qualquer par de arestas $\{i, j\}$ e $\{v, w\}$ ocorre quando $1 \leq i < v < j < w \leq n$ e ambas estão na mesma página (veja Figura 2). A equação 1 determina o número de cruzamentos de arestas de um desenho linear D.

$$\sum_{i=1}^{n-3} \sum_{j=i+2}^{n-1} \left\{ A[i,j] \sum_{k=i+1}^{j-1} \sum_{l=j+1}^{n} A[k,l] + A[j,i] \sum_{k=i+1}^{j-1} \sum_{l=j+1}^{n} A[l,k] \right\} \quad (1)$$

2.2. A Meta-Heurística

A meta-heurística denominada *Times Assíncronos* (*"Asynchronous Teams"*, em inglês) consiste de uma meta-heurística desenvolvida por [de Souza e Talukdar 1993], sendo aplicada à diversos problemas de otimização combinatória de diferentes tipos de restrições e objetivos. Para uma maior compreensão de times assíncronos indicamos o texto de [Talukdar 1998].

A estrutura de um time assíncrono é formada por uma coleção de agentes autônomos, conectados através de memórias compartilhadas a fim de formar um fluxo de dados cíclico e possibilitar comunicação assíncrona entre os diversos "agentes" do time. A Figura 3 apresenta uma representação do time assíncrono proposto para o LCNP. Intuitivamente, o time funciona como segue: um agente "iniciador" do time (ST+SM e ST+NN) preenche a memória com soluções iniciais. Então, os agentes "construtores" (UP, DP, NN e VE) produzem novas soluções fazendo modificações em soluções escolhidas ao acaso na memória. A escolha de qual agente construtor deve trabalhar é feita de maneira aleatória. Quando uma nova solução é criada, um agente "destruidor" ($COST$ ou $TOUR$) insere a nova solução na memória e, em seguida, decide qual solução deve ser descartada da memória, de acordo com uma "política de destruição".

O maior atrativo dos times assíncronos é a forma com que lidam com funções multi-objetivo e, por esta razão, fizemos a escolha desta meta-heurística para o LCNP. Em problemas multi-objetivo, outras heurísticas otimizam uma função simples, expressa como uma soma ponderada de todos os objetivos e/ou restrições do problema. Assim, é preciso decidir a importância relativa dos vários objetivos. Em um time assíncrono, os objetivos e restrições individuais de um

problema são capturados nos agentes que interagem entre si de maneira autônoma e assíncrona através de memórias compartilhadas contendo soluções candidatas. O problema é particionado por restrições e objetivos e assim, os agentes são desenvolvidos para melhorar soluções com respeito a uma restrição/objetivo específico.

A combinação destes agentes, especialistas em suas próprias restrições e objetivos, produz uma "sinergia" demonstrada em resultados melhores que os produzidos pelos agentes isoladamente. Além disso, a escalabilidade dos times assíncronos permite uma fácil inserção e remoção de agentes, proporcionando maior flexibilidade ao método, de forma que novas alterações podem ser facilmente realizadas de acordo com características ou necessidades particulares.

2.3. Motivação e Objetivos

Dizemos que uma ordem dos vértices de um grafo é *ótima* se existe um desenho linear ótimo para o grafo onde os vértices aparecem nesta ordem na espiral.

É importante observar que a ordem dos vértices na espiral é fundamental para a obtenção de um desenho linear ótimo de um grafo [Shahrokhi et al. 1995]. Assim, se um algoritmo para o LCNP-fixo recebe uma ordem pré-determinada de vértices na espiral que não é ótima, os esforços para posicionar as arestas nas páginas resultarão, no melhor caso, em um desenho próximo do ótimo.

Na literatura, poucos trabalhos dedicam-se ao problema em que a ordem de vértices não é fixa. A razão é que o número de desenhos lineares possíveis aumenta quando permitimos diferentes ordens de vértices, ou seja, dado um grafo $G = (V, E)$ com n vértices e m arestas, o número de desenhos lineares possíveis de G em LCNP-fixo é 2^m. Se a ordem dos vértices não é fixa, existem $(n-1)!/2$ ordens diferentes de vértices e 2^m configurações possíveis das arestas, resultando em $2^{m-1}(n-1)!$ desenhos lineares possíveis.

Além disso, um desenho linear ótimo é obtido pela combinação de uma ordem ótima dos vértices na espiral e de um posicionamento ótimo das arestas nas páginas. Isto é, para resolver o LCNP precisamos "mover" os vértices na espiral, em busca da ordem ótima dos vértices, e "mover" as arestas de uma página para a outra, em busca do posicionamento ótimo das arestas. Por esta razão, a escolha de times assíncronos.

3. O Algoritmo

Para determinar ordens dos vértices na espiral desenvolvemos um agente baseado na técnica conhecida por st-numeração. Dado um grafo biconexo não-orientado $G = (V, E)$ com n vértices, m arestas e uma aresta $\{s, t\}$ qualquer, uma st-numeração de G é uma numeração na qual os vértices de G são numerados de 1 a n, tal que o vértice s recebe o número 1, o vértice t recebe o número n e qualquer vértice v com numeração i (exceto s e t) é adjacente a, pelo menos, um vértice u com numeração menor que i e a, pelo menos, um vértice w com numeração maior que i. A Figura 1 (b) mostra um desenho linear com a ordem dos vértices determinada por uma st-numeração do $K_{3,3}$. Note que, neste caso, esta ordem também é uma ordem hamiltoniana.

Um grafo G é *biconexo* se o grafo $G - v$ é conexo qualquer que seja o vértice v de G.

A st-numeração foi introduzida por [Lempel et al. 1967] para grafos biconexos a uma complexidade de $O(nm)$. Posteriormente, surgiram algoritmos com complexidade $O(n + m)$ [Even e Tarjan 1976, Ebert 1983, Tarjan 1986].

Uma estratégia aconselhável para estabelecer a ordem dos vértices de um grafo G na espiral é segundo uma ordem hamiltoniana. O sucesso da abordagem está em diminuir as possibilidades de cruzamentos de arestas em G, pois a aresta $\{1, n\}$ e as arestas entre vértices consecutivos na espiral não podem se envolver em cruzamentos, de acordo com as restrições do problema. No entanto, determinar se um dado grafo possui um ciclo hamiltoniano é NP-Completo [Garey e Johnson 1979].

Além disso, é possível demonstrar que nem toda ordem hamiltoniana é uma ordem ótima dos vértices na espiral [Chung et al. 1987].

De maneira empírica optamos pela ordem dos vértices determinada pela st-numeração. Dentre as vantagens dessa abordagem estão sua baixa complexidade e a diversidade de soluções, pois existem muitas st-numerações possíveis para cada grafo biconexo e se um grafo admite um circuito hamiltoniano, a ordem hamiltoniana é uma st-numeração válida.

É importante observar que os grafos utilizados em nossos testes são todos biconexos.

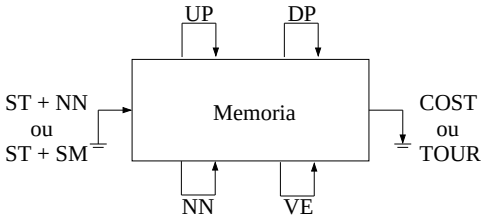


Figura 3: Time Assíncrono para o LCNP

Apresentamos, a seguir, uma descrição dos agentes do time assíncrono para o LCNP (chamaremos por simplicidade de time).

- st-***Numbering*** **(ST)**: Posiciona os vértices na espiral utilizando uma st-numeração do grafo de entrada. Como o algoritmo de st-numeração escolhe os vértices aleatoriamente, diferentes ordens de vértices são disponibilizadas na memória;
- ***Start Memory*** **(SM)**: Posiciona as arestas aleatoriamente nas páginas;
- ***Neural Network*** **(NN)**: Posiciona as arestas nas páginas através da rede neural de [Cimikowski e Shope 1996];
- ***Vertex Exchange*** **(VE)**: Inverte a posição de dois vértices na espiral, modificando apropriadamente suas arestas nas páginas;
- ***Upper Page*** **(UP)**: Método guloso que, para cada aresta na página superior, tenta posicioná-la na página inferior na tentativa de diminuir o número de cruzamentos associado à aresta [Shahrokhi et al. 1996, Cimikowski 2002];
- ***Down Page*** **(DP)**: Método guloso que, para cada aresta na página inferior, tenta posicioná-la na página superior na tentativa de diminuir o número de cruzamentos associado à aresta;
- ***COST***: Agente destruidor que remove soluções da memória de acordo com uma distribuição linear de probabilidades, onde soluções com maior custo tem maior probabilidade de serem eliminadas;
- ***Tournament*** **(TOUR)**: Seleciona duas soluções S_1 e S_2 na memória utilizando a seleção do agente $COST$ (probabilidade proporcional ao custo). Então, compara o custo das soluções S_1 e S_2 e remove a solução de maior custo.

4. Resultados Experimentais

A implementação do time foi feita em Borland^{TM} Delphi^{TM} 7.0. Os testes foram realizados em um PC AMD Athlon XP 1.6 GHz com 256 Mb de RAM.

Representamos o número de soluções na memória por M. O critério de parada do time é o número máximo de iterações ($MaxIter$) e o número de máximo de iterações sem modificação

(Nit) nas soluções da memória. Cada iteração é representada pela execução de um agente construtor. Dessa forma, o algoritmo termina depois de Nit iterações sem nenhuma modificação na memória ou quando $MaxIter$ é alcançado.

Nos experimentos, descartamos a utilização da rede neural (agente NN) quer como agente iniciador quer como agente construtor devido ao alto tempo de execução do mesmo. Além disso, os experimentos mostraram que a ausência do agente NN no time não prejudica a busca pelos melhores resultados.

Dessa forma, quando nos referirmos a resultados da rede neural, nos referimos à rede neural de [Cimikowski e Shope 1996] executada sobre M soluções da memória, onde a ordem dos vértices é determinada por uma st-numeração (agente ST).

Tabela 1: Resultados para grafos aleatórios

Grafo	$\|V\|$	$\|E\|$	M	Time	Neural
g_1	10	22	20	3	3
g_2	45	85	20	6	7
g_3	10	24	20	0	0
g_4	10	25	20	1	1
g_5	10	26	20	3	3
g_6	10	27	20	3	3
g_7	10	34	20	16	17
g_8	25	69	20	0	1
g_9	25	70	20	1	2
g_{10}	25	71	20	9	9
g_{11}	25	72	20	12	14
g_{12}	25	90	20	82	87
g_{13}	50	367	200	5062	5124
$g_1.cimi$	10	21	20	2	2
$g_2.cimi$	60	166	20	20	50
$g_3.cimi$	28	75	20	2	2
$g_4.cimi$	10	22	20	3	3
$g_5.cimi$	45	85	20	5	6
$g_6.cimi$	43	63	20	9	11

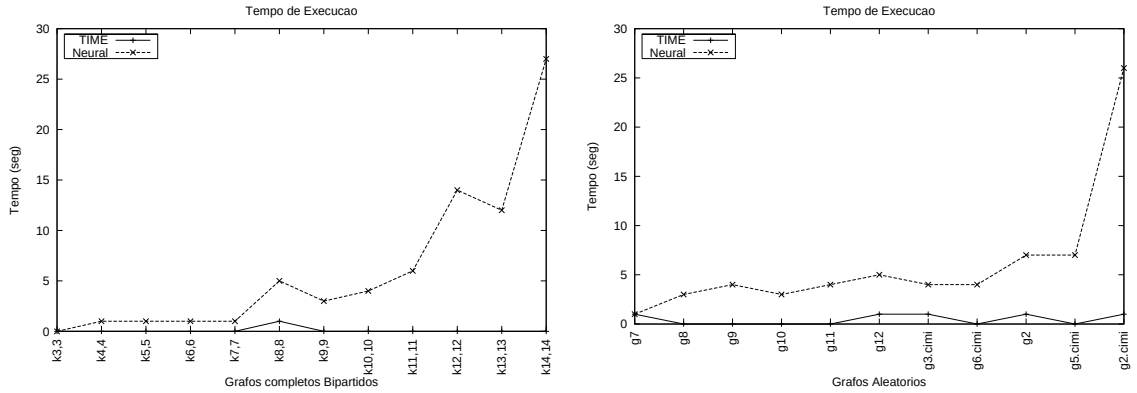
Nestes primeiros testes utilizamos um conjunto de grafos aleatórios descritos na literatura [Goldschmidt e Takvorian 1994, Cimikowski 1995]. Os resultados são apresentados na Tabela 1 com destaque para os valores alcançados pelo time em, aproximadamente, 261 iterações para cada grafo.

O algoritmo também foi aplicado a grafos completos (K_5, K_6, ..., K_{14}) e grafos completos bipartidos ($K_{n,n}$, onde $n = 3, 4, \ldots, 14$). Para cada um destes grafos, tanto o time como a rede neural alcançaram resultados ótimos. Entretanto, ressaltamos que a rede neural exigiu maior tempo de execução, conforme mostra a Figura 4.

Chamaremos de *ordem de adjacências* a ordem em que os vértices são visitados durante o algoritmo de busca em profundidade, utilizado na construção de uma st-numeração. Agora, para verificar a influência da st-numeração na determinação de desenhos lineares ótimos, o time foi executado com três formas de posicionamento dos vértices na espiral: ordem determinada pela st-numeração, ordem de adjacências e ordem aleatória. Os resultados obtidos em grafos aleatórios podem ser vistos nas Figuras 5, 6 e 7.

Observamos que nossos experimentos com grafos completos e grafos completos bipartidos não apresentam nenhum ganho (ou perda) na qualidade da solução final conforme modificamos a forma de posicionamento dos vértices na espiral. Este resultado era esperado para grafos completos, pois neste caso qualquer ordem de vértices é um ciclo hamiltoniano. Também para grafos completos bipartidos, encontrar um ciclo hamiltoniano é trivial.

A vantagem do uso da ordenação determinada pela st-numeração ficou evidente quando



(a) (b)

Figura 4: Tempo de execução do time e da rede neural

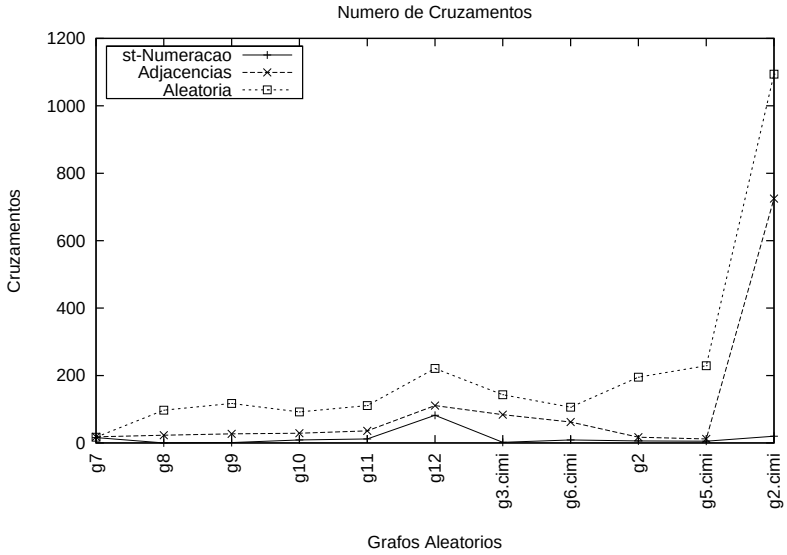


Figura 5: Comparação entre estratégias de posicionamento de vértices

aplicada a grafos gerais, pois neste caso esta ordem (determinada pela st-numeração) apresentou resultados superiores à ordem de adjacências e à ordem aleatória.

Os grafos aleatórios da Figura 5 e 6 são grafos que admitem a existência de um ciclo hamiltoniano. Os grafos da Figura 7 são grafos aleatórios não-planares em que a existência de ciclos hamiltonianos é desconhecida.

Observe que a ordem de adjacências e a ordem aleatória têm resultados mais aproximados para a classe de grafos da Figura 7, quando a existência de ciclos hamiltonianos é desconhecida.

5. Conclusões

Apresentamos um time assíncrono para minimização do número de cruzamentos de arestas em desenho linear de grafos em que a ordem dos vértices na espiral não é fixa. Para encontrar soluções ótimas ou próximas do ótimo, o problema exige a combinação de uma ordem ótima dos vértices na espiral e o posicionamento ótimo das arestas nas páginas. Nosso algoritmo supera a dificuldade encontrada na literatura onde a maioria dos algoritmos apresentados trabalha apenas com o problema onde a ordem dos vértices é fixa.

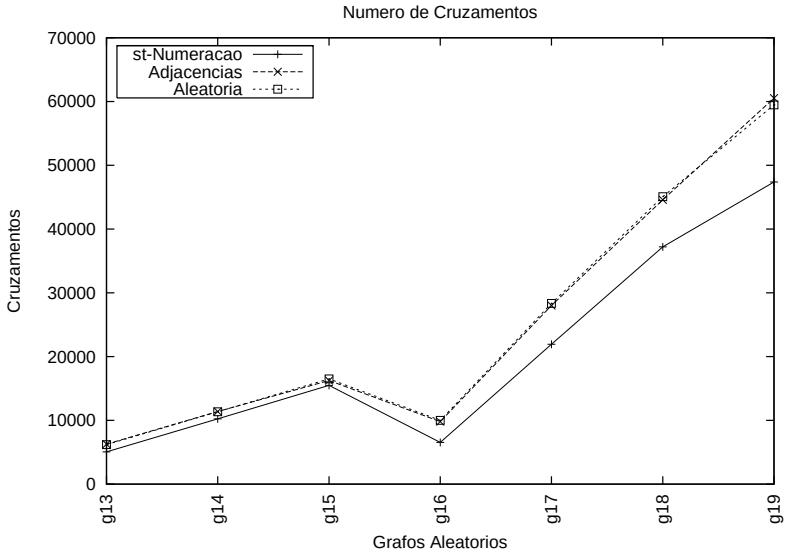


Figura 6: Comparação entre estratégias de posicionamento de vértices

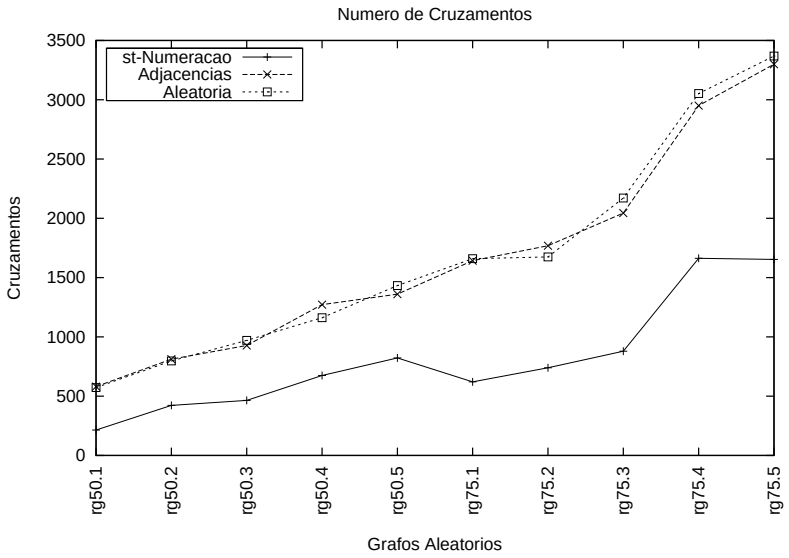


Figura 7: Comparação entre estratégias de posicionamento de vértices

Devido às características dos times assíncronos e sua forma de lidar com funções multi-objetivos obtivemos uma abordagem prática e flexível, onde podemos afirmar, existe uma certa sinergia entre os diferentes agentes trabalhando aspectos particulares do problema. Nos experimentos em classes de grafos com valor ótimo conhecido, nosso time alcançou o número mínimo de cruzamentos em tempo inferior ao obtido em [Cimikowski 2002]. A facilidade de inserção e remoção de novos agentes permite ainda que alterações sejam realizadas de acordo com as características particulares de cada aplicação. Finalmente, a facilidade de implementação de times assíncronos em arquitetura paralela [de Souza e Talukdar 1993] provavelmente resultará em melhores soluções dentro de um tempo mais aceitável.

A st-numeração utilizada de maneira empírica para determinar a ordem dos vértices na espiral, apresentou melhores resultados que as outras ordenações experimentadas. Como a

determinação da ordem dos vértices através de ciclos hamiltonianos é difícil devido à sua complexidade computacional, a st-numeração demonstrou ser uma estratégia superior.

6. Agradecimentos

Os autores agradecem o suporte parcial do CNPq (processo 305534/2004-1).

Referências

Bondy, J. A. e Murty, U. S. R. (1976). *Graph Theory with Applications*. MacMillan, London.

Chung, F. R. K., Leighton, F. T., e Rosenberg, A. L. (1987). Embedding graphs in books: a layout problem with applications to VLSI design. *SIAM Journal of Algebra Discrete Methods*, 8:33–58.

Cimikowski, R. (1995). An analysis of heuristics for the maximum planar subgraph problem. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 322–331.

Cimikowski, R. (2002). Algorithms for the fixed linear crossing number problem. *Discrete Applied Mathematics*, 122:93–115.

Cimikowski, R. e Shope, P. (1996). A neural network algorithm for a graph layout problem. *IEEE Transactions on Neural Networks*, 7(2):341–349.

de Souza, P. S. e Talukdar, S. N. (1993). Asynchronous organizations for multi-algorithm problems. *ACM Symposium on Applied Computing*.

di Batista, G., Eades, P., Tamassia, T., e Tollis, I. G. (1994). Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4:235–282.

Ebert, J. (1983). st-ordering the vertices of biconnected graphs. *Computing*, 30(1):19–33.

Even, S. e Tarjan, R. E. (1976). Computing an *st*-numbering. *Theoretical Computer Science*, 2:339–344.

Garey, M. R. e Johnson, D. S. (1979). *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA.

Garey, M. R. e Johnson, D. S. (1983). Crossing number is NP-Complete. *SIAM Journal on Algebraic and Discrete Methods*, 4:312–316.

Goldschmidt, O. e Takvorian, A. (1994). An efficient graph planarization two-phase heuristic. *Networks*, 24:69–73.

Lempel, A., Even, S., e Cederbaum, I. (1967). An algorithm for planarity testing of graphs. In *Theory of Graphs: International Symposium (Rome 1966)*, pages 215–232, New York. Gordon and Breach.

Masuda, S., Nakajima, K., Kashiwabara, T., e Fujisawa, T. (1990). Crossing minimization in linear embeddings of graphs. *IEEE Transactions on Computers*, 39(1):124–127.

Nicholson, T. A. J. (1968). Permutation procedure for minimising the number of crossings in a network. *Proc. Inst. Elec. Eng.*, 115(1):21–26.

Shahrokhi, F., Sýkora, O., Székely, L., e Vrťo, I. (1997). Crossing numbers: Bounds and Applications. In I. Barany, K. B., editor, *Intuitive Geometry*, volume 6, pages 179–206. Bolyai Soc. Math. Studies, Akademia Kiado, Budapest.

Shahrokhi, F., Sýkora, O., Székely, L. A., e Vrťo, I. (1995). Book embeddings and crossing Numbers. In *WG'94: Proceedings of the 20th International Workshop on Graph-Theoretic*

Concepts in Computer Science, volume 903 of *Lecture Notes in Computer Science*, pages 256–268, Berlin. Springer-Verlag.

Shahrokhi, F., Székely, L. A., Sýkora, O., e Vrťo, I. (1996). The book crossing number of a graph. *Journal of Graph Theory*, 21(4):413–424.

Talukdar, S. N. (1998). Autonomous cyber agents: rules for collaboration. *Proceedings of the thirty-first Hawai International Conference on System Sciences (HICSS-31)*.

Tarjan, R. E. (1986). Two streamlined depth-first search algorithms. *Fundamenta Informaticae*, 9:85–94.

Times AssÃncronos; uma arquitetura de Time AssÃncrono para o. treinamento de Redes Neurais Artificiais; comparaÃ§Ã£o entre os resultados obtidos com o Time.Â princÃpio da Regra Delta, ou seja, tem o objetivo de. minimizar o erro quadrÃ¡tico mÃ©dio da camada de saÃda. da Rede Neural Artificial e utiliza informaÃ§Ãµes do. gradiente da funÃ§Ã£o erro quadrÃ¡tico mÃ©dio.Â alternativas para a estrutura de Time AssÃncrono. Uma. alternativa seria um time formado por vÃ¡rias cÃ³pias do.