Geração Quase-Instantânea de Orientações Acíclicas em Sistemas Distribuídos Anônimos

Gladstone Moises Arantes Junior e Felipe Maia Galvão França
Engenharia de Sistemas e Computação-COPPE, Universidade Federal do Rio de Janeiro
Caixa Postal 68511, 21945-970, Rio de Janeiro, Brasil
{glads,felipe}@cos.ufrj.br

Resumo—
Este artigo discute um conjunto de algoritmos distribuídos não-determinísticos para a geração de orientações acíclicas em um sistema distribuído anônimo de topologia arbitrária. Embora possam ser concebidas outras aplicações práticas para essa forma de quebra de simetria, estaremos focando o problema da inicialização de um sistema para a operação do *Escalonamento por Reversão de Arestas (ERA)*, um simples e poderoso algoritmo de escalonamento distribuído. O ERA requer uma orientação acíclica inicial no grafo que representa o compartilhamento de recursos do sistema alvo para executar corretamente e tal condição inicial determina a "quantidade de concorrência" associada à dinâmica do escalonamento. Os algoritmos são analisados tanto em termos de tempo de convergência quanto em termos de quantidade de concorrência produzida. Em particular, é proposto um novo algoritmo chamado *Alg-Arestas* que, em certas condições, é capaz de produzir orientações acíclicas quase instantaneamente, isto é, em menos de dois (2) passos.

Palavras-chave— **sistemas anônimos, escalonamento distribuído, algoritmos distribuídos randômicos, quebra de simetria.**

Abstract—
This paper discusses a set of non-deterministic distributed algorithms for the generation of acyclic orientations upon anonymous distributed systems of arbitrary topology. Although other practical applications of this form of symmetry breaking may be devised, we'll be focusing on the problem of priming a target system for *Scheduling by Edge Reversal (SER)*, a simple and powerful distributed scheduling algorithm. SER requires an initial acyclic orientation on the graph representing the target resource sharing system in order to work correctly and such initial orientation determines an amount of "concurrency" associated to the scheduling dynamics. The set of algorithms is analysed both in terms of convergency time and in terms of the amount of *concurrency* produced. In particular, a new algorithm called *Alg-Arestas* is proposed which, under appropriate conditions, is able to produce acyclic orientations quasi instantaneously, i.e., in less than two (2) steps.

Keywords— **anonymous system, distributed scheduling, randomized distributed algorithms, symmetry breaking.**

I. Introdução

Neste trabalho apresentaremos três algoritmos probabilísticos que geram orientações acíclicas em sistemas distribuídos anônimos. Sendo um dos possíveis métodos de quebra de simetria (criação de conjuntos independentes maximais e coloração são outras formas, todas equivalentes entre si em algum nível), esta é uma aplicação muito útil, com muitos trabalhos publicados, incluindo algoritmos paralelos, distribuídos e seqüenciais, além de determinísticos e probabilísticos ([GOL 87], [GOL 88], [ITA 90], [LUB 86], [PAN 92], [SZE 93]).

Porém, nesse artigo, estaremos analisando uma aplicação em especial: como passo inicial necessário para o funcionamento correto de um algoritmo bem conhecido para escalonamento distribuído chamado Escalonamento por Reversão de Arestas (*Scheduling by Edges Reversal*) ou ERA Este algoritmo define uma regra muito simples para a operação de cada nó em um sistema distribuído de tal forma que vizinhos nunca operam ao mesmo tempo. Este algoritmo foi desenvolvido tendo em vista o problema de escalonamento de processos em sistemas com alta carga, nos quais cada nó precisa de um subconjunto de todos os recursos disponíveis para poder operar.

O ERA pressupõe uma orientação acíclica em um grafo que modela os nós e o compartilhamento de recursos do sistema (definido na próxima seção). Cada possível orientação tem uma "qualidade" associada que é a medida da eficiência ou da concorrência do escalonamento do sistema. Os algoritmos que geram as orientações são analisados, portanto, não só em termos da velocidade da sua convergência como também da quantidade de concorrência das orientações que geram.

Na seção II, descrevemos e analisamos as principais propriedades do ERA e a representação do sistema distribuído em termos de grafos. Na seção III, introduzimos dois dos algoritmos de geração de orientações acíclicas (Alg-Viz e Alg-Arestas), discutindo as suas velocidades de convergência. As qualidades das orientações são analisadas na seção IV, onde também é introduzido um algoritmo otimizado para geração de boas orientações: Alg-Cor.

II. O sistema distribuído e o ERA.

A. Sistema Distribuído Anônimo.

Um sistema distribuído pode ser simplificadamente descrito como um conjunto de nós de processamento

(podendo ser muito simples ou muito complexos) organizados de tal forma que cada nó está conectado a um subconjunto dos outros por um canal de comunicação qualquer. A comunicação entre cada par de nós se dá através de envio de mensagens. Comumente, assume-se que as conexões são bidirecionais, o que significa que um nó conectado a outro tanto pode enviar mensagens a ele quanto receber. Esta é uma definição bastante genérica, podendo ser aplicada para uma enorme gama de sistemas.

Um sistema distribuído é considerado anônimo quando não existe um identificador que diferencie um nó de outro. No nosso caso, estaremos assumindo ainda que não exista nenhuma informação global acerca do sistema, ou seja, não há disponível, *a priori*, informações como o número total de nós, o número de canais ou ainda qual a topologia do sistema. Esse tipo de sistema apresenta grandes dificuldades para ser manipulado [BAR 96].

B. Objetivos do ERA e o grafo de dependências.

O *Escalonamento por Reversão de Arestas* [BAR 89] visa resolver o problema de controlar o acesso a recursos compartilhados entre processadores em um ambiente distribuído. Assim, dado um conjunto N de nós de processamento, um conjunto R de recursos e um conjunto $D \subseteq (P \times R)$ dos pares $(n \in N, r \in R)$ em que o nó n utiliza o recurso r, determinar a ordem de execução (ou operação) dos nós de tal forma que dois nós que utilizam o mesmo recurso não executem ao mesmo tempo e de tal forma que não ocorra *deadlock* nem *starvation*. O ERA foi criado para condições de alta carga no sistema, ou seja, cada nó precisa de todos os recursos para que possa operar.

Todo sistema distribuído pode ser modelado como um grafo $G(N, E)$. Nesse grafo, N é o conjunto dos nós n do sistema e E é o conjunto de arestas $m=(n_i,n_j)$, onde n_i e n_j são nós de N. Se $m=(n_i,n_j) \in E$, então existe um canal de comunicação entre n_i e n_j.

A partir dessa definição podemos construir um *Grafo de Dependências* $G^D(N,E^D)$, $E^D \subseteq E$ que, sendo um subgrafo de G, irá representar os padrões de compartilhamento do sistema. O conjunto E^D é composto das arestas que ligam dois nós que compartilham pelo menos um recurso, ou seja, $m=(n_i,n_j) \in E^D$ se, e somente se, $(n_i,n_j) \in E$ e $\exists r \in R$, tal que, $(n_i,r) \in D$ e $(n_j,r) \in D$. Observe que, nessa definição, assumimos que um recurso só poderá ser compartilhado entre nós que tenham um canal de comunicação entre si, o que é razoável. Também vale observar que, nessa notação, cada recurso compartilhado corresponde a um subgrafo completamente conectado, já que todos os nós que o compartilham terão arestas entre eles.

C. O funcionamento do ERA.

O algoritmo ERA admite a existência inicial de uma orientação acíclica no grafo G^D. Basicamente uma aresta (n_i,n_j) orientada no sentido $n_i \rightarrow n_j$ indica que o nó nj tem precedência em relação ao nó n_i na próxima operação. A necessidade de a orientação ser acíclica é facilmente percebida, já que, de acordo com a definição acima, um ciclo implicaria uma cadeia de precedências entre nós organizada de forma a gerar um bloqueio perpétuo ou *deadlock*.

Assim, um nó para operar precisa ser um sumidouro, ou seja, deverá ter todas as arestas orientadas em sua direção, já que isso equivale a ter precedência sobre todos os vizinhos (como já foi dito, vizinhos em G^D não podem operar ao mesmo tempo). Uma orientação acíclica em G^D implica a existência de pelo menos um sumidouro em N, assim é garantido que o algoritmo pode sempre iniciar com os sumidouros operando. Após sua operação, cada sumidouro reverte as suas arestas na direção oposta. A nova orientação gerada é também acíclica, pois todos os sumidouros foram transformados em fontes e esse tipo de operação claramente não cria ciclos. Por isso, um novo conjunto de sumidouros será obtido. O processo, então se repete, com os sumidouros operando e revertendo suas arestas, definindo assim o escalonamento pretendido.

D. Propriedades do ERA.

Esse simples passo de escalonamento apresenta propriedades muito interessantes que serão introduzidas aqui. Para uma descrição completa destas e de outras propriedades, incluindo provas, ver [BAR 89].

Chamamos de ω_i uma orientação acíclica qualquer. Uma operação ou escalonamento é uma seqüência $\sigma(\omega_1)=\omega_1,\omega_2,\ldots$ de orientações acíclicas, cada uma gerada pela reversão das arestas da orientação anterior nos moldes do ERA. Pode-se perceber que o escalonamento realizado de forma síncrona (como estaremos considerando aqui, com todos os sumidouros operando e revertendo as arestas sincronamente) é totalmente definido pela orientação inicial do grafo de dependências, já que cada passo é bem definido e determinístico. Assim, podemos encarar ω_1 como a própria operação.

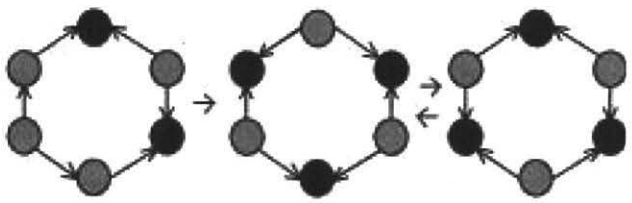

Fig.1: Exemplo de operação com um ciclo de apenas 2 orientações acíclicas.

Também é fácil verificar que, dado que existe um número finito de orientações possíveis, é garantido que o algoritmo entra em um ciclo de orientações, com seqüências de orientações se repetindo a cada ciclo (fig. 1). Assim, podemos escrever que $\exists i,k\ 1 \leq i < k$ para os quais $\omega_i=\omega_k$. Observe que poderá existir um conjunto de

orientações iniciais que não serão parte do ciclo que será formado, mas pelo qual a operação irá passar antes de entrar na primeira orientação pertencente ao ciclo.

Denotamos $m_i(\omega_k,q)$ o número de vezes que n_i operou nos q passos do algoritmo após ω_k. Uma importante propriedade do ERA é que, em um ciclo, o número de vezes que cada nó do sistema opera é igual, independentemente de serem vizinhos ou não em G^D.

Assim, como o ciclo se repete indefinidamente, temos que $\forall n_i,n_j \in N\ lim_{q\to\infty} m_i(\omega_l,q) = lim_{q\to\infty} m_j(\omega_l,q)$, o que garante não um escalonamento *starvation-free*, como ainda oferece uma forma de justiça na operação, pois, no limite, nenhum nó opera mais do que outro. Na verdade, a diferença só poderá ocorrer na seqüência de orientações anteriores ao ciclo.

Podemos, portanto, chamar apenas $m(\omega_l)$ o número de vezes que um nó opera dentro do ciclo (como é igual para todos os nós, não é necessário o subscrito). Também podemos chamar $p(\omega_l)$ o tamanho do ciclo alcançado a partir de $\omega 1$. Este é, na verdade, o número de diferentes orientações acíclicas existentes no ciclo.

Uma boa medida da eficiência de uma operação é a razão $m(\omega_l)/p(\omega_l)$. Esta é uma matematização da "quantidade de concorrência" gerada por uma operação e é representada por $\gamma(\omega_l)= m(\omega_l)/p(\omega_l)$.

É fácil notar a relação $1/2 \leq \gamma(\omega_l) \leq 1/n$ (onde n é o número de nós de N), já que, na pior das hipóteses, cada nó do sistema opera uma vez sozinho e, na melhor, cada par de nós vizinhos se reveza na operação sem intervalos em que nenhum dos dois opera.

Existe uma expressão que representa o valor de $\gamma(\omega_l)$. Para mostrá-la, devemos introduzir algumas notações. Se não for uma árvore, então, existe pelo menos um ciclo simples κ em G^D (ciclos simples são aqueles que passam, no máximo, uma vez por cada nó). $|\kappa|$ é o número de nós de κ e K é o conjunto de todos os ciclos simples de G^D.

Depois de aplicada a orientação ω_l no grafo, denotamos por $n^+(\kappa,\omega_l)$ e $n^-(\kappa,\omega_l)$ o número de arestas, em κ, orientadas na direção horária e anti-horária respectivamente.

Definimos a expressão abaixo:

$$\rho(\kappa,\omega_1) = \min\left\{\frac{n^+(\kappa,\omega_1)}{|\kappa|}, \frac{n^-(\kappa,\omega_1)}{|\kappa|}\right\}$$

Teorema 1. Dado um grafo G^D orientado inicialmente por ω_l, a concorrência do escalonamento gerado pela operação do ERA é dada pela expressão

$\gamma(\omega_l)=min_{\kappa\in K}\rho(\kappa,\omega)$.

Prova. Em [BAR 89]. □

III. Os algoritmos.

A. Calabrese/França

A.1 Apresentação do Algoritmo.

Os algoritmos que aqui descrevemos foram desenvolvidos a partir das idéias básicas contidas em [CAL 94] e [CAL 97]. Este utiliza um gerador de números aleatórios simples que pode gerar 0 ou 1 randomicamente. Por isso, dizemos tratar-se de uma moeda.

O algoritmo executa sincronamente. Cada nó é dito probabilístico se ainda está participando do algoritmo (nesse caso, ele ainda tem arestas incidentes não orientadas e continua tomando parte nos sorteios) ou determinístico, no caso de não participar mais por já ter tido todas as arestas incidentes orientadas. Em cada passo do algoritmo todos os nós probabilísticos lançam uma moeda, obtendo 0 ou 1. Um nó que obtiver 1 e cujos vizinhos probabilísticos restantes tiverem obtido 0 irá orientar todas as suas arestas ainda não orientadas na sua direção. O algoritmo executa até que todas as arestas sejam orientadas.

Tal algoritmo é bastante ineficiente se a probabilidade de obter-se 0 ou 1 no sorteio for 1/2. Por isso, comumente, a moeda é "viciada" ou polarizada, da seguinte forma: $P\{moeda_i=1\} = 1/(viz(n_i)+1)$ e $P\{moeda_i=0\} = 1 - P\{moeda_i=1\}$, onde $viz(n_i)$ representa o número de vizinhos probabilísticos do nó n_i. Desta forma, intuitivamente, em uma dada vizinhança, apenas um dos nós tende a obter 1 no sorteio.

A.2 Tempo de convergência.

Tempo de convergência é a medida do número de passos que o algoritmo precisar dar para que este convirja (termine), ou seja, até que todos os nós se tornem determinísticos. Em [CAL 97] existem duas análises de convergência do algoritmo para sistemas com características específicas. A primeira mostra que a convergência para grafos completos (ou seja, totalmente conectados) é $O(|N|)$, ou simplesmente, $O(n)$. Intuitivamente, temos que, em cada passo do algoritmo, para grafos completos, $P\{moeda_i=1\}=1/n_{prob}$, ($n_{prob}$ é o número de nós probabilísticos no passo). A prova demonstra que a probabilidade de algum nó ganhar no sorteio (tirar 1 e todos os outros tirarem 0) é $\geq 1/e$ e, portanto, haverá um ganhador, na média, em menos de e passos. Portanto, para todos os nós tornarem-se determinísticos deverá ocorrer um número de passos menor que $n.e$, portanto $O(n)$.

A outra análise é sobre grafos onde o número de vizinhos de cada nó de um grafo menor que uma constante k independente de n. Essa grandeza é denominada grau máximo do grafo ou Δ. Nesses grafos, que são um caso de grafos esparsos, o algoritmo converge em $O(log\ n)$.

B. Alg-Viz.

Esse algoritmo é uma extensão muito simples do anterior e propõe que o sorteio seja realizado com números em uma faixa de valores inteiros. Isso equivale à utilização de um dado de f faces em vez de uma moeda. Assim, poderemos obter como resultado do sorteio números inteiros de *0* a *f-1*.

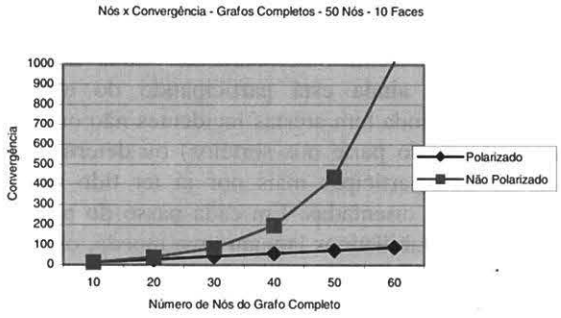

Fig. 2: Alg-Viz. Dados polarizados e não polarizados.

Assim, Alg-Viz pode ser descrito da mesma forma que o algoritmo Calabrese/França, sendo necessário chamar a atenção apenas para o fato de que, agora, um nó ganha no sorteio quando tira o maior valor entre os seus vizinhos.

Da mesma forma que no algoritmo Calabrese/França, a utilização de um gerador de números aleatórios não polarizados torna o algoritmo muito ineficiente, a não ser que existisse alguma relação entre f e alguma característica global do grafo, o que não é razoável, já que trabalhamos com a hipótese de não existência de informações globais do sistema. A figura 2 dá uma idéia, através de resultados de simulações, da melhora ocorrida com a utilização de polarizações. As simulações apresentadas nesse trabalho são versões seqüenciais dos algoritmos escritas em linguagem C. Cada ponto do gráfico é obtido pela simulação com centenas de grafos conexos gerados aleatoriamente.

O método de polarização utilizado é diferente para cada nó do sistema e é mostrado abaixo:

$$f \le viz(n_i)+1:$$

$$P\{dado_i = f-1\} = \frac{1}{viz(n_i)+1}$$

$$P\{dado_i = \alpha\} = \frac{1}{viz(n_i)+1}\left(1 - \sum_{k=\alpha+1}^{f-1} P\{dado_i = k\}\right) 0 \le \alpha \le f-2$$

$$f > viz(n_i)+1:$$

$$P\{dado_i = \alpha\} = \frac{1}{f}, \forall \alpha \text{ tal que } 0 \le \alpha \le f-1$$

Desta forma, mantém-se uma polarização similar àquela utilizada em Calabrese/França para os casos em que $f \le viz(n_i)+1$ e, para os casos em que $f > viz(n_i)+1$, aproveita-se o fato de haver mais faces do que vizinhos para particionar melhor a probabilidade de cada face. Esta polarização foi escolhida por ter se mostrado a mais eficiente na comparação com outras alternativas.

B.1 Tempo de convergência.

Obviamente, as análises de convergência realizadas para o algoritmo Calabrese/França são igualmente válidas para Alg-Viz, já que, Alg-Viz apenas faz um particionamento na maior na distribuição de probabilidades de Calabrese/França. Também não parece que a utilização de dados em vez de moedas modifique a complexidade da convergência a não ser que haja uma relação entre f e uma característica global do sistema (o que já foi dito não ser uma opção para o nosso caso). A grande vantagem de *Alg-Viz* em relação a Calabrese/França é diminuir (e, no caso, de dados com muitas faces, anular) o efeito dos empates entre vizinhos na convergência do algoritmo, permitindo uma análise mais fiel do tempo de convergência do algoritmo.

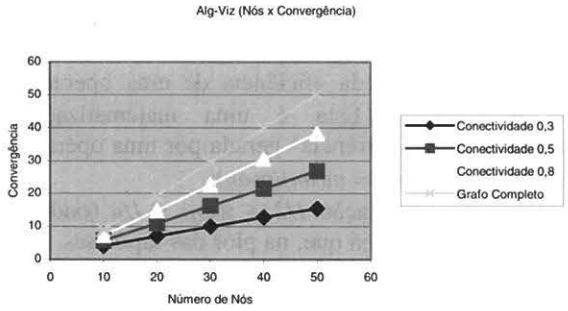

Fig. 3: Alg-Viz - com q constante, conv é função linear de n.

Antes de prosseguirmos na apresentação das propriedades de convergência de Alg-Viz, precisamos definir exatamente o que consideraremos conectividade de um grafo daqui para frente. Isso se deve ao fato de a conectividade de um grafo, no sentido de "quão conexo" é o grafo, exerce forte influência sobre o tempo de convergência, já que grafos mais conexos tenderão a convergir mais devagar para o fim.

Para nós, a conectividade q de um grafo será a razão entre o número de arestas que o grafo possui e o número de arestas que ele poderia possuir com o mesmo número de nós. Logo

$$q = \frac{2m}{n(n-1)}$$

Pela figura 3, pode-se observar uma característica interessante acerca do tempo de convergência (*conv*) de Alg-Viz. Note que utilizamos daqui para frente números grandes de f de tal forma que este não influencie na verificação da convergência do algoritmo, já que praticamente anula a possibilidade de empates nos sorteios. Pelo gráfico, claramente, temos que, mantendo-se q

constante, *conv* é função linear de n. Além disso, o coeficiente da reta fica muito próximo de q.

C. Alg-Arestas.

Este algoritmo apresenta uma filosofia um pouco diferente dos anteriores (Alg-Viz e Calabrese/França). Em vez tentar gerar sumidouros entre os nós probabilísticos, Alg-Arestas muda o foco de atenção para as arestas.

O algoritmo também é simples e funciona assim: considere um nó n_i e todos os seus vizinhos v_{ij} ligados a n_i pelas arestas m_{ij}. Todos os nós do sistema sorteiam um número com um dado de f faces, obtendo $dado(n_i)$. n_i irá comparar $dado(n_i)$ com $dado(v_{ij})$ para todo v_{ij} tal que m_{ij} ainda não foi orientada. Para cada m_{ij} em que $dado(n_i)>dado(v_{ij})$, orienta m_{ij} na direção de n_i. O processo é repetido até não sobrar nenhuma aresta sem orientação.

Uma prova formal de que Alg-Arestas funciona é dada abaixo:

Teorema 2. Alg-Arestas sempre gera orientações acíclicas em qualquer grafo $G(N,E)$.

Prova. Definimos K como sendo o conjunto de todos os ciclos simples (sem nós repetidos) em $G(N, E)$. A idéia é provar que o algoritmo não gera ciclos para nenhum $\kappa \in K$. κ é composto dos nós $u_0,...,u_{|\kappa|-1}$, considerados em ordem de vizinhança no ciclo, de tal forma que u_i é conectado por uma aresta de κ a $u_{(i+1) \bmod |\kappa|}$, $0 \le i < |\kappa|$. Os números sorteados para os nós u_i são dados por u^s_i. Temos duas possibilidades:

(a) $u^s_0=u^s_1=...=u^s_{|\kappa|-1}$ - Nesse caso, nenhuma aresta do ciclo é orientada e o sorteio é executado novamente;

(b) $\exists a$, $0 \le a \le |\kappa|-1$, tal que $u^s_a < u^s_{(a+1) \bmod |\kappa|}$ - Nesse caso, tomemos uma nova nomeação dos nós como $v_0,v_1,...,v_{|\kappa|-1}$ tal que $v_i=u_{(a+i) \bmod |\kappa|}$. Como $v_0=u_a$ e $v_1=u_{(a+1) \bmod |\kappa|}$, já sabemos que $v^s_0 < v^s_1$, portanto a aresta (v_0,v_1) terá a direção $v_0 \rightarrow v_1$. Desta forma, se $\forall j$, $1 \le j < |\kappa|-2$, tivermos que $v^s_j \le v^s_{j+1}$, então $v^s_{|\kappa|-1} > v^s_0$ e, portanto, a aresta $(v_{|\kappa|-1},v_0)$ será orientada na direção $v_0 \rightarrow v_{|\kappa|-1}$, tornado v_0 um sumidouro nas arestas de κ, impossibilitando a formação de ciclo em κ nos passos seguintes do algoritmo. Por outro lado, se $\exists j$, $1 \le j < |\kappa|-2$ tal que $v^s_j > v^s_{j+1}$, então a aresta (v_j,v_{j+1}) será orientada na direção $v_j \leftarrow v_{j+1}$. Assim, teremos a seguinte situação (o símbolo $\leftrightarrow$ significa orientação indefinida) que define a impossibilidade de formação de um ciclo em κ nos passos seguintes do algoritmo.

$$v_{|\kappa|-1} \leftrightarrow v_0 \rightarrow v_1 \leftrightarrow ... \leftrightarrow v_j \leftarrow v_{j+1} \leftrightarrow ... \leftrightarrow v_{|\kappa|-1}. \ \square$$

C.1 Convergência de Alg-Arestas.

É bastante fácil perceber que Alg-Arestas tem convergência muito mais rápida que Alg-Viz. Vamos apresentar duas medidas analíticas dessa convergência.

A análise intuitiva é a seguinte: em um sorteio, a probabilidade de empate entre dois vizinhos é de $1/f$. Assim, enquanto o número de arestas é grande, esta é a proporção de empates ocorridos a cada passo do algoritmo. Assim, em cada passo, uma proporção de $1/f$ das arestas não orientadas até aquele passo permanecem sem serem orientadas. Portanto, a convergência é dada por $conv(Alg\text{-}Arestas) = \lceil \log_f m \rceil$, onde m é o número de arestas inicialmente. Apesar de intuitivamente razoável e de apresentar resultados próximos da realidade, esta análise não é realmente precisa, pois a proporção $1/f$ só é válida quando o número de arestas sendo consideradas é suficientemente grande.

Abaixo, introduzimos e demonstramos um teorema que é resultado de uma análise mais complexa, porém bastante fiel, do tempo de convergência do algoritmo ($E[x]$ representa a média de x - *expected value*).

Teorema 3. Dado um grafo $G(N,E)$ com m arestas e um dado com m faces, o algoritmo Alg-Arestas converge, na média, de acordo com a seguinte expressão:

$$E[conv] = \sum_{i=1}^{\infty} \left[\left(1 - f^{-i}\right)^m - \left(1 - f^{-i+1}\right)^m \right]$$

Prova. Cada aresta é orientada independentemente das outras. Portanto, podemos definir uma variável aleatória de Bernoulli que modela cada tentativa de orientação de uma aresta, ou seja, a disputa de dados entre os dois nós que são conectados por esta. A probabilidade de fracasso, ou seja, de a aresta não se orientar, é igual à de dar empate na disputa entre dois dados de número de faces iguais a f, ou seja, $1/f$. Assim:

Probabilidade de fracasso: $q=1/f$
Probabilidade de sucesso: $p=1-q=1-1/f$

Desta forma, o número de sorteios necessários até que uma dada aresta consiga se orientar pode ser modelado por uma variável aleatória geométrica X com p e q dados acima. Assim, a probabilidade de uma aresta ter que dar i passos até se orientar é dada por:

$$P(X = i) = pq^{i-1} = \left(1 - \frac{1}{f}\right)\left(\frac{1}{f}\right)^{i-1} = \frac{f-1}{f^i} \quad (1)$$

As tentativas de orientação das diversas arestas são independentes. Tratam-se, portanto, de m variáveis iguais e independentes: $X_o,X_1,...X_{m-1}$ (podendo ser representadas individualmente apenas por X). Para uma dada execução do algoritmo, a convergência é dada por $Max\{X_o,X_1,...X_{m-1}\}$, já que é o número de tentativas da última aresta a ser orientada que nos importa. Assim, a convergência média do algoritmo é dada por:

$$E[conv] = \sum_{i=1}^{\infty} i.P[Max\{X_0,...,X_{m-1}\} = i] \quad (2)$$

Porém, temos que:

$$P[\mathrm{Max}\{X_0,\ldots,X_{m-1}\}=i]=$$
$$=\sum_{j=1}^{m}\left[\binom{m}{j}.P(X=i)^j.P(X<i)^{m-j}\right] \quad (3)$$

A equação (3) é um binômio de Newton sem a primeira parcela (j=*0*). Portanto, podemos reescrevê-la:

$$[P(X=i)+P(X<i)]^m - P(X<i)^m \quad (4)$$

Sabemos também que:

$$P(X<i)=1-P(X\geq i)=1-\sum_{j=i}^{\infty}P(X=j)=\frac{f^i-f}{f^i} \quad (5)$$

Substituindo (5) e (1) em (4) - que é igual a (3) -, podemos calcular (2), obtendo a expressão desejada. □

A tabela abaixo confronta, para alguns valores de *m* e *f*, os resultados obtidos por simulação, pela expressão aproximada ($\lceil \log_f m \rceil$) e pelo somatório. Também foi incluída a expressão $\log_f m+1$ que, para alguns valores, apresenta uma melhor aproximação do que $\lceil \log_f m \rceil$, apesar de não ter uma relação clara com a média dos valores.

TABELA I

COMPARAÇÃO ENTRE AS CONVERGÊNCIAS OBTIDAS NAS SIMULAÇÕES E OS VALORES PREVISTOS NAS ANÁLISES

Arestas	Faces	Simulação	Σ	$\lceil \log_f m \rceil$	$\log_f m+1$
100	10	2.70	2.74	2	3
200	110	1.82	1.86	2	2.13
300	210	1.79	1.77	2	2.07
400	310	1.72	1.73	2	2.04
500	410	1.70	1.71	2	2.03

O tempo de convergência de Alg-Arestas é bastante pequeno quando comparado com Alg-Viz. Na verdade, na situação em que $f>>m$, podemos considerar que o algoritmo converge em apenas 1 passo, o que é um resultado muito bom, já que podemos sem muito custo utilizar valores bastante grandes de *f*. Na figura 4 podemos observar o tempo de convergência do algoritmo para diferentes valores de *f* e *m*. Note que os valores são realmente pequenos, mesmo para um número reduzido de faces do dado.

IV. CONCORRÊNCIA.

Nesta seção pretendemos mostrar a relação existente entre as concorrências geradas e os algoritmos propostos. Nas figuras 5 e 6, são mostrados gráficos para as concorrências geradas por Alg-Viz e Alg-Arestas, respectivamente. Os valores são confrontados com a conectividade dos grafos. Em cada gráfico, há curvas para grafos com diferentes números de nós.

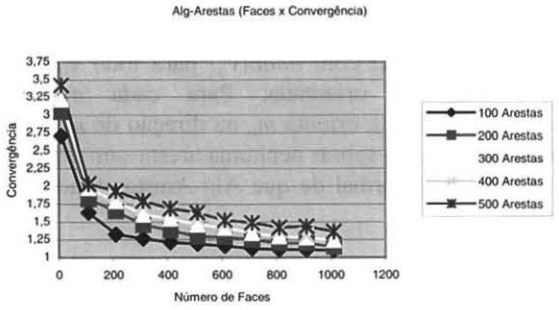

Fig. 4: Tempo de convergência de Alg-Arestas.

As concorrências geradas por Alg-Viz são normalmente superiores àquelas geradas por Alg-Arestas (apesar de a escala e o valor fixo para o caso de árvores prejudicarem um pouco essa percepção).

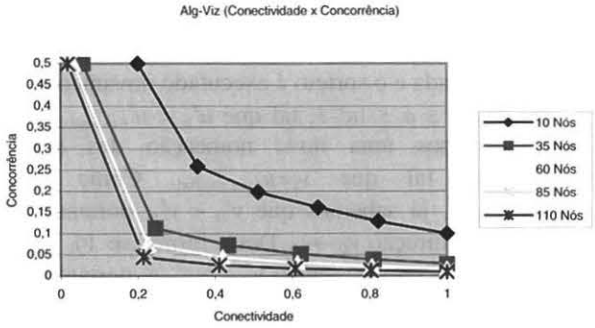

Fig. 5: q x γ para Alg-Viz.

Considerando apenas a expressão da concorrência apresentada na seção II não é possível ter uma idéia de quais características uma orientação precisa ter para resultar em uma boa concorrência. Assim, é igualmente difícil avaliar porque um algoritmo pode gerar orientações melhores do que outros.

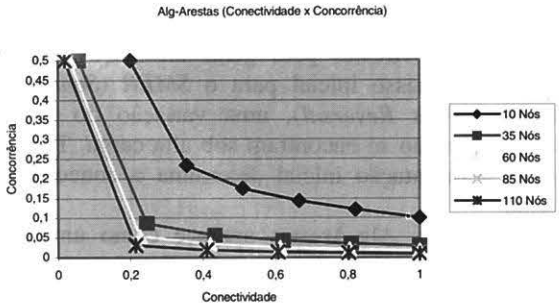

Fig. 6: q x γ para Alg-Arestas.

Uma suposição intuitiva era a de que a geração de um número maior de sumidouros inicialmente deveria corresponder a orientações com potencialmente maior concorrência. Realmente, se considerarmos o tipo de orientação gerada por Alg-Viz e Alg-Arestas, não é difícil prever que o primeiro tenderá a produzir um número maior de sumidouros (até porque é baseado em criação de sumidouros) o que casa com a constatação de que Alg-Viz realmente produz melhores concorrências que Alg-Arestas.

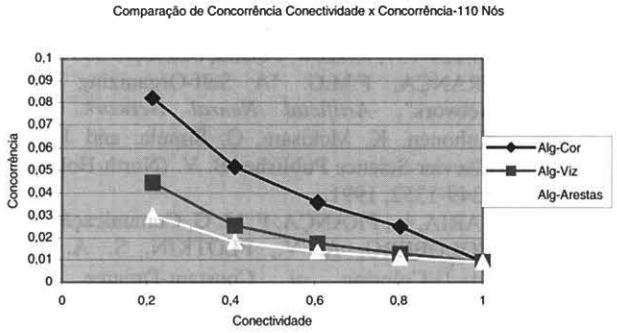

Fig. 7: Comparando γ entre os algoritmos.

A. Alg-Cor.

Desta forma, introduzimos um outro algoritmo, chamado de Alg-Cor, que é uma extensão imediata de Alg-Viz e que tem por objetivos gerar concorrências melhores que os anteriores, privilegiando a criação de sumidouros.

A idéia é a seguinte: considere um caminho simples (sem repetição de nós) no grafo formado pelos nós $n_0,n_1,...,n_k$ em ordem de vizinhança. Imagine que Alg-Viz oriente estes nós exatamente na ordem apresentada. O único sumidouro possível entre estes nós é n_0. Alg-Cor tende a aumentar o número de sumidouros não orientando as arestas do nó tão logo ele ganhe o sorteio, mas atrasando a orientação das arestas para um momento posterior.

Funciona assim: o nó, quando ganha o sorteio, escolhe uma cor (número inteiro positivo qualquer). Esta cor não deve ser igual à de nenhum dos vizinhos já coloridos e deve ser a menor possível. Assim, por exemplo, se 2 vizinhos de um nó já estão coloridos com 0 e 2, o nó escolhe 1. Se já estão coloridos com 2 e 3, o nó escolhe 0. Cada nó que for colorido orienta as arestas incidentes a vizinhos já coloridos na direção da maior cor.

Na seqüência discutida acima, os nós pares não estariam impedidos de se tornarem sumidouros, pois escolheriam a sempre o 0 como cor, já que os ímpares escolheriam 1. É claro que estamos desconsiderando os outros vizinhos do nó fora da seqüência, o que não invalida o argumento de que um maior número de sumidouros seria gerado.

Como pode ser observado na figura 7, as concorrências geradas por Alg-Cor superam em grande margem as geradas por Alg-Viz e Alg-Arestas.

B. Concorrência e o maior caminho orientado.

A figura 8 mostra um gráfico de dispersão com o resultado de 100 execuções de cada algoritmo, comparando o número de sumidouros gerados em uma orientação e a concorrência da mesma. Apesar do sucesso no desenvolvimento de Alg-Cor, esta figura mostra que não é exatamente o número de sumidouros que influi tão fortemente na concorrência gerada. Tal conclusão vem do fato de, entre as orientações geradas por um mesmo algoritmo, essa relação não se concretizar.

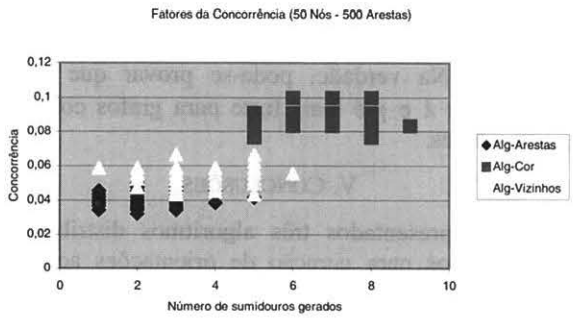

Fig. 8: O número de sumidouros não determina a qualidade da concorrência.

Nesse ponto, introduzimos o conceito de decomposição por sumidouros ou *sink decomposition*. Esta é uma partição de N em conjuntos disjuntos $S_0,S_1,...,S_{\lambda-1}$, de tal forma que $n \in S_k$ se, e somente se, o maior caminho direcionado de n até um sumidouro tem k arestas. Estes conjuntos podem ser construídos assim: em S_0 insira todos os sumidouros do grafo, retirando-os deste. Insira todos os sumidouros do grafo induzido restante em S_1 e assim sucessivamente.

No ERA, a operação se dá primeiramente pelos nós de S_0. Após operarem, estes nós revertem as suas arestas, indo fazer parte de outros conjuntos. Os nós de S_1 passam a ser

do conjunto S_0 e isso vai indefinidamente. Note que λ é o número de arestas do maior caminho orientado do grafo. Note também que λ nunca aumenta com a operação do ERA, pois ao sair de S_0 nunca irá para um suposto S_λ, já que todos os nós do antigo $S_{\lambda-1}$ fazem parte agora de $S_{\lambda-2}$.

Uma relação interessante, que surge do entendimento dessa dinâmica, é $\gamma \geq 1/\lambda$ [FAR 99], já que, na pior das hipóteses, λ não diminui e cada nó opera uma vez e vai para compor $S_{\lambda-1}$. Portando, λ define indiretamente um limite inferior para a concorrência gerada pela orientação. Logo, é um bom candidato a ser examinado como fator relevante para a formação da concorrência.

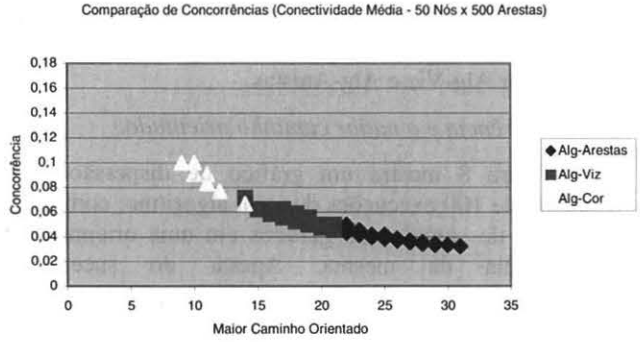

Fig. 9: λ x q - grafo com conectividade média.

De fato, pela figura 9, vemos a importância de λ na composição de γ. Esta relação não é determinística, pois é possível obter duas orientações ω e ω' em que $\lambda(\omega)>\lambda(\omega')$ e $\gamma(\omega)>\gamma(\omega')$. Na verdade, pode-se provar que a relação inversa entre λ e γ é mais forte para grafos com maiores conectividades.

V. Conclusões.

Foram apresentados três algoritmos distribuídos não determinísticos para geração de orientações acíclicas em sistemas distribuídos anônimos: Alg-Viz (uma extensão direta do algoritmo Calabrese/França); Alg-Cor (semelhante a Alg-Viz e que gera melhores concorrências); e Alg-Arestas (que, em certas circunstâncias, gera orientações acíclicas em menos de 2 passos).

Os tempos de convergência dos algoritmos foram analisados tanto através de simulações quanto de modelos matemáticos. Particularmente, para Alg-Arestas, além das simulações, foram mostrados dois modelos matemáticos descrevendo a convergência do sistema.

Os três algoritmos foram testados como passo inicial para o algoritmo de escalonamento distribuído conhecido como ERA, particularmente no que tange à "quantidade de concorrência" gerada.

A. Trabalhos futuros.

Um possível trabalho futuro é a utilização de uma versão de Alg-Arestas para gerar orientações em multi-grafos como passo inicial para o *SMER* (*Scheduling by Multiple Edges Reversal*), uma variação do ERA para sistemas que não se encontram sob alta carga. Também no SMER, a orientação inicial determina a concorrência do sistema.

Além disso, Alg-Arestas aparece como um poderoso algoritmo de quebra de simetria. Estuda-se também a utilização desse algoritmo em outras aplicações que possam se valer de uma orientação acíclica para sua resolução.

Referências

[BAR 89] BARBOSA, V. C. "Concurrency in Heavily Loaded Neighborhood-Constrained Systems", *ACM Transactions on Programming Languages and Systems*, Vol. 11, n. 4, pp. 562-584, Outubro 1989.

[BAR 96] BARVOSA, V. C. *An Introduction to Distributed Algorithms,* MIT Press 1996.

[CAL 94] CALABRESE, A., FRANÇA, F.M.G. "Randomized Distributed Primer for the Updating Control of Anonymous ANNs", *Proceedings of ICANNR94*, Sorrento, Italy, 1994.

[CAL 97] CALABRESE, A. "Distributed Acyclic Orientation of Asynchronous Networks", *11th International Symposium of Fundamentals of Computation Theory*, pp.129-137, Kraków, Poland, Setembro 1997.

[FRA 91] FRANÇA, F.M.G. "A Self-Organizing Updating Network", *Artificial Neural Network,* eds. T. Kohonen, K, Mckisara, O. Simula, and J. Kangas, Elsevier Science Publisher B. V. (North-Holland), pp. 1349-1352, 1991.

[FAR 99] FARIA, L., FRANÇA, F. M. G. Comunicação Pessoal

[GOL 87] GOLDBERG, A. V., PLOTKIN, S. A. "Parallel $(\Delta+1)$-Coloring of Constant-Dregree Graphs", *Information Processing Letters,* n. 25, pp. 241-245, 1987.

[GOL 88] GOLDBERG, A. V., PLOTKIN, S. A. "Parallel Symmetry-Breaking in Sparse Graphs", *SIAM Journal of Discrete Mathematics,* Vol. 1, n .4, pp. 434-445, Novembro 1988.

[ITA 90] ITAI, A., RODEH, M. "Symmetry-Breaking in Distributed Networks", *Information and Control,* n. 88, pp. 60-87, 1990.

[LUB 86] LUBY, M., "A Simple Parallel Algorithm for the Maximal Independent Set Problem", *SIAM Journal of Computing,* Vol. 15, n. 4, pp. 1036-1053, Novembro 1986.

[PAN 92] PANCONESI, A., SRINIVASAN, A., "Improved Distributed Algorithms for Coloring and Network Decomposition Problems", *24th ACM Symposium on Theory of Computation,* pp. 581-591, Victoria B. C., Canada, Maio 1992.

[SZE 93] SZEGEDY, M., VISHWANATHAN, S. "Locality Based Graph Coloring", *25th ACM Symposium on Theory of Computation,* pp. 201-207, California, USA, Maio 1993.