ADAM SALGADO BANZI

DISSEMINAÇÃO BIO-INSPIRADA DE EVENTOS EM REDES DINÂMICAS E DESCENTRALIZADAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientadora:Profa. Dra. Aurora Trinidad Ramirez Pozo
Co-Orientador:Prof. Dr. Elias P. Duarte Jr.

CURITIBA

2011

ADAM SALGADO BANZI

DISSEMINAÇÃO BIO-INSPIRADA DE EVENTOS EM REDES DINÂMICAS E DESCENTRALIZADAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientadora:Profa. Dra. Aurora Trinidad Ramirez Pozo
Co-Orientador:Prof. Dr. Elias P. Duarte Jr.

CURITIBA

2011

SUMÁRIO

RESUMO

O advento de novos paradigmas de comunicação, tais como a computação móvel e o modelo *peer-to-peer*, demandam modelos de rede auto gerenciáveis e descentralizados. O controle e administração dos recursos em modelos descentralizados não é uma tarefa simples, uma vez que essa tarefa deve ser realizada conjuntamente pelos integrantes da rede. Uma maneira promissora de lidar com esse problema e que vem obtendo sucesso é o uso do algoritmo da Otimização por Colônia de Formigas (*Ant Colony Optimization* (ACO)). O presente trabalho apresenta uma estratégia de gerência distribuída baseada em inteligência coletiva, que permite a descoberta e a disseminação de novidades em redes de topologia dinâmica. Uma novidade, também chamada de evento, é definida como a mudança de estado de um nodo ou enlace da rede. Um nodo que detecta um evento em sua vizinhança dispara a disseminação das novidades pela rede. O algoritmo simula o comportamento das colônias de formigas da natureza em busca de alimento. Através de "marcas" deixadas no meio, chamadas feromônios, as formigas guiam-se indiretamente umas as outras em direção ao alimento através das melhores rotas. No algoritmo proposto, as formigas são representadas por agentes móveis que circulam e disseminam as novidades pela rede. O uso do ACO apresenta a vantagem de não necessitar de uma entidade central para controlar a disseminação de informações, característica de interesse no contexto do trabalho. Um estudo empírico foi realizado, comparando a estratégia proposta com a disseminação por inundação (*flooding*) e a disseminação epidêmica (*gossip*). Resultados permitem observar que o algoritmo apresenta uma solução de compromisso entre o tempo necessário para a disseminação e a sobrecarga em termos do número de mensagens utilizadas.

ABSTRACT

Dynamic networks, such as *peer-to-peer* and mobile *ad hoc* networks require decentralized management: it is not feasible to delegate monitoring and control tasks to a single manager. In this type of environment, management tasks have to be collectively executed by all active nodes. This work presents a strategy based on swarm intelligence for spreading events in dynamic and decentralized networks. An event is defined as a state transition of a node or link. Ants, which correspond to mobile agents, spread event information throughout the network. A node that detects an event in its neighborhood starts disseminating the new information. Pheromones are used to both control the ant population and to help define the paths that ants traverse. An empirical study was performed, in which the proposed strategy was compared with two classical dissemination strategies: *flooding* and *gossip* algorithms. Simulations' results show that the proposed strategy presents a good trade-off between the time required to disseminate information and the overhead in terms of the number of messages.

CAPÍTULO 1

INTRODUÇÃO

Redes dinâmicas e descentralizadas apresentam um comportamento particular. Uma rede dinâmica sofre alterações frequentes em sua composição. Assim, são exemplos de acontecimentos comuns em tais redes a criação e remoção de enlaces, bem como o surgimento e saída de nodos da rede. Modelos dinâmicos e descentralizados representam redes *peer-to-peer* não estruturadas e redes sem fio *ad hoc*.

A disseminação de eventos é um dos desafios das redes dinâmicas, tais como redes móveis *ad hoc* e *peer-to-peer*. Em modelos tradicionais de rede, ou seja, aqueles em que há pouca modificação da composição da rede, o controle e administração dos recursos é, comparativamente, simplificado [11]. Por outro lado, nas redes dinâmicas, a própria frequência com que nodos entram e saem é alta, impedindo que as decisões de disseminação sejam tomadas de forma antecipada a partir de um conhecimento prévio da rede [2]. Outra característica relevante é a descentralização, ou seja, a ausência de um ponto único que possa ser responsabilizado pelo controle. Devido ao alto dinamismo, a manutenção do funcionamento da rede deve ser realizada conjuntamente pelos nodos ativos.

Para auxiliar na tarefa de gerenciamento de recursos envolvidos inclusive na disseminação de informações, é importante que os integrantes da rede tornem-se cientes das mudanças ocorridas na topologia de maneira rápida e eficiente. Em modelos centralizados e com pouco dinamismo, pode ser usada uma estratégia determinística para disseminação de eventos. Para as redes dinâmicas, a descoberta e disseminação de novidades aos participantes necessita de estratégias também dinâmicas e descentralizadas, que se adaptam a alterações imprevistas.

Algumas estratégias podem ser utilizadas para tratar deste problema. De um lado, o algoritmo da inundação (*flooding*) [26] oferece disseminação de informações de maneira rápida e confiável, com a desvantagem da grande quantidade de informações redundantes

disseminadas para atingir o objetivo. Como solução alternativa, a disseminação epidêmica (*gossip*) [16] procura comunicar as informações com uma menor quantidade de mensagens, embora não o faça de forma confiável e, dependendo do tipo de topologia da rede, possa não ser capaz de atingir o objetivo de comunicar as informações para todos os nodos. Assim, uma estratégia eficaz deve tanto ser capaz de comunicar as informações para todos os nodos quanto não sobrecarregar a rede com uma quantidade excessiva de mensagens para atingir tal objetivo.

Uma solução promissora para este problema origina-se da observação da natureza. Certos grupos de seres vivos, tais como as colônias de formigas e enxames de abelhas, apresentam a chamada inteligência coletiva [7]. Inteligência coletiva é a propriedade apresentada por sistemas constituídos por agentes dotados de comportamento autônomo, capazes de executar ações simples. Tais agentes, coletivamente, podem apresentar comportamento inteligente [33]. Em outras palavras, as interações entre os diversos elementos é a chave para a realização da tarefa desejada.

A estratégia apresentada neste trabalho é baseada no algoritmo *EvAnt*, que utiliza inteligência coletiva para disseminar eventos em redes dinâmicas e descentralizadas. O algoritmo de inteligência coletiva é requisitado somente em situações nas quais há novidades a serem informadas aos elementos da rede. Estudos empíricos foram realizados a fim de avaliar a eficiência e a eficácia da abordagem. A comparação da estratégia proposta foi realizada com a inundação (*flooding*) e a disseminação epidêmica (*gossip*). A eficiência é avaliada através da velocidade que o algoritmo consegue disseminar novidades e a carga imposta à rede durante tal processo. A eficácia é medida ao avaliar se um determinado método consegue ou não disseminar as novidades para toda a rede.

1.1 Objetivos

Ao longo da dissertação foram explorados temas relacionados ao uso de inteligência coletiva aplicado a redes dinâmicas e descentralizadas. Especificamente, foi discutida a disseminação de informações em tais modelos de rede, uma vez que essa tarefa é a base para diversos tipos de serviços, tais como roteamento, compartilhamento de recursos,

entre outros.

A fim de verificar a eficiência e eficácia da solução proposta, uma comparação foi realizada entre o método apresentado neste trabalho com outros algoritmos de referência em matéria de disseminação de informações.

1.2 Organização do trabalho e contribuições

A lista a seguir apresenta a organização do trabalho bem como um breve resumo da contribuição de cada capítulo.

Capítulo 2: o capítulo 2 apresenta alguns conceitos básicos que são importantes para o entendimento do trabalho. O capítulo inicia com a definição de topologia de uma rede seguida pela descrição de alguns modelos conhecidos de topologia, onde incluem-se aqueles que serão utilizados para as simulações realizadas.

Capítulo 3: no capítulo 3 é descrito o conceito de inteligência coletiva, bem como sua origem e funcionamento. A seguir, é tratado o algoritmo da Otimização por Colônia de Formigas, descrevendo como ele funciona e algumas das variações que possui.

Capítulo 4: o capítulo 4 contém a descrição dos métodos com os quais a estratégia proposta foi comparada. Trabalhos que possuem alguma relação com a solução proposta também aparecem descritos no capítulo. Assim, o conteúdo deste capítulo somado ao que foi mostrado nos dois capítulos anteriores compõem a base teórica necessária para a apresentação da estratégia proposta.

Capítulo 5: o capítulo 5 trata sobre a disseminação de informações baseada em eventos, seguida de uma descrição detalhada do algoritmo *EvAnt*, a base da solução proposta.

Capítulo 6: o capítulo 6 apresenta todas as informações relativas às simulações realizadas, tais como o ambiente utilizado, as métricas avaliadas e o procedimento utilizado para realizar as simulações. Em seguida, são expostos os resultados obtidos entre a comparação do algoritmo *EvAnt* com outros métodos, além da análise dos resultados obtidos.

Capítulo 7: por fim, o capítulo 7 apresenta, de forma sucinta, os pontos principais discutidos ao longo do trabalho. Em seguida, são colocadas as conclusões obtidas, bem como as ideias para pesquisas futuras.

CAPÍTULO 2

CONCEITOS BÁSICOS

Ao longo deste capítulo serão apresentados vários conceitos amplamente utilizados ao longo do trabalho e cujo conhecimento é essencial para o bom entendimento da solução proposta.

2.1 Topologia

Redes de computadores são formadas por dois componentes básicos: o primeiro deles são os *hosts*, ou seja, os computadores. O outro componente são os enlaces (*links*), que conectam os *hosts* um com os outros. À maneira como os *hosts* e enlaces estão dispostos na rede dá-se o nome de *topologia*.

A topologia de uma pequena rede é mostrada na figura 2.1. A topologia dessa rede é composta pelos nodos numerados de 0 a 6 e pelos seguintes enlaces: $(0,1), (0,2), (0,3), (1,3), (1,4), (2,5), (3,6)$ e $(5,6)$.

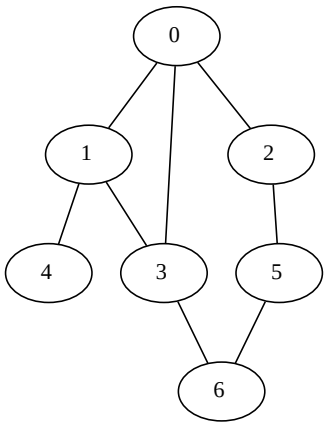


Figura 2.1: Exemplo de uma rede com 7 nodos e 8 enlaces.

Alguns modelos de rede seguem determinadas regras específicas. Em uma rede totalmente conectada (*fully connected*) cada nodo está diretamente conectado com todos os outros. Um exemplo de rede totalmente conectada é mostrado na figura 2.2.

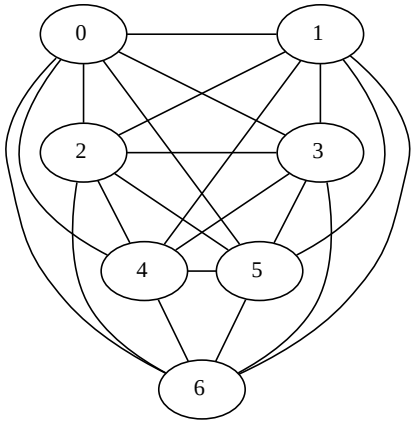


Figura 2.2: Rede totalmente conectada de 7 nodos.

Em uma rede de *topologia aleatória*, os nodos de origem e destino de cada enlace são escolhidos aleatoriamente. Se a mesma semente for utilizada para a escolha de todos os enlaces, haverá uma distribuição equilibrada dos graus que cada nodo terá. Esse modelo de topologia não faz sentido em uma rede real, sendo utilizado somente para fins de pesquisa e simulação.

A topologia *powerlaw* [17] surgiu com o objetivo de ser um modelo que reflete a proporção entre *hosts* e enlaces tal como ela é na Internet. Foi constatado que, na Internet, há um determinado conjunto de *hosts* "populares", ou seja, que possuem muitos enlaces, enquanto o restante dos nodos possui uma quantidade bem menor de conexões.

Um algoritmo utilizado para gerar redes *powerlaw* é formado por dois passos [9]. Primeiramente, um conjunto de m_0 nodos é conectado através de $m_0 - 1$ enlaces. O segundo passo consiste em adicionar novos enlaces ou novos nodos:

- Com uma probabilidade p, são adicionados $m \leq m_0$ novos enlaces. Nodos "populares" possuem maiores chances de estarem conectados por estes novos enlaces.
- Com uma probabilidade $1 - p$, é adicionado um novo nodo à rede. Esse novo nodo possuirá m novos enlaces. Novamente, os nodos com maior número de conexões possuem preferência para serem o destino destes novos enlaces.

Um exemplo de rede *powerlaw* é mostrado na figura 2.3. Os nodos altamente conectados estão numerados de 0 a 4. Um ponto importante a ser notado é que todos os

outros nodos, com exceção do nodo 8, estão ligados a algum membro do grupo de nodos altamente conectados.

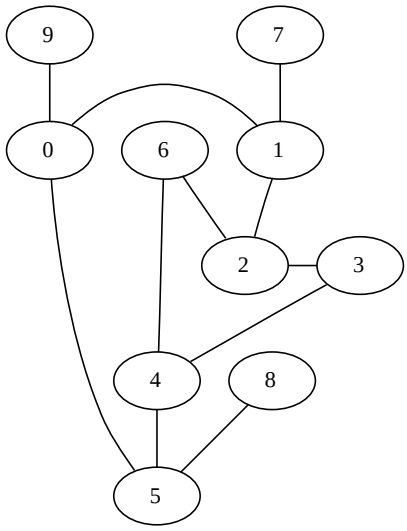


Figura 2.3: Rede *powerlaw* de 10 nodos.

2.2 Redes dinâmicas e descentralizadas

Redes dinâmicas e descentralizadas apresentam um comportamento particular. Modelos desse tipo representam redes *peer-to-peer* não estruturadas e redes sem fio *ad hoc*.

Uma rede dinâmica sofre alterações frequentes em sua topologia. Assim, são eventos comuns em tais redes a criação e remoção de enlaces, bem como o surgimento e saída de nodos da rede.

Descentralização implica em não haver um determinado ponto responsável pelo gerenciamento dos recursos da rede. Redes descentralizadas devem ser "auto gerenciáveis", ou seja, um conjunto de participantes torna-se responsável pela administração dos recursos. Nas redes consideradas neste trabalho o gerenciamento dos recursos é realizado conjuntamente por todos os nodos ativos.

O controle descentralizado deve ser realizado conjuntamente por cada integrante. Uma vez que qualquer nodo da rede pode tornar-se inativo a qualquer momento, é indesejável que os nodos dependam da comunicação com os vizinhos para a tomada de decisões. Assim, para decidir sobre aspectos referentes ao gerenciamento de recursos, cada nodo

toma suas decisões com base apenas em informações locais.

Com a crescente adoção de modelos *peer-to-peer* na Internet, torna-se valioso o conhecimento de estratégias para a disseminação eficaz de informações em tais modelos de rede.

2.3 Agentes móveis

O uso de agentes móveis é uma estratégia que pode ser utilizada para disseminar informações em redes dinâmicas e descentralizadas. Um agente pode ser interpretado como sendo um pacote que carrega as informações a serem comunicadas aos participantes da rede. Periodicamente, tais agentes partem de um determinado nodo em direção a um vizinho, carregando as informações coletadas. Tanto o nodo onde o agente está localizado quanto o próprio agente podem ser responsáveis pela escolha do próximo destino a ser visitado. Essa decisão depende da estratégia utilizada.

Neste trabalho, o termo "mensagem" será utilizado para descrever um agente móvel enviado de um nodo a outro. O conteúdo da mensagem são as informações carregadas pelo agente, ou seja, as informações que o agente coletou pelos nodos os quais já visitou.

CAPÍTULO 3

INTELIGÊNCIA COLETIVA

Na natureza, observa-se que diversas espécies estruturam-se em grupos. Através desse meio de organização, os diversos indivíduos compartilham uma série de vantagens, tais como melhor proteção contra predadores, divisão de tarefas, coleta de alimentos, entre outras [35]. O termo *coletivo* (ou enxame) é usado para referenciar um determinado grupo de indivíduos (também chamados de *agentes*) capazes de interagirem mutuamente. Há vários exemplos reais de agrupamentos existentes na natureza. Um deles é o enxame de abelhas. Outro exemplo é a colônia de formigas. A ideia de agrupamento pode ser estendida para outras situações diferentes de comportamentos observados na natureza. Por exemplo, um engarrafamento pode ser visto como um agrupamento no qual os indivíduos são os carros; uma economia é um agrupamento composto por agentes econômicos no papel de indivíduos [35].

Pelos diversos benefícios que a maneira de mútua colaboração entre diversos agentes de um agrupamento apresenta no mundo real, surgiu a ideia de usar essa forma de organização para a criação de soluções computacionais, inicialmente com o objetivo de coordenar as ações de robôs [35]. O nome inteligência coletiva (*swarm intelligence*) surgiu primeiramente para referir-se a sistemas de robôs formados por agentes de comportamento simples que interagiam segundo as regras de um determinado meio. Segundo [33], a inteligência coletiva é uma propriedade apresentada por sistemas formados por agentes com pouca capacidade individual, porém com comportamento coletivo inteligente.

Assim, na Ciência da Computação, a inteligência coletiva surge como uma boa abordagem para a resolução de problemas de natureza distribuída os quais possam ser modelados através de agentes capazes de interagirem uns com os outros e com o meio onde estão inseridos.

A seguir serão apresentados alguns modelos de algoritmos que fazem uso da inteligência

coletiva. Um deles, a Otimização por Colônia de Formigas (*Ant Colony Optimization*), será descrito com maior detalhamento, por ser o algoritmo usado na estratégia apresentada neste trabalho.

3.1 Otimização por Colônia de Formigas

A Otimização por Colônia de Formigas (*Ant Colony Optimization* (ACO)) é um conjunto de algoritmos que se inspira no comportamento, observado na natureza, das colônias de formigas reais. A versão inicial do ACO foi apresentado em 1992 em [12]. O ACO procura simular em ambiente computacional o comportamento das formigas reais em busca do menor caminho entre o ninho e o local onde se localiza o alimento [31]. Esses caminhos utilizados pelas formigas representam, no algoritmo, uma possível solução para o problema a ser resolvido.

Na natureza, para as formigas comunicarem-se umas com as outras, elas depositam, no ambiente, durante os trajetos de ida e volta entre o ninho e o local da comida, uma substância chamada *feromônio*. A presença de feromônios nos diversos caminhos influencia o comportamento das formigas, uma vez que, ao sentirem a presença dessas substâncias por meio do olfato, as formigas dão preferência por rotas com maior concentração de feromônios [15]. A essa forma de comunicação por trilha de feromônios é dado o nome de *estigmergia*. Esse comportamento de escolha de determinadas rotas por meio das trilhas de feromônios é exemplificado na figura 3.1.

Em 3.1(a), as formigas que saem do ninho ("*Nest*") devem decidir entre pegar um dos dois caminhos até o local da comida ("*Food*"). Inicialmente, as formigas não possuem nenhuma informação adicional para guiá-las pelo melhor caminho. Logo, a escolha é feita de forma aleatória. Assim, é esperado que metade siga por um caminho enquanto a outra parte siga pelo outro, conforme mostrado em 3.1(b). Se considerado que as formigas movem-se com velocidades iguais, aquelas que tomarem o menor caminho chegarão antes ao local desejado, conforme visto em 3.1(c). Logo, a quantidade de feromônios depositada nas rotas mais curtas, representada pelas linhas ao longo do caminho, será maior. Com o tempo, a diferença das quantidades de feromônios entre os caminhos longo e curto

será expressiva o suficiente para influenciar as novas formigas do sistema a escolherem o caminho mais curto, que possui a maior quantidade da substância, conforme mostrado na figura 3.1(d). Com um número cada vez maior de formigas que utilizam o menor caminho, a trilha de feromônios ali presente aumenta ainda mais, em um efeito denominado retorno positivo (*positive feedback*) [22]. Após mais algum tempo, todas as formigas já escolherão a rota mais curta.

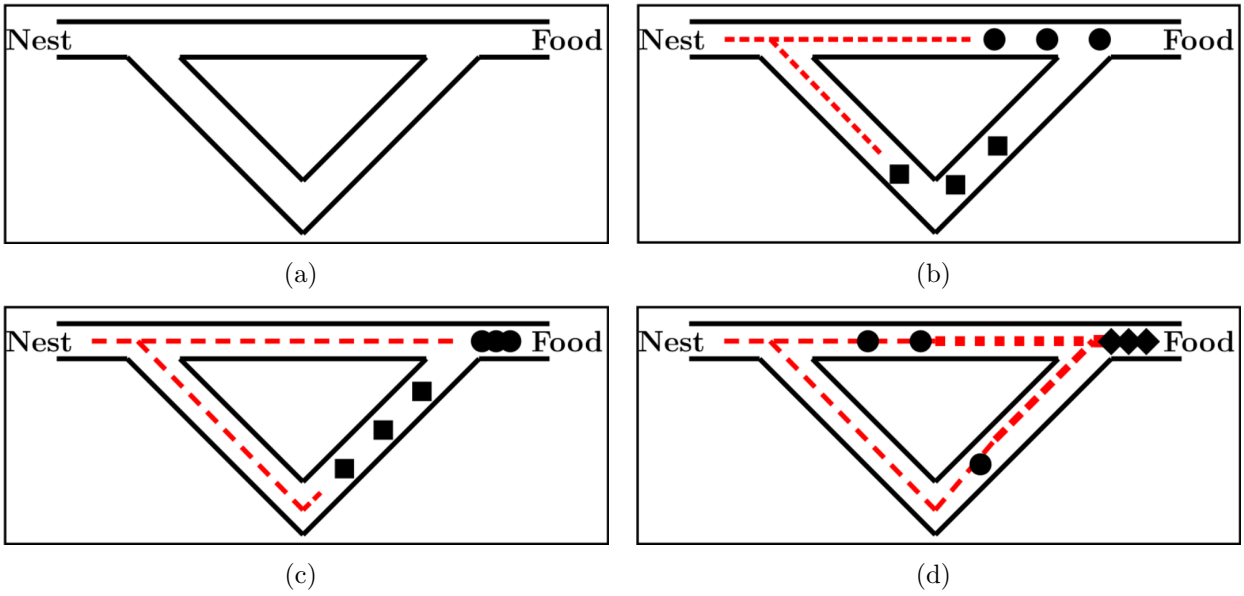


Figura 3.1: Como as formigas encontram o menor caminho [5].

Daqui para a frente, o termo "formiga" deverá ser entendido como formiga artificial, ou seja, uma entidade que possui inspiração nas formigas reais, porém aplicada a uma implementação computacional. O comportamento das formigas artificiais será de acordo com o problema de interesse, mesmo que tal atuação possa não fazer sentido no caso das formigas reais encontradas na natureza.

Para facilitar a compreensão de alguns pontos do algoritmo ACO, a serem mostrados a seguir, considere um problema exponencial relevante, o Caixeiro Viajante (*Traveling Salesman Problem* (TSP)). Dado um conjunto de cidades como entrada, uma solução candidata é dada por uma ordenação dessas cidades, indicando em que sequência elas devem ser visitadas a fim de que a viagem total possua o menor custo possível. Por exemplo, uma solução $s = \{2, 1, 3, 4\}$ indica que a ordem de visita das cidades deve começar pela de número 2, seguida pela cidade 1 e assim por diante. Em outras palavras,

dado um grafo com peso nas arestas $G = (V, E)$, cada vértice $v \in V$ representa uma cidade e cada aresta $e_{ij} \in E$ representa o custo para deslocar-se da cidade i em direção a j. Uma solução s representa um caminho de G.

3.1.1 Terminologia

A compreensão dos termos mostrados a seguir facilitará o entendimento do algoritmo ACO.

Formiga artificial: trata-se do conjunto de valores que representam uma solução do problema.

Conjunto de componentes (C)**:** conjunto de todos os valores possíveis para a criação de uma solução. Para o problema TSP, C representa o conjunto V de vértices do grafo.

Solução parcial (s^p)**:** denomina-se solução parcial aquela que possui parte dos componentes necessários para que uma formiga seja considerada uma solução completa. Para a solução $s = \{2, 1, 3, 4\}$ mostrada acima, uma possível solução parcial é $s^p = \{2, 1, 3\}$.

Passo da construção: conjunto de instruções responsável pela adição de um valor (componente) a uma solução parcial s^p.

Componente disponível: representa todos os possíveis componentes que ainda não fazem parte de uma solução e podem ser adicionados a uma s^p. Por exemplo: considere o problema TSP, com o conjunto de componentes (que também podem ser entendidos como cidades ou vértices do grafo) $C = \{1, 2, 3, 4\}$. O conjunto $N(s^p)$ de componentes disponíveis para uma solução parcial $s^p = \{2, 1\}$ é $N(s^p) = \{e_{13}, e_{14}\}$.

Trilha de feromônios: matriz composta por valores referentes aos caminhos que levam as novas soluções a tomarem rotas já antes utilizadas por soluções com potencial.

Função heurística (η)**:** mede a qualidade de um componente candidato a participar de uma solução parcial s^p. A função retorna um valor real denominado valor heurístico.

3.1.2 Pseudocódigo

O pseudocódigo do ACO é mostrado a seguir no algoritmo 1. F_{total} denota o total de formigas e N_{total} o total de vértices do grafo que representa a solução. O conjunto das melhores soluções encontradas é representado por S_{best}. τ indica o valor do feromônio do componente e_{ij} e η representa a função heurística. Primeiramente, são setados os valores dos parâmetros e iniciada a trilha de feromônios. Então, enquanto a condição de parada do algoritmo não for atingida, há a ocorrência de quatro eventos: construção das soluções, execução de ações opcionais, avaliação das soluções obtidas e atualização da trilha de feromônios. Esses eventos serão descritos a seguir.

Algoritmo 1: Otimização por Colônia de Formigas.

```
1  Inicializa parâmetros
2  Inicializa trilha de feromônios
3  while condição de parada não é atingida do
       // Para cada solução (formiga)
4      for f = 1 to F_total do
5          Adiciona o primeiro vértice em s_f^p (solução parcial da formiga f)
           // Para cada vértice adicional da solução
6          for i = 1 to N_total − 1 do
7              Escolhe de maneira probabilística um novo vértice j baseado em τ e η
8              Adiciona novo componente e_ij à solução parcial s_f^p
9          end
10     end
11     Executa ações opcionais
12     for f = 1 to F_total do
           // Avalia as soluções obtidas
13         Avalia S_f
14     end
15     if foram encontradas melhores soluções then
16         Atualiza S_best com as melhores soluções
17     end
18     Atualiza trilha de feromônios
19 end
```

3.1.2.1 Construção das soluções (ou formigas)

Inicialmente, cada formiga é iniciada como uma solução parcial vazia ($s_f^p = \emptyset$). O primeiro vértice da solução é escolhido de forma aleatória ou probabilística, de acordo com a im-

plementação. Em seguida, para cada uma das $\left(N_{total}-1\right)$ iterações, um novo componente é adicionado à s_f^p. Por exemplo, para o problema TSP, um componente candidato a ser inserido na formiga é uma aresta e_{ij}, indicando que o passo atual do caminho parte da cidade i para a cidade j. O impacto da adição desse componente é o peso da aresta e_{ij} (a distância entre as cidades i e j).

É válido observar que poderão existir alguns componentes inválidos. Novamente com base no TSP, considere que não há ligação direta entre duas cidades j e k. Logo, a aresta e_{jk} torna uma determinada solução s_f^p inválida.

A escolha de um determinado componente $e_{ij} \in N(s_f^p)$ é feita de forma probabilística. A forma com que tal probabilidade é calculada pode variar entre as implementações do ACO. Na versão utilizada pelo primeiro algoritmo do ACO, o *Ant System* (AS) [14], a probabilidade de escolha de e_{ij} é calculada de acordo com a expressão 3.1.

$$P(e_{ij}|s_f^p) = \frac{\tau_{ij}^{\alpha} \cdot \eta(e_{ij})^{\beta}}{\sum_{c_{il} \in N(s_f^p)} \tau_{il}^{\alpha} \cdot \eta(e_{il})^{\beta}}, \ \forall e_{ij} \in N(s_f^p) \tag{3.1}$$

τ_{ij} denota o valor do feromônio do componente e_{ij} e $\eta(e_{ij})$ refere-se ao valor da função heurística para e_{ij}. α e β, com $\alpha > 0$ e $\beta > 0$, determinam respectivamente o impacto que o valor de feromônio e a função heurística terão sobre a escolha do componente.

$P(e_{ij}|s_f^p)$ é calculado pelo quociente entre dois valores: o primeiro é o produto entre os valores de feromônio e da função heurística do componente e_{ij}. O segundo é o somatório do mesmo produto citado anteriormente, porém para todos os componentes disponíveis.

Por exemplo: suponha uma determinada solução formada pela sequência $s = \{s_1 = 1, s_2 = 4, s_3 = 3, i, \ldots, s_n\}$, com $n = 6$. Suponha que é desejado conhecer a probabilidade de o número 2 ser o próximo valor da sequência, ou seja, $s_4 = i = 2$. O último valor da sequência ($s_3 = 3$) e o valor 2 compõem o componente candidato a tornar parte da solução. Pela multiplicação do valor de feromônio relativo ao componente e_{32} com o valor da função heurística desse mesmo componente e incluídas as constantes α e β, é obtido o numerador: $\tau_{32}^{\alpha} \cdot \eta(e_{32})^{\beta}$. O denominador é uma soma dos valores do numerador, porém, em vez de se utilizar o componente candidato, são utilizados todos os componentes disponíveis. Assim, o denominador do exemplo é: $\tau_{32}^{\alpha} \cdot \eta(e_{32})^{\beta} + \tau_{35}^{\alpha} \cdot \eta(e_{35})^{\beta} + \tau_{36}^{\alpha} \cdot \eta(e_{36})^{\beta}$.

Outro aspecto que também costuma variar de acordo com a implementação é a função heurística. Essa função é a ligação entre o ACO e o problema que deseja-se resolver. Logo, tanto a função heurística quanto a trilha de feromônios compõem o cerne do ACO. Uma das funções heurísticas mais utilizadas é a do *Ant System* [14], mostrada na expressão 3.2. d_{ij} refere-se ao custo da relação entre i e j. No problema TSP, por exemplo, esse custo é a distância entre as cidades i e j. Logo, quanto maior a distância entre as duas cidades, menos interessante fica tomar esse caminho.

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{3.2}$$

3.1.2.2 Ações opcionais

Na linha 11 do algoritmo 1 há um "espaço" para ações que podem ou não ser executadas antes de avaliar as soluções obtidas. Uma estratégia muito utilizada no ACO e que vem obtendo bons resultados [18] é o uso de uma *busca local*, que consiste basicamente em alterar pequenas partes de uma solução com o objetivo de melhorá-la.

Um algoritmo muito utilizado para realizar buscas locais é o *Hill Climbing* [28]. O algoritmo funciona basicamente da seguinte maneira: a solução corrente é alterada. Caso a alteração melhore o resultado, o mesmo procedimento é realizado novamente porém com a nova solução. Essa computação é realizada durante um determinado número de vezes ou enquanto as alterações resultarem em melhores soluções. Caso seja de interesse, pode-se reiniciar o *Hill Climbing* de um ponto diferente [28].

O *Hill Climbing* pode ser um ótimo aditivo para melhorar, a um baixo custo, o desempenho de algoritmos como o ACO. Porém, se utilizado isoladamente, não é muito eficaz, uma vez que sofre de problemas como a estagnação em ótimos locais.

3.1.2.3 Atualização dos feromônios

A atualização dos feromônios trata de dois pontos: aumentar, na trilha de feromônios, os valores associados a soluções de interesse e diminuir os valores relativos a soluções ruins.

Geralmente, a diminuição dos valores é realizada através da simulação da evaporação

dos feromônios reais depositados pelas formigas encontradas na natureza. O aumento dos valores relativos às boas soluções é realizado por meio do acréscimo dos feromônios associados ao conjunto de soluções de interesse S_{upd}. A expressão 3.3 define como é a atualização dos níveis de feromônio. Ela inclui tanto a evaporação quanto o incremento das taxas da substância.

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta_{ij}^{k}, k \in S_{upd} \tag{3.3}$$

A primeira metade da expressão, $(1 - \rho) \cdot \tau_{ij}$, expressa a parte relativa à evaporação dos feromônios. O parâmetro ρ, com $\rho \in [0, 1)$ é chamado taxa de evaporação. Quanto maior o valor de ρ, mais rapidamente os feromônios evaporam.

A segunda metade da expressão refere-se ao incremento dos feromônios. m representa a quantidade total de formigas. Se um determinado componente e_{ij} pertence a alguma das soluções de S_{upd}, então o valor de τ_{ij} deve ser incrementado. Δ_{ij}^{k} é definido por:

$$\Delta_{ij}^{k} = \begin{cases} \frac{1}{L^k} & \text{, se o componente } e_{ij} \text{ faz parte da solução } k \\ 0 & \text{, caso contrário} \end{cases} \tag{3.4}$$

Onde L^k representa o custo da solução k.

3.1.3 Variações do ACO

A seguir, serão descritas as 3 principais variantes do algoritmo ACO: *Ant System* (AS), *Ant Colony System* (ACS) e *Max-Min Ant System* (MMAS).

O *Ant System* (AS) foi o primeiro algoritmo de ACO, proposto em [12]. Uma das principais características do AS é que todas as formigas são utilizadas para a atualização da trilha de feromônios. Essa implementação foi usada como a base para a descrição do ACO. Assim, a probabilidade de escolha de um novo componente já foi descrita anteriormente na expressão 3.1 bem como a maneira como a trilha de feromônios é atualizada, mostrada na expressão 3.3.

O *Ant Colony System* (ACS) foi proposto em [13]. A principal diferença deste algo-

ritmo é a atualização local dos feromônios. Ela é feita após cada passo da construção, por todas as formigas. A fórmula de atualização dos feromônios é mostrada pela expressão 3.5.

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0 \tag{3.5}$$

$\rho \in (0, 1)$ é a taxa de evaporação e τ_0 é o valor inicial do feromônio.

Do mesmo modo que foi mostrado pelo algoritmo 1, o ACS também possui uma etapa de atualização de feromônios no final de cada iteração. Ela é realizada apenas pela formiga que representa a melhor solução encontrada na iteração ou mesmo a melhor solução existente até então [15]. Há apenas uma pequena diferença: enquanto na equação 3.4 era atribuído o valor 0 caso o componente e_{ij} não fizesse parte da solução, no caso do ACS o valor do feromônio para τ_{ij} permanece inalterado.

Essa mudança na política de atualizações tem por objetivo diversificar os locais de busca. Uma vez que a concentração de feromônios é decrementada durante a construção do caminho, as outras formigas são menos influenciadas por rotas com maior quantidade de feromônios, o que aumenta a probabilidade delas enveredarem por outras regiões.

Outro aspecto diferente no ACS é relativo à regra de decisão utilizada durante a montagem da solução. Por exemplo: para o problema TSP, a probabilidade de uma formiga migrar de uma cidade i para j depende de dois fatores: o primeiro é uma variável q, aleatória e distribuída de maneira uniforme no intervalo $[0, 1]$. O segundo é o parâmetro q_0. Se $q \leq q_0$ então:

$$P_{ij}^k = \tau_{ij} \cdot \eta_{ij}^\beta \tag{3.6}$$

Os significados de η e β são os mesmos que os presentes na equação 3.1.

Caso $q > q_0$, a probabilidade de escolha de um determinado destino é dada pela expressão 3.1.

Uma outra variação do ACO, chamada *Max-Min Ant System* (MMAS), foi proposta por [30]. Duas foram as principais mudanças em relação a modelos anteriores. A primeira é que apenas as melhores soluções (formigas) são utilizadas para atualizar a trilha de

feromônios. A outra novidade é o estabelecimento de valores mínimo e máximo para a trilha de feromônios, a fim de evitar que a busca dirija-se para máximos locais.

A atualização a trilha de feromônios do MMAS é definida pela expressão 3.7.

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \Delta\tau_{ij}^{best} \tag{3.7}$$

$\Delta\tau_{ij}^{best}$, o valor de atualização dos feromônios, é definido pela expressão 3.8.

$$\Delta_{ij}^{best} = \begin{cases} \frac{1}{L^{best}} & , \text{ se o componente } e_{ij} \text{ faz parte da solução } k \\ 0 & , \text{ caso contrário} \end{cases} \tag{3.8}$$

L^{best} é o custo das melhores soluções. Tais soluções podem ser tanto a melhor encontrada na iteração quanto a melhor encontrada desde o início do algoritmo, ou mesmo uma combinação dessas duas soluções. A definição dos limites mínimo e máximo para a trilha de feromônios, segundo [30], devem ser definidos de forma experimental, de acordo com o problema a ser resolvido.

3.2 Outras estratégias

Organismos Multicelulares (*Multicellular Organisms*) [24] são formados por várias unidades independentes, denominadas células. Uma célula, nesse contexto, é uma unidade autônoma capaz de se comunicar com as demais por meio de mensagens. Dada uma certa tarefa a ser realizada, partes dela são distribuídas entre as várias células que compõem o organismo, de acordo com as capacidades e características de cada célula.

No algoritmo Otimização por Nuvem de Partículas (*Particle Swarm Optimization*) [23], as partículas, ou seja, as possíveis soluções de um problema, percorrem o espaço de busca na direção das melhores posições encontradas até o momento.

A inteligência coletiva é composta de outras estratégias além das descritas acima. Algumas delas podem ser encontradas em [1].

CAPÍTULO 4

TRABALHOS RELACIONADOS

A disseminação de informações em redes dinâmicas e descentralizadas tem sido bastante estudada ao longo dos últimos anos. Como métodos de referência nesse meio, dois métodos merecem destaque: a inundação (*flooding*) e a disseminação epidêmica (*gossip*).

4.1 Inundação (*flooding*)

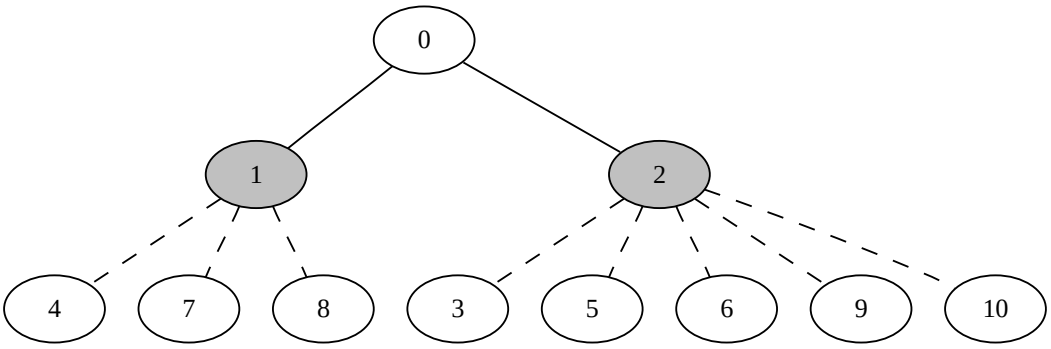
A inundação, mais conhecida como *flooding* [26] é um método que dissemina as mensagens de maneira confiável. As informações alcançam toda a rede da maneira mais rápida possível, uma vez que cada mensagem que um determinado nodo recebe é encaminhada para todos os vizinhos. Esses vizinhos, então, repassam a mensagem para os nodos com os quais estão ligados, e assim por diante, até que toda a rede possua conhecimento da informação. Cada nodo dissemina determinada informação apenas 1 vez.

Um exemplo de como os nodos disseminam as informações pela rede é mostrado na figura 4.1. Inicialmente, o nodo 0, mostrado em cinza, carrega informações inéditas, que serão repassadas aos seus vizinhos, nodos 1 e 2, conforme mostra a figura 4.1(a). A seguir, na figura 4.1(b), são mostrados os nodos 1 e 2, de cor cinza, já cientes do conteúdo disseminado pelo nodo 0. Esses dois nodos irão disseminar as informações para todos os seus vizinhos.

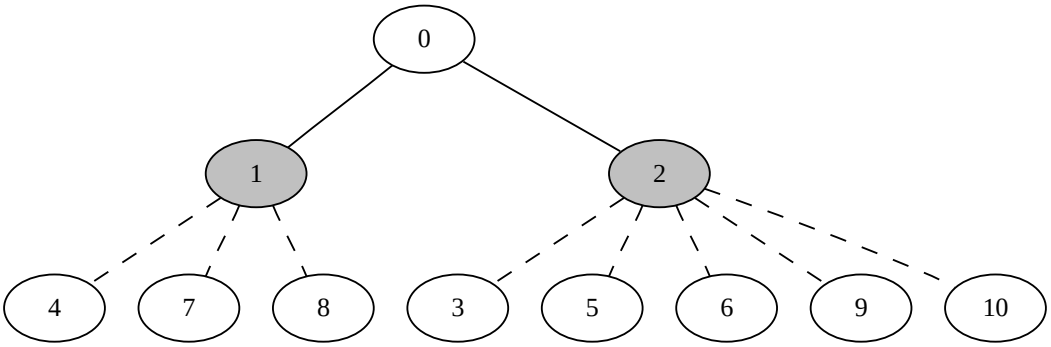
O algoritmo 2 apresenta as ações do nodo i no *flooding*. As novas informações presentes no nodo são retransmitidas para toda a vizinhança.

Algoritmo 2: *Flooding*: ações do nodo i

```
1 if i possui informações inéditas then
2 |   forall vizinho k do nodo i do
3 |   |   Envia informações para o vizinho k
4 |   end
5 end
```



(a) Nodo 0 repassa informações inéditas aos nodos 1 e 2.



(b) Nodos 1 e 2 retransmitem informações recebidas às suas respectivas vizinhanças.

Figura 4.1: Exemplo do funcionamento da inundação.

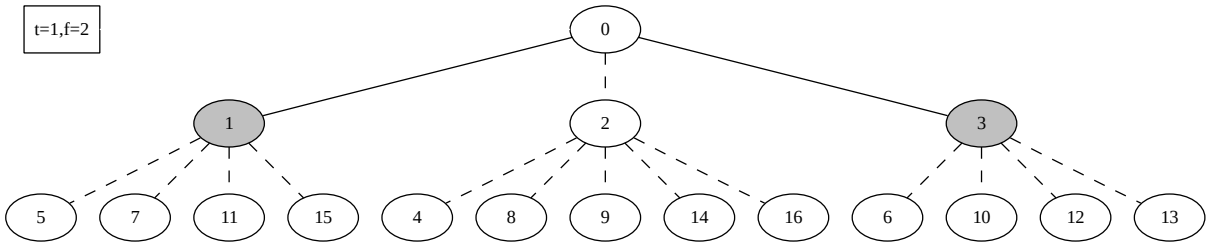
4.2 Difusão epidêmica

Uma estratégia para a difusão de informações em redes dinâmicas e descentralizadas é a que faz uso dos chamados algoritmos epidêmicos (*epidemic algorithms*), método também conhecido pelo nome de *gossip* [16]. Tais algoritmos buscam inspiração na natureza, mais especificamente no comportamento apresentado pela disseminação de epidemias e rumores. Na natureza, uma epidemia é transmitida de um ser infectado para um ou mais seres ainda não contaminados. Então, os novos indivíduos contaminados a retransmitem para outros ainda não infectados, e assim por diante. Uma das características das epidemias é a sua permanência no meio mesmo após a morte de vários seres infectados. Esta é uma característica de interesse para aplicações computacionais que funcionam sob ambientes dinâmicos e descentralizados.

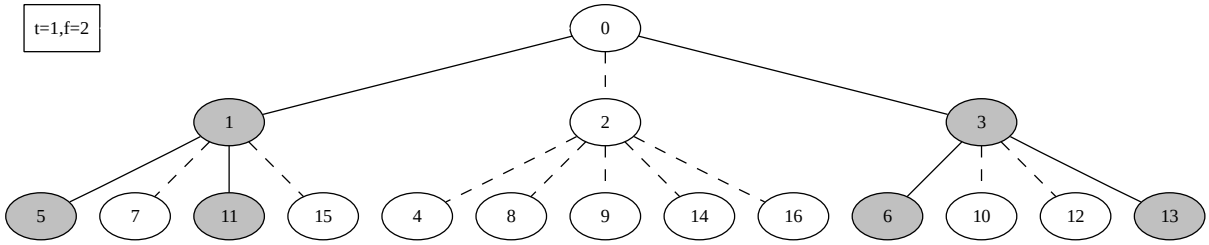
Nos algoritmos epidêmicos, cada integrante da rede transmite a informação recebida para uma certa quantidade de destinos selecionados aleatoriamente [16]. Os nodos que receberem tais informações as reenviam para outro conjunto de nodos também selecionados

de maneira aleatória e assim sucessivamente.

Um exemplo desse processo é mostrado na figura 4.2. Primeiramente, na figura 4.2(a), o nodo 0 sorteia $f = 2$ vizinhos, nodos 1 e 3, marcados em cor cinza, como destino das informações. Após tais nodos receberem as informações vindas do vértice 0, cada um deles sorteia os próximos $f = 2$ destinos para as informações recebidas, conforme mostrado na figura 4.2(b).



(a) Nodo 0 sorteia os vizinhos 1 e 3 para disseminar informações.



(b) Nodos 1 e 3 sorteiam próximos destinos das informações.

Figura 4.2: Exemplo de disseminação epidêmica.

As áreas de aplicação dos algoritmos epidêmicos são variadas, tais como tolerância a falhas [21] e replicação de bases de dados [10]. Uma vez que a disseminação epidêmica distribui os dados de forma seletiva, ao contrário de métodos que espalham a informação de forma confiável [20], as aplicações que fazem uso de disseminação epidêmica economizam recursos da rede.

4.2.1 Parâmetros

O funcionamento de um algoritmo epidêmico é basicamente determinado por três parâmetros. Cada mensagem recebida por um nodo é armazenada em um buffer de capacidade b. Essa mensagem recebida é retransmitida t vezes, uma a cada rodada. Cada retransmissão possui como destino uma quantidade f, denominada *fan-out*, de nodos selecionados aleatoriamente [16].

As variações dos algoritmos epidêmicos são geralmente diferenciadas pelos valores dos três parâmetros, b, t e f. Esses valores podem ser estabelecidos sem levar em conta a quantidade n de nodos da rede. A confiabilidade da disseminação das informações é atrelada aos valores setados para os parâmetros. Para uma melhor adaptação do sistema à mudança do tamanho n da rede, b, t e f podem ter os valores modificados de acordo com n [16].

4.2.2 Variações

Dentre as variações de algoritmos epidêmicos existentes, duas delas originam-se ao variar o valor t referente ao número de rodadas que um determinado nodo permanece "infectado", disseminando uma determinada informação recebida. Em um extremo, no modelo chamado de *infect and die*, o nodo dissemina um certo dado que recebeu por apenas uma rodada. Mesmo que uma certa informação seja recebida novamente pelo nodo, ela não é repassada. O outro extremo ocorre quando o nodo espalha a informação ao longo de todo o tempo em que estiver ativo, modelo conhecido pelo nome de *infect forever* [16].

4.2.3 Métricas

Para se avaliar a qualidade dos algoritmos epidêmicos, algumas métricas foram definidas.

A proporção Y_r de processos infectados após uma quantidade r de rodadas é dada pela expressão 4.1 [16].

$$Y_r = \frac{Z_r}{n} \tag{4.1}$$

Onde Z_r denota a quantidade de processos que receberam determinada informação após r rodadas e n representa o total de nodos da rede.

No modelo *infect forever*, considerando que cada nodo tenta contaminar f outros nodos por rodada, a proporção de processos infectados após r rodadas é aproximada pela expressão 4.2 [4].

$$Y_r \approx \frac{1}{1 + ne^{-fr}} \tag{4.2}$$

Ou seja, a proporção entre indivíduos infectados e não infectados cresce exponencialmente por, em média, um fator de e^f a cada rodada.

Denomina-se *latência* o tempo necessário para uma determinada informação alcançar todos os integrantes do sistema. No modelo *infect and die*, para que as informações alcancem todo o sistema, a quantidade f de destinos que os nodos repassam a informação deve ser da ordem de $\log(n)$. Considerando que f seja da magnitude citada anteriormente, o número de R de rodadas necessárias para infectar todo o sistema é dado pela expressão 4.3 [6].

$$R = \frac{\log(n)}{\log\big(\log(n)\big)} + O(1) \tag{4.3}$$

No modelo *infect forever*, considerando que cada nodo dissemine informação para f destinos aleatórios, a latência R é dada pela expressão 4.4 [29]:

$$R = \log_{f+1}(n) + \frac{\log(n)}{f} + O(1) \tag{4.4}$$

Através das duas expressões acima é possível observar que as informações alcançam a totalidade do sistema rapidamente, uma vez que, no pior caso, é necessária uma quantidade de ordem logarítmica de rodadas para a disseminação ser completa.

4.2.4 Gerência do buffer

Um ponto que merece atenção nos algoritmos epidêmicos é o que se refere a que tipo de informação guardar no buffer. Dependendo do tamanho b desse buffer e da taxa de informações recebidas pelo nodo, o limite pode ser insuficiente para armazenar todos os dados. Assim, alguma maneira eficaz de selecionar quais informações devem ser descartadas e quais devem permanecer no buffer se faz necessária.

Duas alternativas surgem em uma primeira análise. A primeira é descartar, quando o limite do buffer for alcançado, todas as novas mensagens que chegarem ao nodo. Essa estratégia possui o grave inconveniente de as novas informações não serem repassadas aos demais integrantes da rede. Por outro lado, se as mensagens mais antigas do buffer forem

as escolhidas para o descarte, corre-se o risco de não tê-las encaminhado um número suficiente de vezes [16].

Uma alternativa interessante é classificar as informações do buffer de acordo com o número de vezes que foram retransmitidas. Assim, dá-se preferência para eliminar mensagens que tiverem o maior número de retransmissões [16]. Procedendo dessa maneira, diminui-se consideravelmente as chances de descartar dados que ainda não foram suficientemente disseminados.

4.2.5 O algoritmo

As ações executadas por um determinado nodo i na disseminação epidêmica são mostradas pelo algoritmo 3

Algoritmo 3: *Gossip*: ações do nodo i

```
1 forall mensagem m presente no nodo i do
2 |  if m foi enviada menos de t vezes, uma por rodada then
3 |  |  for vizinho escolhido aleatoriamente k = 1 to f do
4 |  |  |  Cria mensagem m_k com informações do nodo i
5 |  |  |  Envia m_k ao nodo k
6 |  |  end
7 |  end
8 end
```

4.3 Disseminação com uso de inteligência coletiva

Alguns métodos têm utilizado inteligência coletiva para realizar determinadas tarefas em redes *ad hoc* e *peer-to-peer*. Aqui são citados aqueles que possuem maior relação com a estratégia proposta neste trabalho. Nenhum desses métodos é diretamente comparável pelas diferenças de asserções e/ou objetivos. Em [8], com o objetivo de localizar recursos em *grid*, a disseminação de informações ocorre através de formigas especializadas em várias funções, tais como a manutenção de enlaces em situações de baixo tráfego. Um ponto importante a ser destacado neste trabalho é que ele cria formigas quando o estado dos enlaces dos nodos é alterado. O algoritmo altera a topologia da rede, isto é, cria e

remove enlaces, de modo a manter uma rede fracamente conexa.

O trabalho [2] também trata sobre disseminação de agentes em redes dinâmicas, porém sem utilizar a abordagem da Otimização por Colônia de Formigas. Segundo os autores, as estratégias baseadas em formigas até então apresentadas continuavam a disseminar as formigas mesmo em situações nas quais não havia novidades a serem disseminadas. Os agentes são hierarquizados em dois tipos. Um tipo de agente percorre a rede em busca de informações e as informa ao outro tipo. Há uma diferença conceitual neste trabalho, uma vez que existe a hierarquização dos nodos em dois tipos.

A disseminação de informações através de formigas é utilizada em [3] para auxiliar no roteamento. Tabelas cache em cada nodo colaboram para as mensagens enviadas encontrarem o destino. Este trabalho também possui formigas especializadas em realizar determinadas tarefas. A circulação de pacotes na rede é permanente, com o objetivo de manter as tabelas cache dos nodos atualizadas.

O roteamento também é tratado em [19], especificamente utilizando Otimização por Colônia de Formigas. Porém, este trabalho possui um componente centralizador, uma vez que um determinado nodo é que concentra as informações sobre as rotas da rede.

Em [32] é proposto um protocolo para a descoberta da topologia em redes *ad hoc* através do uso de agentes móveis. Uma taxa de mudança dos vizinhos de um nodo foi estabelecida. Quanto maior for essa taxa, maior chance que os agentes terão de migrarem para aquele nodo. Por outro lado, nodos cuja vizinhança pouco muda ou muito visitados destroem os agentes que ali chegam.

O trabalho [27] trata sobre a descoberta e disseminação da topologia de redes dinâmicas e descentralizadas, utilizando para isso inteligência coletiva, mais especificamente a Otimização por Colônia de Formigas. Informações mantidas tanto pelos nodos quanto pelas formigas, após passarem de uma determinada idade, são descartadas. O algoritmo dissemina formigas continuamente pela rede. Além disso, no modelo de sistema deste trabalho, os relógios locais de cada nodo são sincronizados entre si, de modo que a diferença entre dois instantes de tempo físico é a mesma para todos os nodos. Os conceitos de estigmergia e método de disseminação de informações apresentados pelo al-

goritmo *EvAnt* (capítulo 5) foram adaptados deste trabalho. Uma comparação direta dos dois métodos não foi possível pois cada estratégia baseia-se em um modelo de sistema diferente.

CAPÍTULO 5

O ALGORITMO *EVANT*

O algoritmo *EvAnt* faz uso de inteligência coletiva para disseminar, através de formigas, novidades em redes dinâmicas e descentralizadas. Tais redes possuem comportamento imprevisível e podem sofrer alterações (saída e entrada de nodos e links) frequentes.

5.1 Disseminação de novidades baseada em eventos

O algoritmo *EvAnt* é disparado com a ocorrência de um evento. Denomina-se *evento*, ou *novidade*, alguma alteração no estado de um componente da rede. Uma alteração corresponde à criação ou eliminação de um enlace bem como o surgimento ou a saída de algum nodo da rede. Após a ocorrência de um evento, os nodos que o *detectam* são aqueles que iniciam a *disseminação* das informações. Os nodos que inicialmente tornam-se cientes dos eventos e disparam a disseminação diferem de acordo com o tipo de evento:

surgimento de um nodo: a disseminação da novidade inicia-se pelo próprio nodo que acabou de surgir, bem como pelos nodos que são seus vizinhos;

saída de um nodo: os vizinhos do nodo que saiu da rede iniciam a disseminação;

criação de um enlace: os nodos que estão ligados pelo novo enlace iniciam a propagação do evento;

eliminação de um enlace: os nodos que estavam ligados pelo enlace eliminado iniciam a propagação da novidade.

5.2 Modelo do sistema

O ambiente em que o algoritmo é executado é uma rede dinâmica e descentralizada, como são as redes *peer-to-peer* não estruturadas e redes *ad hoc*. A entrada e saída de *hosts* pode

ser intensa, de modo que as mudanças no estado da rede são frequentes e imprevisíveis. Outro ponto a ser observado é a total ausência de componentes que centralizam a tarefa de coletar e disseminar as novidades.

O modelo de sistema a ser considerado no algoritmo *EvAnt* é definido como um grafo não direcionado [27] $G(t) = (V_t, E_t)$. Tal grafo é composto, em um instante de tempo t, por um conjunto de vértices V_t e um conjunto de arestas E_t. Por ser definido em função do tempo, pode ocorrer que, para dois instantes de tempo físico distintos t_1 e t_2, $G(t_1) \neq G(t_2)$.

As seguintes características definem o modelo de sistema considerado.

1. Os nodos sempre possuem, para um dado instante de tempo t, o conhecimento correto sobre o estado dos vizinhos, isto é, dos vértices aos quais estão diretamente conectados através de um único enlace.

2. O envio de mensagens é livre de erros. O tempo para o envio de informações é não desprezível e não há limite de tempo definido para que uma mensagem chegue ao destino.

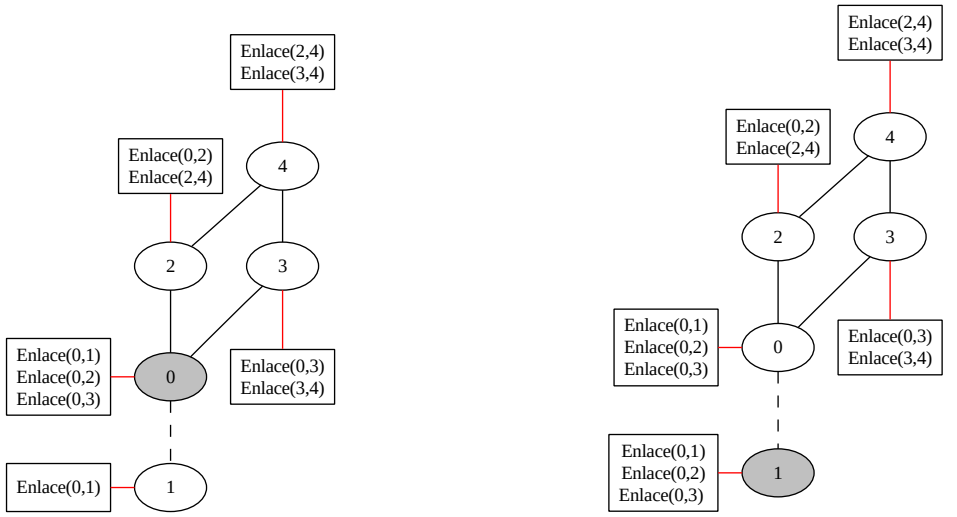
3. Não há relógio global compartilhado pelos nodos. Cada nodo possui apenas um relógio local e os diversos relógios locais dos nodos não são sincronizados entre si. Logo, a diferença entre dois instantes de tempo físico t e $(t+1)$ não é a mesma para os vários participantes da rede.

4. O grafo que representa o sistema é não direcionado. Os enlaces são bidirecionais, ou seja, $e_{ijt} = e_{jit}$.

5.3 Disseminação das formigas

As informações são disseminadas pela rede através de formigas. A formiga, no contexto do algoritmo *EvAnt*, é uma abstração computacional de um agente móvel, ou seja, de uma mensagem que carrega as novidades pela rede. Um exemplo de como as formigas são transmitidas entre os nodos é mostrado na figura 5.1. Na primeira figura, 5.1(a),

a formiga localiza-se no nodo 0, mostrado em cor cinza. Neste caso, a formiga carrega informações relativas aos enlaces que o nodo 0 possui, conforme mostrado no quadro à esquerda do nodo. O destino escolhido pela formiga é o nodo 1, caminho mostrado pelo enlace pontilhado. Na segunda figura, 5.1(b), o nodo 1 já foi atualizado com as informações trazidas pela formiga.



(a) Formiga (localizada no nodo cinza) migra do nodo 0 para o 1, carregando as informações do nodo 0.

(b) Nodo 1 é atualizado com as informações trazidas pela formiga.

Figura 5.1: Disseminação de informações pela formiga.

A disseminação das novidades, com o objetivo de permitir que as formigas consigam transmitir as informações para toda a rede, utiliza a abordagem baseada na estigmergia. Cada nodo contém um depósito de feromônios associado a cada um dos enlaces que possui. Quando uma formiga viaja de um nodo i em direção a algum vizinho j, a concentração $C_{ij}(t)$ de feromônios daquele enlace é incrementada em uma unidade, conforme mostra a expressão 5.1 [27].

$$C_{ij}(t) = C_{ij}(t-1) + 1 \tag{5.1}$$

Os feromônios também são utilizados para auxiliar na escolha do próximo destino das formigas. Essa escolha é feita de forma probabilística: enlaces com menor concentração de feromônios possuem maior probabilidade de serem escolhidos, ao contrário do ACO

padrão, no qual os melhores caminhos possuem as maiores concentrações de feromônio. A expressão 5.2 [27] define a probabilidade de uma formiga migrar do nodo i para o j no instante de tempo t. O tamanho da vizinhança do nodo i é representado por $|e_{it}|$ e $E_{it}(k)$ denota o k-ésimo vizinho do nodo i no tempo t. A constante α, tal que $\alpha > 0$, representa a força dos feromônios. Maior o valor de α, maior a diferença entre as probabilidades obtidas. Os feromônios, assim, servem para "afastar" as formigas umas das outras, fazendo com que elas sejam melhor distribuídas pela rede.

$$P_{ij}(t) = \frac{\left(C_{ij}(t) + 1\right)^{-\alpha}}{\sum_{k=1}^{|e_{it}|} \left(C_{iE_{it}(k)}(t) + 1\right)^{-\alpha}} \tag{5.2}$$

Com o passar do tempo, os feromônios enfraquecem. O enfraquecimento dos feromônios é mostrado pela expressão 5.3 [27]. A nova concentração $C_{ij}(t)$ de feromônios do enlace e_{ij} no tempo t é dada em função da concentração do enlace em um instante de tempo anterior t_0 e uma constante ρ, tal que $0 \leq \rho < 1$. Essa constante controla a velocidade da evaporação. Maior o ρ, mais rapidamente os feromônios evaporam.

$$C_{ij}(t) = C_{ij}(t_0) \cdot (1 - \rho)^{(t - t_0)} \tag{5.3}$$

5.4 Controle da população de formigas

No algoritmo *EvAnt*, a criação e descarte de mensagens da rede é realizada através da criação e destruição de formigas, respectivamente. A quantidade de formigas na rede possui grande impacto no desempenho do algoritmo. O excesso de formigas faz com que as mesmas informações sejam repetidamente disseminadas pela rede. Por outro lado, com um número insuficiente de formigas as novidades não conseguem alcançar todos os nodos. Para controlar a população de formigas de maneira dinâmica, também são utilizados feromônios.

A concentração total $C_i(t)$ de feromônios de um nodo i é dada pela a soma dos valores de feromônio de cada enlace desse nodo, conforme mostrado na expressão 5.4 [27].

$$C_i(t) = \sum_{k=1}^{|e_{it}|} C_{iE_{it}(k)}(t) \tag{5.4}$$

$C_i(t)$ é então comparado aos limites mínimo $\big(L_{min}(t)\big)$ e máximo $\big(L_{max}(t)\big)$, definidos pelas expressões 5.5 e 5.6 [27] respectivamente. A constante δ, tal que $\delta \geq 0$, define o papel do tamanho da vizinhança sobre a definição dos limites. À medida que o valor de δ aumenta, maior a influência que o tamanho da vizinhança terá sobre os valores de $L_{min}(t)$ e $L_{max}(t)$. As constantes γ_{min} e γ_{max} devem ser escolhidas de modo que $\gamma_{min} > 0$ e $\gamma_{max} > \gamma_{min}$.

$$L_{min}(t) = \gamma_{min} \cdot |e_{it}|^{\delta} \tag{5.5}$$

$$L_{max}(t) = \gamma_{max} \cdot |e_{it}|^{\delta} \tag{5.6}$$

Caso $C_i(t)$ seja menor ou igual a $L_{min}(t)$ e o nodo tenha recebido uma formiga que traga novidades, uma formiga é criada pelo nodo. Essa formiga carregará inicialmente as mesmas informações que aquele nodo possui. Se, por outro lado, $C_i(t)$ for superior ou igual a $L_{max}(t)$, todas as formigas localizadas no nodo são destruídas.

5.5 Especificação do algoritmo

A estratégia proposta é composta de duas partes principais: a primeira, apresentada no algoritmo 4 (*EvAnt*: ciclo do nodo), descreve o comportamento de um nodo i. O algoritmo 5 (*EvAnt*: ciclo da formiga), por sua vez, mostra as ações executadas pela formiga f.

Inicialmente, no algoritmo 4, um novo evento faz com que os nodos que detectaram o acontecimento criem formigas para disseminar a novidade pela rede. Os nodos que detectaram o evento criam formigas em toda rodada na qual a concentração local de feromônios $C_i(t)$ for menor ou igual ao limite mínimo $L_{min}(t)$. Nodos que receberem formigas com novidades também criam uma nova formiga a cada rodada, enquanto a

condição $C_i(t) \leq L_{min}(t)$ for respeitada.

Caso, em alguma rodada, o nodo i possua a concentração local de feromônios $C_i(t)$ maior ou igual ao limite superior $L_{max}(t)$, todas as formigas que no momento estão no nodo i são destruídas.

O ciclo das formigas, mostrado no algoritmo 5, é composto por 5 ações. Inicialmente, a formiga, vinda do nodo i, chega ao nodo j e incrementa a concentração de feromônios relativa ao enlace (i, j). Em seguida, a formiga atualiza o nodo com as novidades que carrega consigo. Após a formiga ficar inativa ("dormir") no nodo por um determinado período de tempo, que é parâmetro do algoritmo, a formiga então atualiza-se com informações que ela não possui. Por fim, a formiga seleciona o próximo destino, de acordo com a expressão 5.2.

Algoritmo 4: *EvAnt*: ciclo do nodo i

```
1  while true do
2  |  evapora feromônios → C_ij(t) = C_ij(t_0) · (1 − ρ)^(t−t_0)
3  |  if i detectou evento or (i recebeu formiga que carrega novas informações and
   |  C_i(t) ≤ L_min(t)) then
4  |  |  cria nova formiga
5  |  |  escalona ciclo para a nova formiga (Algoritmo 5)
6  |  end
7  |  else
8  |  |  if C_i(t) ≥ L_max(t) then
9  |  |  |  destrói todas as formigas localizadas no nodo i
10 |  |  end
11 |  end
12 end
```

Algoritmo 5: *EvAnt*: ciclo da formiga f

```
1  while true do
2  |  formiga chega ao nodo j, vinda do nodo i → C_ij(t) = C_ij(t − 1) + 1
3  |  formiga atualiza informações do nodo
4  |  formiga fica inativa no nodo por um determinado tempo
5  |  formiga atualiza-se com informações do nodo
6  |  formiga seleciona o próximo destino → P_ij(t) = (C_ij(t)+1)^(−α) / Σ_{k=1}^{|e_it|} (C_{iE_it(k)}(t)+1)^(−α)
7  end
```

A fim de que os nodos e formigas saibam quais informações são as mais atuais e, assim,

devem ser assimiladas, passos mostrados nas linhas 4 e 5 do algoritmo 5, as informações transmitidas possuem *timestamps*.

Considere, por exemplo, que as informações transmitidas são referentes ao estado dos nodos da rede, com cada nodo podendo assumir dois estados: ativo ou inativo. Assim, inicialmente, quando um nodo i sabe que um determinado vizinho j está ativo, o *timestamp* contido na descrição do nodo i relativo ao enlace (i, j) possui o valor 0. Se em algum momento o nodo j ficar inativo, o *timestamp* no nodo i relativo ao enlace (i, j) será 1. Caso o nodo j volte a ficar ativo na rede, a descrição contida no nodo i terá o *timestamp* relativo ao enlace (i, j) com o valor 2, e assim por diante.

Através desse método, caso algum nodo sofra duas mudanças de estado consecutivas e os demais nodos não sejam informados separadamente de tais mudanças, as formigas atualizarão os nodos com o estado correto sobre os nodos alterados.

O tempo de inatividade das formigas (linha 4 do algoritmo 5) auxilia a determinar o "dinamismo" da atualização das informações nos nodos. Um intervalo muito curto acarreta maior movimentação das formigas pela rede, podendo as mesmas informações serem repetidamente disseminadas. Por outro lado, um tempo muito longo implica em os nodos ficarem um longo tempo sem receber as novidades, ou, alternativamente, as formigas carregarem informações já desatualizadas.

CAPÍTULO 6

SIMULAÇÕES E RESULTADOS

Ao longo deste capítulo serão apresentados resultados de simulações realizadas para validar e avaliar o algoritmo proposto. Será mostrada uma comparação de resultados entre o algoritmo *EvAnt*, a inundação (*flooding*) [26] e a disseminação epidêmica (*gossip*) [16]. Os três algoritmos são utilizados na disseminação de informações sobre um evento detectado. Ambos os algoritmos foram escolhidos para a comparação por serem métodos de referência em disseminação de informações. As simulações possuem como objetivo observar a eficiência e eficácia dos algoritmos na atualização das informações da rede, baseado na detecção de eventos.

6.1 Descrição das simulações

As simulações foram realizadas com o uso da biblioteca *SMPL (Simulation Programming Language)* [25] da linguagem *C*. O código foi desenvolvido na linguagem *C++*. Duas máquinas foram utilizadas para realizar as simulações. A primeira delas, utilizada para a grande maioria dos casos, é composta por 2 processadores Quad-Core Intel Xeon E5430, rodando a 2.66GHz, com 128KB de cache L1 (32KB por núcleo) e 6MB de cache L2 em cada CPU. A memória total é de 8GB. As simulações foram realizadas utilizando-se Debian GNU Linux, *kernel 2.6.24-1-686-bigmem*, com o compilador g++ (Debian 4.3.4-6) 4.3.4.

Para algumas simulações, que necessitavam de uma quantidade maior de memória, foram utilizadas as máquinas do CT-INFRA/UFPR. A configuração da máquina utilizada é formada por 8 processadores Dual Core AMD Opteron Processor 865, com frequência de 1,8GHz, cache L1 de 128KB (64KB por núcleo) e 1MB de memória cache L2 cada. A memória total é de 16GB, com a disponibilidade de se alocar até 8GB. O sistema operacional utilizado foi o Debian GNU Linux, *kernel 2.6.32.9*, com o compilador g++

(Debian 4.4.4-11) 4.4.5 20100824.

As simulações foram conduzidas conforme descrito a seguir. Inicialmente, todos os nodos da rede possuem, em suas descrições locais, informações corretas sobre o estado de todos os nodos da rede. Então, a partir desse estado, eventos são disparados ao longo do tempo e informações relativas ao processo da detecção desses eventos são coletadas. As métricas escolhidas possuem como objetivo avaliar a velocidade de detecção dos acontecimentos bem como o peso que cada método impõe à rede para todos os nodos ficarem cientes dos eventos.

Para avaliar a velocidade da detecção de eventos, foram utilizadas duas métricas: latência (*latency*) e a taxa de ciência (*awareness rate*).

A latência (*latency*) é definida como o número de rodadas necessárias para que todos os eventos sejam detectados por todos os nodos. A taxa de ciência (*awareness rate*), métrica proposta para este trabalho, tem como função mostrar a quantidade de nodos cientes dos eventos ao longo do tempo. Esta taxa pode ser obtida através da expressão 6.1, onde μ_i denota o número de nodos cientes do i-ésimo evento e ε representa o conjunto de eventos. Caso não tenha ocorrido eventos na rede ($|\varepsilon| = 0$), a métrica não se aplica.

$$awareness\ rate = \frac{\sum_{i=1}^{|\varepsilon|} |\mu_i|}{|\varepsilon|} \tag{6.1}$$

Por exemplo: para uma rede com 100 nodos na qual foram disparados 3 eventos ($\varepsilon = 3$), a taxa de ciência é dada pela expressão 6.2.

$$awareness\ rate = \frac{(|\mu_1| + |\mu_2| + |\mu_3|)}{3} \tag{6.2}$$

As duas métricas utilizadas para medir a carga imposta pelo método de detecção de eventos são o número de mensagens trocadas (*# messages exchanged*) e a quantidade de mensagens criadas (*# agents created*).

O número de mensagens trocadas (*# messages exchanged*) representa a quantidade de formigas que se deslocaram de um nodo a outro, mais uma vez considerando um determinado intervalo de tempo. A outra métrica, número de mensagens criadas

(*# agents created*), denota a quantidade de novas formigas criadas ao longo do processo de detecção de eventos.

Redes de topologia *powerlaw* [17] e aleatória foram utilizadas. Uma rede de topologia aleatória possui os enlaces escolhidos de maneira aleatória, sendo conexa e cada nodo com ao menos um enlace. As redes aleatórias possuem uma distribuição mais uniforme dos graus de seus nodos, se comparadas às redes *powerlaw*. Foram utilizadas redes de 1000, 2000 e 2500 nodos. Para cada tamanho, 20 redes foram geradas, 10 com topologia *powerlaw* e outras 10 com topologia aleatória. As redes *powerlaw* foram criadas de acordo com o seguinte critério [9]: $m_0 = 20\%$ do tamanho da rede, onde m_0 representa o conjunto de nodos fortemente conectados, e sementes que variam de 0 a 9. As redes de topologia aleatória possuem o mesmo número de nodos e enlaces que as redes *powerlaw*.

As simulações tiveram tempo limite de 50 unidades de tempo. As rodadas, definidas como o intervalo no qual as métricas são coletadas, possuem duração de 0.2 unidades de tempo.

Para cada uma das redes, um conjunto aleatório de eventos correspondente a 1% do tamanho da rede foi gerado. Esse conjunto de eventos foi estabelecido de modo que, após a sua ocorrência, a rede continue conexa. Os eventos possíveis são a criação e remoção de nodos e enlaces. Cada evento é disparado em intervalos de 5 rodadas (1 unidade de tempo).

A simulação do algoritmo *EvAnt* termina quando todos os eventos forem detectados (*awareness rate* igual ao tamanho da rede) ou o tempo limite da simulação for atingido. O algoritmo *EvAnt* foi comparado com a inundação (*flooding*) e a disseminação epidêmica (*gossip*).

No *flooding* [26], a disseminação de informações inicia-se a partir do ponto de origem dos eventos. Quando um nodo receber uma mensagem que traga novidades às informações contidas naquele nodo, essas informações locais são atualizadas bem como a mensagem de modo que ambos possuam as mesmas informações. Essa nova mensagem é então disseminada por inundação a partir do nodo. Os termos mensagem e formiga referem-se à mesma entidade, responsável por disseminar os eventos pela rede.

No *flooding* um nodo repassa uma mensagem para todos os vizinhos inclusive àquele que acabou de enviá-la ao nodo. A mensagem atualizada pode conter informações que o vizinho não possui. Logo, a mensagem atualizada deve ser repassada a todos os vizinhos. O algoritmo termina após uma rodada na qual nenhum nodo disseminou novas mensagens.

O pseudocódigo do *flooding* simulado é mostrado a seguir, no algoritmo 6. Os nodos que detectam eventos ou recebem mensagens que tragam novidades criam cópias da mensagem e as disseminam entre todos os vizinhos.

Algoritmo 6: *Flooding simulado*

```
1  while Algum nodo disseminar novas mensagens na rodada do
2      forall nodo i do
3          if (i detectou evento or i recebeu mensagem que traga novas informações)
           then
4              Atualiza nodo i com a mensagem m
5              Atualiza mensagem m com informações do nodo i
6              forall vizinho k do nodo i do
7                  Cria mensagem m_k com as mesmas informações do nodo i
8                  Envia m_k para o nodo k
9              end
10         end
11     end
12 end
```

No *gossip* [16], a disseminação de informações também é iniciada no ponto onde os eventos são originados. Os nodos que recebem mensagens contendo novas informações são atualizados com as novidades e atualizam a mensagem de modo que ambos contenham as mesmas informações. O nodo então dissemina essa mensagem para $f \geq 1$ vizinhos escolhidos aleatoriamente. Tal mensagem é disseminada pelo nodo durante $t \geq 1$ rodadas. Os parâmetros t e f são os mesmos para todos os nodos da rede. O pseudocódigo do *gossip* simulado é mostrado no algoritmo 7.

O *gossip* utilizado neste trabalho possui como parâmetros $f = \log_{10}(N)$ e $t = 1$. Esse valor foi escolhido de acordo com o critério explicado em 4.2.3.

Os parâmetros utilizados para o algoritmo *EvAnt* são mostrados a seguir:

Força dos feromônios (α)**:** 8

Algoritmo 7: *Gossip simulado*

```
1  while Algum nodo disseminar novas mensagens na rodada do
2      forall nodo i do
3          if (i detectou evento or i recebeu mensagem que traga novas informações)
           then
4              Atualiza nodo i com a mensagem m
5              Atualiza mensagem m com informações do nodo i
6          end
7          forall mensagem m presente no nodo i do
8              if m foi enviada menos de t vezes, uma por rodada then
9                  for vizinho escolhido aleatoriamente k = 1 to f do
10                     Cria mensagem m_k com informações do nodo i
11                     Envia m_k ao nodo k
12                 end
13             end
14         end
15     end
16 end
```

Taxa de evaporação dos feromônios (ρ)**:** 0.4

Peso da vizinhança (δ)**:** 0.5

Limite inferior para a concentração local de feromônios (γ_{min})**:** $\frac{\frac{\log(N)}{4}}{|\text{vizinhança média}|^{\delta}}$

Limite superior para a concentração local de feromônios (γ_{max})**:** $100 \times \gamma_{min}$

Tempo de inatividade das formigas: 0.2 segundos (1 rodada).

Para obter uma comparação com o *gossip*, foi estipulado que o *EvAnt* deve disseminar mensagens por, em média, $\log_{10}(N)$ rodadas. Assim, o limite mínimo $L_{min}(t)$ da concentração de feromônios foi fixado como $\log_{10}(N)$, então, o valor da constante γ_{min} foi obtido. Devido à evaporação dos feromônios, deve-se diminuir o valor de $\log_{10}(N)$ a fim de que os nodos não disseminem mensagens por mais do que $\log_{10}(N)$ rodadas. Daí a escolha do valor $\frac{\log_{10}(N)}{4}$. Com γ_{min} fixado, diversos valores para as outras constantes foram simulados, até que a melhor combinação fosse obtida.

Pode-se obter um paralelo sobre a maneira como o *gossip* e o *EvAnt* disseminam as mensagens:

***gossip*:** cada nodo dissemina $\log_{10}(N)$ mensagens, todas na mesma rodada;

***EvAnt*:** cada nodo dissemina, em média, $\log_{10}(N)$ mensagens, uma por rodada.

A vizinhança média de uma rede é obtida através do quociente entre o total de enlaces da rede e o número de nodos (N), conforme mostrado pela expressão 6.3.

$$\text{vizinhança média} = \frac{\sum_{n=1}^{|N|} |e_n|}{|N|} \tag{6.3}$$

Onde $|e_n|$ representa o tamanho da vizinhança do n-ésimo nodo.

A tabela 6.1 apresenta os valores resultantes das simulações realizadas com a variação dos parâmetros do algoritmo *EvAnt*. Os valores mostrados são as médias obtidas de 10 execuções. Para mostrar como a variação de parâmetros influencia nos valores das métricas, foram criadas 10 redes *powerlaw* e outras 10 com topologia aleatória, todas com 200 nodos. Para cada rede *powerlaw*, há uma rede aleatória com o mesmo número de nodos e enlaces. Em cada rede, dois eventos foram simulados, ou seja, 1% do tamanho total da rede. Cada evento é disparado em um intervalo de 5 rodadas. Na primeira série de resultados, rotulada como "valores padrão", foram utilizados os valores mostrados acima. Nos demais resultados, também foram utilizados os valores mostrados anteriormente, com exceção do parâmetro mostrado na primeira coluna da tabela, ou seja, aquele que foi alterado.

Pela análise da tabela 6.1, percebe-se que, ao aumentar ou diminuir o valor de α, em relação aos valores padrão, também aumenta-se a latência, a quantidade média de mensagens disseminadas e de mensagens criadas, no caso das redes *powerlaw*. Nas redes de topologia aleatória, valores menores de α implicam em menor quantidade de mensagens trocadas e criadas. Ao aumentar o valor de α nas redes de topologia aleatória, nota-se um aumento no número de mensagens trocadas e uma pequena diminuição no número de mensagens criadas. Nas redes de topologia aleatória, a latência também aumentou ao variar o α.

Com a diminuição do valor de δ, não há ganho em relação à latência nas redes *powerlaw*. Nas redes de topologia aleatória, a diminuição do valor de δ acarretou em mais rodadas para a detecção completa dos eventos. Em ambas as topologias, observa-se um

	métrica	*powerlaw*	topologia aleatória
valores padrão	latência	46.3	41.9
	mensagens trocadas	1957.3	1591.2
	mensagens criadas	81.5	67.9
	awareness rate	200.2	200.6
$\alpha = 4$	latência	49.4	48.2
	mensagens trocadas	2147.2	1972.6
	mensagens criadas	83	69.4
	awareness rate	200.2	200.6
$\alpha = 16$	latência	47	47.5
	mensagens trocadas	2062.8	1933
	mensagens criadas	82.4	66
	awareness rate	200.2	200.6
$\delta = 0.1$	latência	46.3	46.2
	mensagens trocadas	2153	2004.7
	mensagens criadas	102.4	78.8
	awareness rate	200.2	200.6
$\delta = 0.9$	latência	47	45.7
	mensagens trocadas	1817.6	1781.8
	mensagens criadas	55.7	60.1
	awareness rate	200.2	200.6
$\gamma_{min} = \frac{\left(\frac{\frac{\log(N)}{4}}{\lvert\text{vizinhança média}\rvert^{\delta}}\right)}{2}$	latência	57.4	52.2
	mensagens trocadas	1660.4	1555.3
	mensagens criadas	56.6	52.3
	awareness rate	200.2	200.6
$\gamma_{min} = \left(\frac{\frac{\log(N)}{4}}{\lvert\text{vizinhança média}\rvert^{\delta}}\right) \times 2$	latência	32.6	30.7
	mensagens trocadas	2156.4	2231
	mensagens criadas	120.5	122.9
	awareness rate	200.2	200.6
$\rho = 0.1$	latência	85.7	101
	mensagens trocadas	1243.4	1278.3
	mensagens criadas	17.9	18.1
	awareness rate	200.2	200.6
$\rho = 0.9$	latência	27.8	27.5
	mensagens trocadas	3006	2798.1
	mensagens criadas	274.4	211.7
	awareness rate	200.2	200.6

Tabela 6.1: Resultados das simulações com a variação de parâmetros do algoritmo *EvAnt*.

aumento considerável no número de mensagens trocadas.

Com o aumento do valor de γ_{min}, os nodos criam mensagens durante um período maior de rodadas, o que explica o aumento na quantidade de mensagens trocadas e de mensagens criadas, além da diminuição no tempo necessário para o algoritmo tornar toda a rede ciente dos eventos. Por fim, ao aumentar a taxa de evaporação dos feromônios,

controlada pela constante ρ, observa-se uma quantidade maior de mensagens criadas e de mensagens trocadas, o que reflete em uma latência menor.

A taxa de ciência (*awareness rate*) é sempre a mesma, pois, em todas as execuções, o algoritmo conseguiu tornar todos os nodos cientes das novidades. O valor obtido é maior que 200 pois houve a criação de novos nodos durante a simulação, o que torna o tamanho final da rede maior que o inicial.

6.2 Exemplo

Para facilitar o entendimento de cada algoritmo na detecção de eventos e disseminação de informações, será apresentado a seguir um exemplo de rede e um dado evento nessa rede. Então, será mostrado o comportamento de cada algoritmo na detecção desse evento e como a novidade será disseminada aos demais membros da rede.

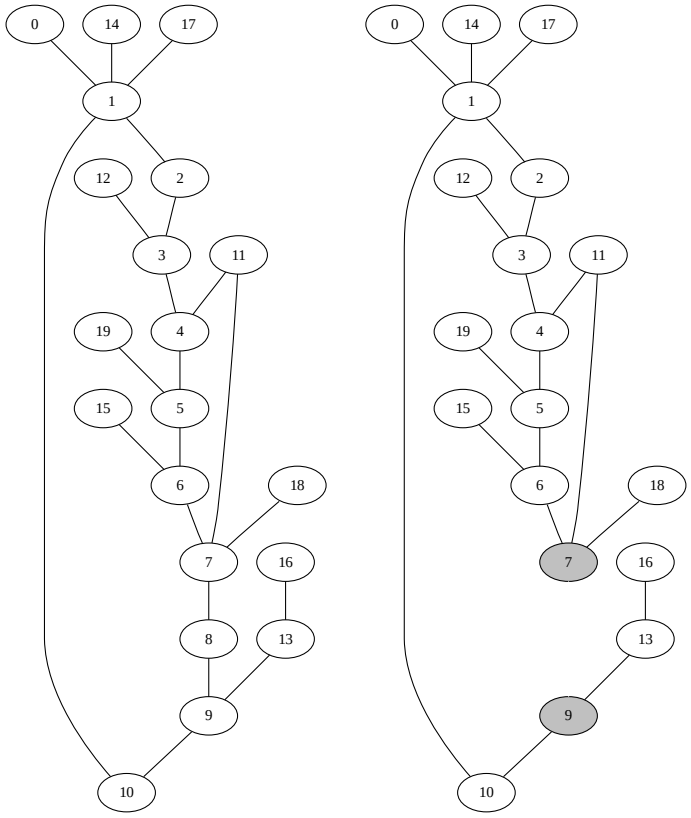
A rede que ilustrará o exemplo é mostrada na figura 6.1(a). Trata-se de uma rede *powerlaw* de 20 nodos, cujo conjunto de nodos "populares" (altamente conectados) é composto pelos nodos numerados de 0 a 9.

O evento utilizado para o exemplo será a falha do nodo 8. Assim, após a ocorrência do evento, a rede terá a topologia mostrada pela figura 6.1(b). Os nodos que detectam o evento são mostrados em cor cinza.

6.2.1 *Flooding*

No *flooding*, os nodos que tornam-se cientes da novidade na primeira e segunda rodadas são mostrados na figura 6.2.

O algoritmo inicia-se do ponto onde o evento foi detectado, ou seja, os nodos 7 e 9. Na primeira rodada, esses dois nodos disseminam as suas informações para todos os seus vizinhos. Assim, o nodo 7 envia mensagens para os nodos 6, 11 e 18. As mensagens enviadas pelo nodo 9 terão como destino os nodos 10 e 13. Na segunda rodada, os nodos que receberam as mensagens as repassam aos seus respectivos vizinhos, de acordo com o esquema abaixo:



(a) Rede antes do evento. (b) Rede após o evento.

Figura 6.1: Rede *powerlaw* de 20 nodos antes e após o evento.

Nodo 6: nodos 5, 7 e 15.

Nodo 11: nodos 4 e 7.

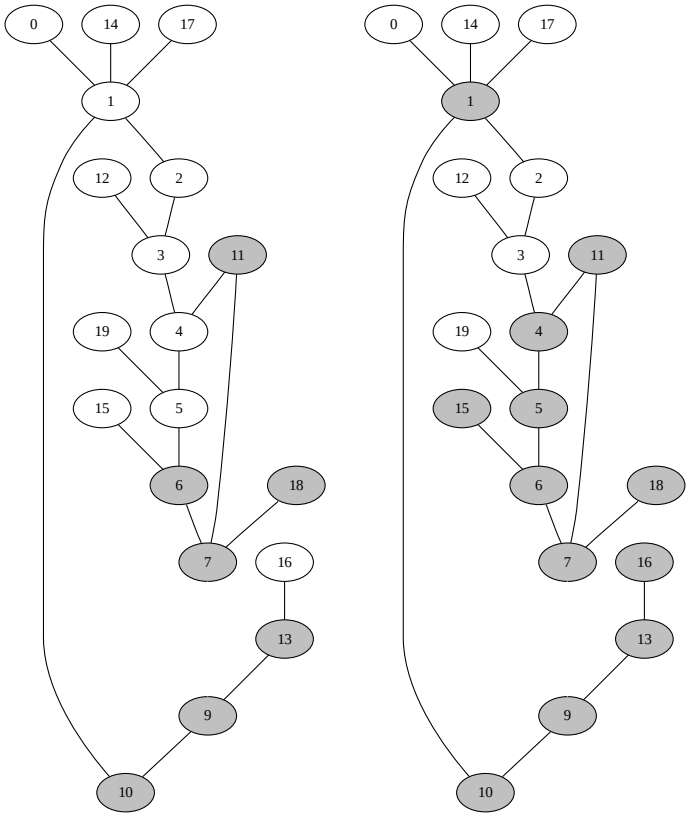
Nodo 18: nodo 7.

Nodo 10: nodos 1 e 9.

Nodo 13: nodos 9 e 16.

Os nodos que receberam mensagens na segunda rodada repassam as informações a todos os seus vizinhos e assim por diante, até que haja alguma rodada na qual todos os nodos já saibam do evento e, assim, não disseminem mais novas mensagens.

Durante as duas primeiras rodadas, a taxa de ciência obtida pelo *flooding* foi de $\frac{12}{19} = 0.63$ ou 63%. Foram disseminadas 15 mensagens, 5 durante a primeira rodada e 10 ao longo da segunda. Houve a criação de 15 mensagens, 5 na primeira rodada e 10 durante a segunda.



(a) Rodada 1. (b) Rodada 2.

Figura 6.2: *Flooding*: nodos cientes da novidade após rodadas 1 e 2.

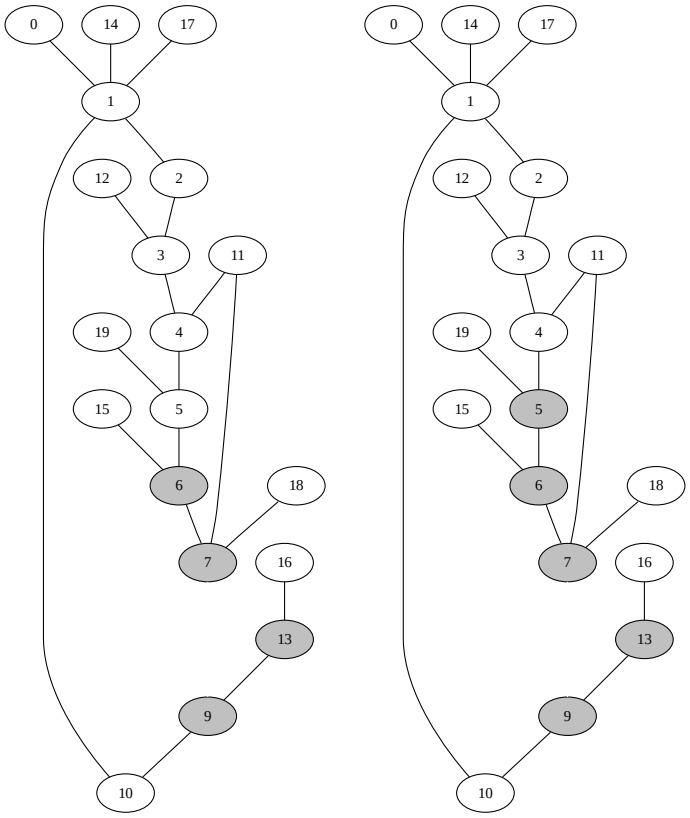
6.2.2 *Gossip*

O comportamento do *gossip* na rede utilizada como exemplo é mostrado na figura 6.3. De maneira semelhante ao *flooding*, o ponto de partida do *gossip* utilizado para as simulações é o local da ocorrência do evento. Os parâmetros utilizados são os mesmos descritos na seção 6.1, ou seja, $f = \log_{10}(N)$ e $t = 1$. Logo, sendo N o tamanho total da rede, $f = \log_{10}(20) = 1$. Durante a primeira rodada, o nodo 7 envia mensagem ao nodo 6. O nodo 9 escolheu o nodo 13 como destino da mensagem. Na segunda rodada, os nodos que receberam mensagens sorteiam os próximos destinos das mensagens. No exemplo, mostrado pela figura 6.3(b), o nodo 6 sorteou como destino o nodo de número 5. O nodo 13, neste caso, fez a mensagem voltar ao destinatário anterior, sorteando o nodo 9 como destinatário da mensagem.

A continuação do algoritmo seria o nodo 5 sortear o próximo destino das informações. Como o nodo 9 já sabia anteriormente da novidade trazida pela mensagem vinda do nodo

13, ela não seria mais repassada pelo nodo 9.

A taxa de ciência obtida pelo *gossip* durante as duas primeiras rodadas foi igual a $\frac{5}{19} = 0.26$ ou 26%. Foram disseminadas 4 mensagens, 2 durante a primeira rodada e mais 2 ao longo da segunda. Houve a criação de 4 mensagens, 2 na primeira rodada e 2 durante a segunda.



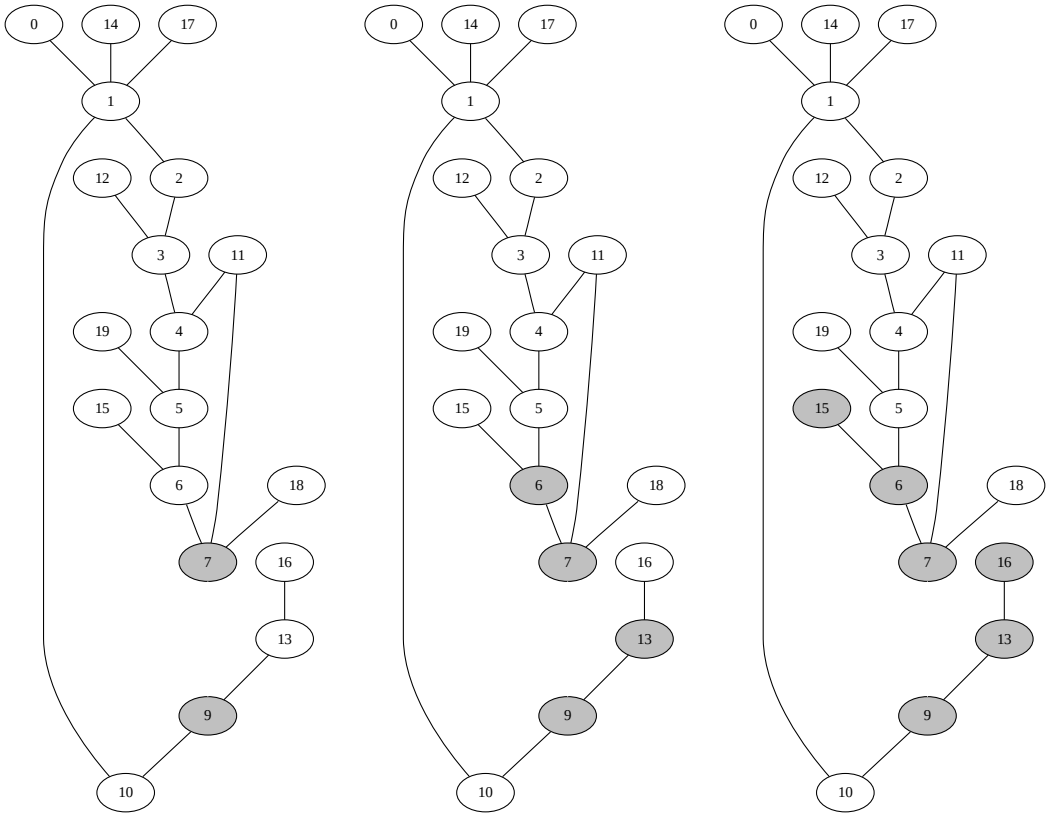
(a) Rodada 1. (b) Rodada 2.

Figura 6.3: *Gossip*: nodos cientes da novidade após rodadas 1 e 2.

6.2.3 *EvAnt*

Os nodos cientes ao longo das rodadas 1, 2 e 3 para o algoritmo *EvAnt* são mostrados na figura 6.4. Os valores dos parâmetros são os mesmos do que os descritos em 6.1. Na primeira rodada, os nodos que detectaram o evento, 7 e 9, iniciam a disseminação das formigas.

Neste caso, como nenhuma mensagem foi enviada anteriormente pelo nodo 7, todos os enlaces possuem a mesma probabilidade de serem escolhidos. Essa probabilidade é



(a) Rodada 1. (b) Rodada 2. (c) Rodada 3.

Figura 6.4: *EvAnt*: nodos cientes da novidade após rodadas 1 e 2.

calculada utilizando-se a expressão 5.2. Para o exemplo, o valor é mostrado pela expressão 6.4. Nesse caso, o nodo escolhido como destino foi o de número 6, conforme mostrado pela figura 6.4(b)

$$
\begin{aligned}
P_{7,j}(t) &= \frac{\left(0+1\right)^{-8}}{\left((C_{7,6}(t)+1)^{-8}+(C_{7,11}(t)+1)^{-8}+(C_{7,18}(t)+1)^{-8}\right)} = \\
&= \frac{(1)^{-8}}{\left((0+1)^{-8}+(0+1)^{-8}+(0+1)^{-8}\right)} = 0.33
\end{aligned}
\tag{6.4}
$$

O nodo 9 seguirá o mesmo método. Porém, como ele possui apenas 2 vizinhos, a probabilidade de cada um deles ser escolhido será de 0.5 ou 50%. Conforme mostrado pela figura 6.4(b), o destino escolhido foi o nodo 13.

Com o início de uma nova rodada, as duas formigas criadas anteriormente chegam nos seus nodos de destino. A primeira formiga, vinda do nodo 7, chega ao nodo 6. A outra formiga, criada pelo nodo 9, alcança o nodo 13. Consequentemente, as concentrações de

feromônio dos enlaces (7,6) e (9,13) são incrementadas em uma unidade, de acordo com a expressão 5.1.

Então, ocorre a evaporação do valor de feromônios de todos os nodos. As duas formigas que se deslocaram até então alcançaram os seus destinos nesta rodada. Logo, ainda não ocorre evaporação de feromônios, uma vez que o t e o t_0 da expressão 5.3 possuem o mesmo valor, como mostrado na expressão 6.5.

$$\begin{aligned} C_{7,6}(t) &= 1 \cdot (1 - 0.4)^{(0.2-0.2)} = 1 \\ C_{9,13}(t) &= 1 \cdot (1 - 0.4)^{(0.2-0.2)} = 1 \end{aligned} \tag{6.5}$$

Os limites mínimo e máximo da concentração de feromônios variam para cada nodo, de acordo com o tamanho da vizinhança, conforme as expressões 5.5 e 5.6. Para o nodo 7, os limites mínimo e máximo da concentração de feromônios são mostrados pelas expressões 6.6 e 6.7

$$L_{min}(t) = \frac{\frac{\log(N)}{4}}{|\text{vizinhança média}|^{0.5}} \cdot 3^{0.5} \tag{6.6}$$

$$L_{max}(t) = \left(100 \cdot \frac{\frac{\log(N)}{4}}{|\text{vizinhança média}|^{0.5}} \right) \cdot 3^{0.5} \tag{6.7}$$

A vizinhança média, definida pela expressão 6.3, considera N e $|e_n|$ sendo a quantidade *inicial* de nodos e enlaces da rede (antes de os eventos ocorrerem). Na rede de exemplo, o cálculo da vizinhança média é mostrado através da expressão 6.8.

$$\text{vizinhança média} = \frac{\sum_{n=1}^{|N|} |e_n|}{|N|} = \frac{42}{20} = 2.1 \tag{6.8}$$

Assim, os valores mínimo e máximo da concentração de feromônios do nodo 7 são mostrados pela expressões 6.9 e 6.10.

$$L_{min_7}(t) = \frac{1.30/4}{2.1^{0.5}} \cdot 3^{0.5} = 0.38 \tag{6.9}$$

$$L_{max_7}(t) = \left(100 \cdot \frac{1.30/4}{2.1^{0.5}}\right) \cdot 3^{0.5} = 38.87 \tag{6.10}$$

Para o nodo 9, os limites mínimo e máximo da concentração de feromônios são mostrados pelas expressões 6.11 e 6.12.

$$L_{min_9}(t) = \frac{1.30/4}{2.1^{0.5}} \cdot 2^{0.5} = 0.31 \tag{6.11}$$

$$L_{max_9}(t) = \left(100 \cdot \frac{1.30/4}{2.1^{0.5}}\right) \cdot 2^{0.5} = 31.71 \tag{6.12}$$

Como a concentração de feromônios dos nodos 7 e 9 ficaram abaixo dos seus respectivos limites mínimos, os dois nodos irão criar mais uma formiga cada.

Na segunda rodada, as mensagens criadas na rodada anterior localizam-se agora nos nodos 6 e 13. Para a mensagem no nodo 6, há maior probabilidade de ela ter os nodos 5 ou 15 como próximo destinatário, uma vez que o enlace $(6, 7)$ já foi utilizado na rodada anterior. Com o nodo 13 ocorre algo semelhante: o nodo 16 possui maior probabilidade de ser o próximo destinatário, pois o enlace $(9, 13)$ já foi utilizado uma vez. As probabilidades de migração para cada vizinho dos nodos 6 e 13 são mostradas nas expressões 6.13 e 6.14, respectivamente.

$$\begin{aligned}
P_{6,7}(t) &= \frac{\left(1+1\right)^{-8}}{\left((C_{6,7}(t)+1)^{-8} + (C_{6,5}(t)+1)^{-8} + (C_{6,15}(t)+1)^{-8}\right)} = \\
&= \frac{(2)^{-8}}{\left((1+1)^{-8} + (0+1)^{-8} + (0+1)^{-8}\right)} = 0.001949318 \\
P_{6,5}(t) &= \frac{\left(0+1\right)^{-8}}{\left((C_{6,7}(t)+1)^{-8} + (C_{6,5}(t)+1)^{-8} + (C_{6,15}(t)+1)^{-8}\right)} = \\
&= \frac{(1)^{-8}}{\left((1+1)^{-8} + (0+1)^{-8} + (0+1)^{-8}\right)} = 0.499025341 \\
P_{6,15}(t) &= \frac{\left(0+1\right)^{-8}}{\left((C_{6,7}(t)+1)^{-8} + (C_{6,5}(t)+1)^{-8} + (C_{6,15}(t)+1)^{-8}\right)} = \\
&= \frac{(1)^{-8}}{\left((1+1)^{-8} + (0+1)^{-8} + (0+1)^{-8}\right)} = 0.499025341
\end{aligned} \tag{6.13}$$

$$
\begin{aligned}
P_{13,9}(t) &= \frac{\left(1+1\right)^{-8}}{\left((C_{13,9}(t)+1)^{-8}+(C_{13,16}(t)+1)^{-8}\right)} = \\
&= \frac{\left(2\right)^{-8}}{\left((1+1)^{-8}+(0+1)^{-8}\right)} = 0.003891051 \\
P_{13,16}(t) &= \frac{\left(0+1\right)^{-8}}{\left((C_{13,9}(t)+1)^{-8}+(C_{13,16}(t)+1)^{-8}\right)} = \\
&= \frac{\left(1\right)^{-8}}{\left((1+1)^{-8}+(0+1)^{-8}\right)} = 0.996108949
\end{aligned}
\tag{6.14}
$$

As formigas dos nodos 6 e 13 escolhem como destino os nodos 15 e 16, respectivamente. No início da rodada 3, as formigas chegarão aos destinos selecionados, conforme mostrado na figura 6.4(c)

Ao final das três rodadas, a taxa de ciência obtida foi de $\frac{6}{19} = 31.5\%$. Foram trocadas 4 mensagens e houve a criação de 4 formigas.

6.3 Resultados obtidos

A figura 6.5 apresenta os gráficos das redes *powerlaw* de 1000 nodos. Inicialmente, a figura 6.5(a) apresenta a latência obtida por cada estratégia. Nas figuras 6.5(b) e 6.5(c) podemos observar o número de mensagens trocadas e a quantidade de formigas criadas ao longo do tempo, respectivamente. De acordo com a figura 6.5(a), o *gossip* não foi capaz de tornar todas as redes cientes do conjunto de acontecimentos. Os outros dois algoritmos disseminaram todos os eventos, com o *flooding* sendo cerca de 64.5% (28 rodadas) mais rápido que o *EvAnt*. Por outro lado, o número de mensagens trocadas ao longo da simulação foi bem maior no *flooding*, cerca de 46.3% (21755 mensagens) a mais do que o algoritmo *EvAnt* gastou para tornar toda a rede ciente dos eventos. O número de mensagens criadas foi cerca de 99% menor no *EvAnt*, uma vez que ele não necessita criar cópias das mensagens que dissemina aos vizinhos. A figura 6.5(d) mostra a taxa de ciência, ou seja, apresenta em mais detalhes os resultados mostrados pela figura 6.5(a).

Na figura 6.6 temos os gráficos referentes às redes de topologia aleatória de 1000 nodos.

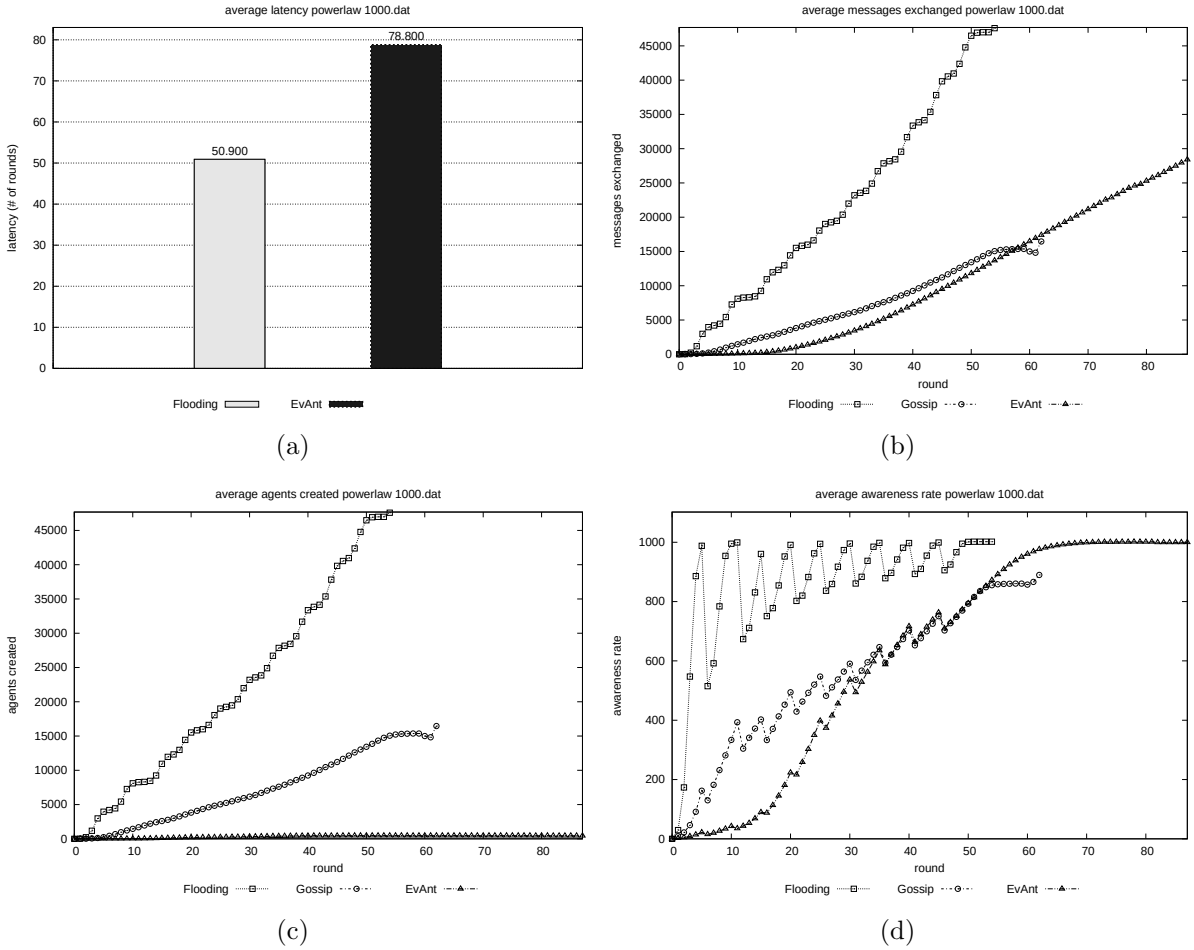


Figura 6.5: Gráficos referentes às redes *powerlaw* de 1000 nodos.

O *gossip* não conseguiu espalhar as novidades para todas as redes, porém ficou próximo, restando, em média, apenas 0.4% dos nodos sem receber todas as novidades. A latência do *flooding* é menor em aproximadamente 34.6% (28 rodadas), se comparado com o algoritmo *EvAnt*, como pode ser observado através da figura 6.6(a). De acordo com a figura 6.6(b), o algoritmo *EvAnt* trocou cerca de 37400 mensagens a menos que *flooding*, uma diferença de 62%. Embora o *gossip* tenha uma taxa de ciência muito próxima daquela obtida pelos outros dois algoritmos, a quantidade de mensagens trocadas por este algoritmo foi em média 39.7% superior àquela obtida pelo *EvAnt*, o equivalente a um excedente de aproximadamente 15000 mensagens trocadas. O número de mensagens criadas ao longo do processo, mostrado na figura 6.6(c), pelo motivo já colocado anteriormente, foi muito menor no algoritmo *EvAnt*, cerca de 99.3% em relação ao *flooding* e 98.9% em relação ao *gossip*.

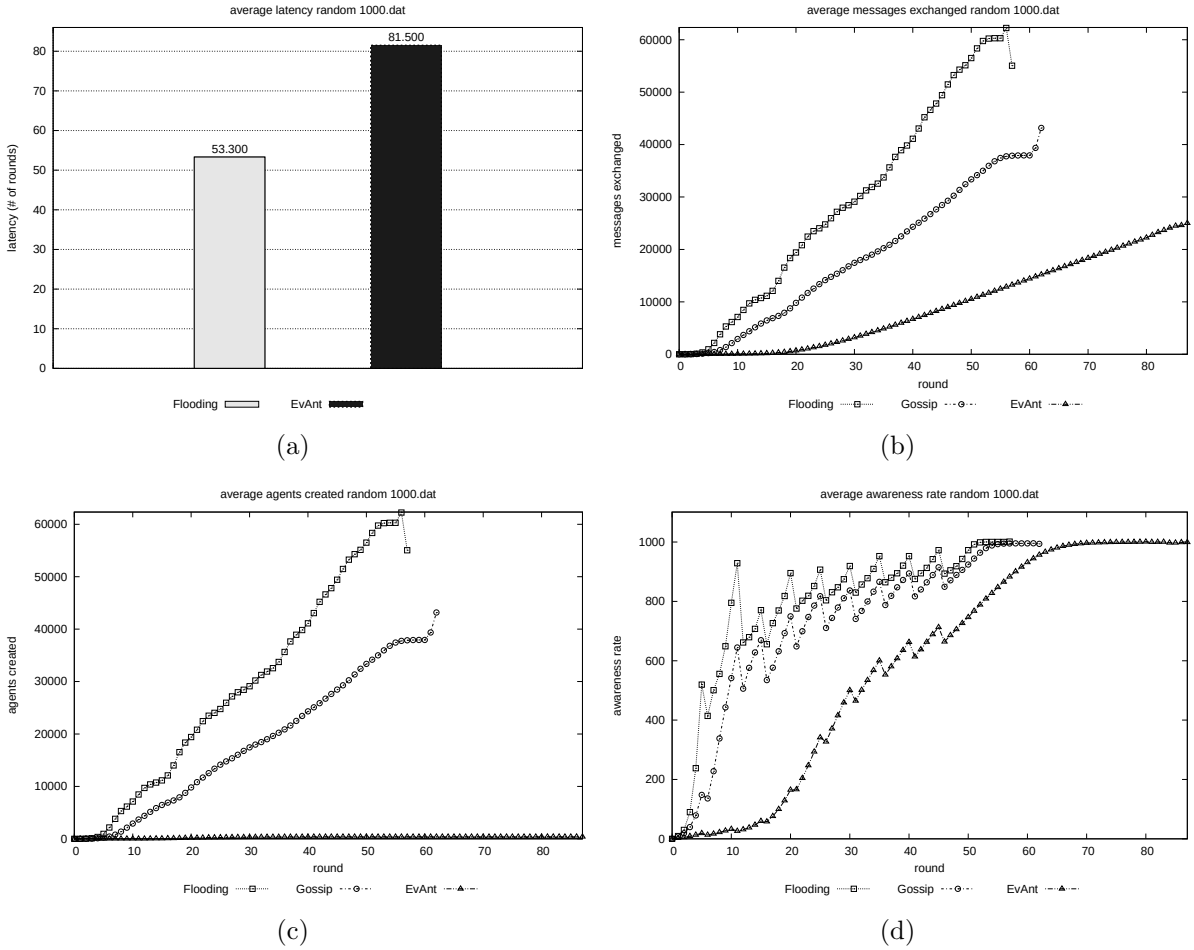


Figura 6.6: Gráficos referentes às redes de topologia aleatória de 1000 nodos.

Na figura 6.7 são apresentados os resultados das simulações com redes *powerlaw* de 2000 nodos. Novamente, o *gossip* foi o único algoritmo que não conseguiu tornar todas as redes cientes dos eventos. O *flooding* foi, em média, 23% (31 rodadas) mais rápido que o *EvAnt*, conforme a figura 6.7(a) apresenta. O número de mensagens trocadas ao longo da simulação, mostrado na figura 6.7(b), continua sendo menor no algoritmo *EvAnt*. Em média, o *flooding* gasta aproximadamente 30000 mensagens a mais do que o *EvAnt*. uma diferença em torno de 22.1%. O número de mensagens criadas pelo *EvAnt*, como já foi explicado, continua bem inferior aos valores obtidos pelos outros dois algoritmos: aproximadamente 99.2% menos mensagens criadas em relação ao *flooding* e 97.6% a menos se comparado ao *gossip*, conforme pode ser observado na figura 6.7(c).

A figura 6.8 contém os resultados para redes de topologia aleatória de 2000 nodos. O *flooding* consegue, em média, ter uma latência 23.3% (32 rodadas) menor que o *EvAnt*,

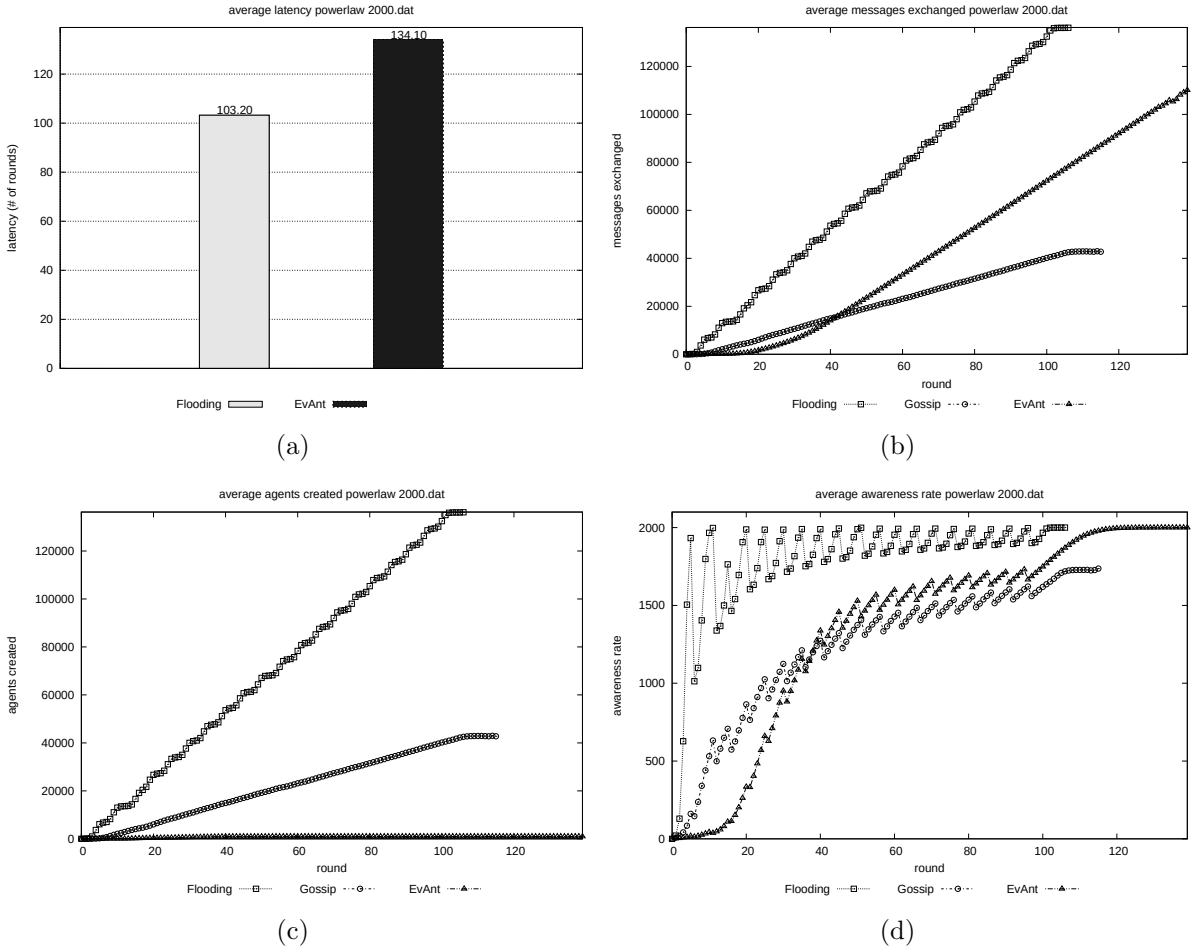


Figura 6.7: Gráficos referentes às redes *powerlaw* de 2000 nodos.

conforme apresentado na figura 6.8(a). Por outro lado, o algoritmo *EvAnt* troca aproximadamente 30% (ou 40571) mensagens a menos, o que pode ser observado na figura 6.8(b). Segundo a figura 6.8(c), o número de mensagens criadas pelo *flooding* continua muito superior, algo em torno de 99.3% a mais se comparado ao *EvAnt*. O *gossip* não foi capaz de tornar todas as redes cientes dos acontecimentos, porém ficou muito próximo, faltando comunicar as novidades para, em média, 0.5% da rede. Porém, para atingir tal resultado, aproximadamente 1,1% mais mensagens foram trocadas em comparação com o *EvAnt*. Em relação à quantidade de mensagens criadas, o *EvAnt* continua com os resultados muito melhores, uma vez que cria, em média, 99.1% mensagens a menos que o *gossip*.

Por fim, temos os resultados referentes às redes de 2500 nodos. De acordo com a figura 6.9(a), o *gossip* não foi capaz de deixar todos os nodos cientes das novidades. Os

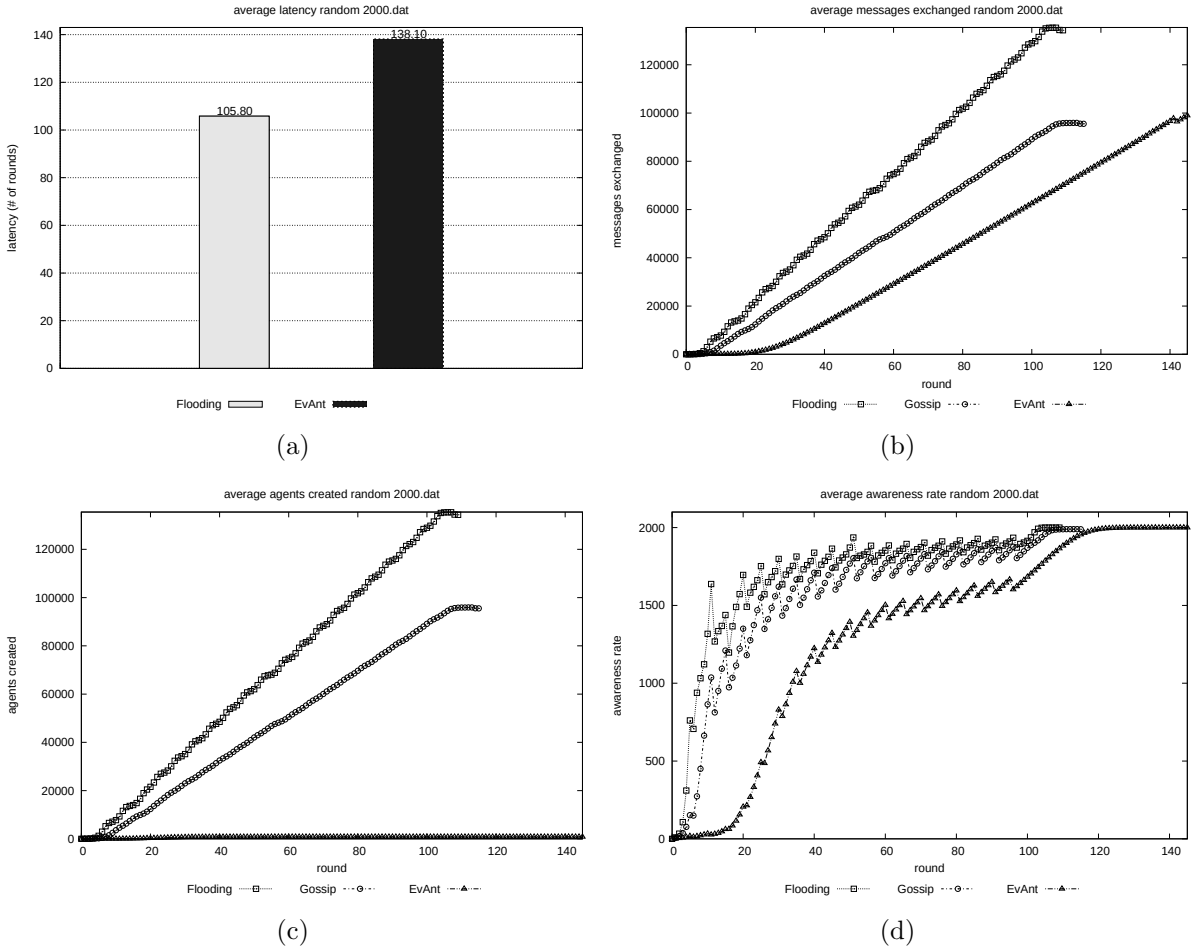


Figura 6.8: Gráficos referentes às redes de topologia aleatória de 2000 nodos.

outros dois algoritmos conseguiram comunicar todos os eventos, com o *flooding* sendo, em média, 18.2% (29 rodadas) mais rápido que o *EvAnt*. Para tal, o *flooding* trocou aproximadamente 19.7% (ou 42000) mensagens a mais do que o *EvAnt*, conforme pode ser visto em 6.9(b). A diferença entre a quantidade de mensagens criadas ao longo das simulações novamente favorece o *EvAnt*. Cerca de 99.4% mensagens a mais foram criadas pelo *flooding* (figura 6.9(c)). O *gossip*, mesmo não conseguindo atingir todos os nodos da rede, criou cerca de 96000 mensagens a mais que o algoritmo *EvAnt*, uma diferença de 98.6%.

Os gráficos para as redes de topologia aleatória de 2500 nodos estão localizados na figura 6.10. A latência do algoritmo *EvAnt* é aproximadamente 18.6% (30 rodadas) maior que a apresentada pelo *flooding* (figura 6.10(a)). O *gossip*, embora não tenha alcançado os nodos das redes, novamente chegou perto, faltando, em média, apenas 0.1% da rede sem

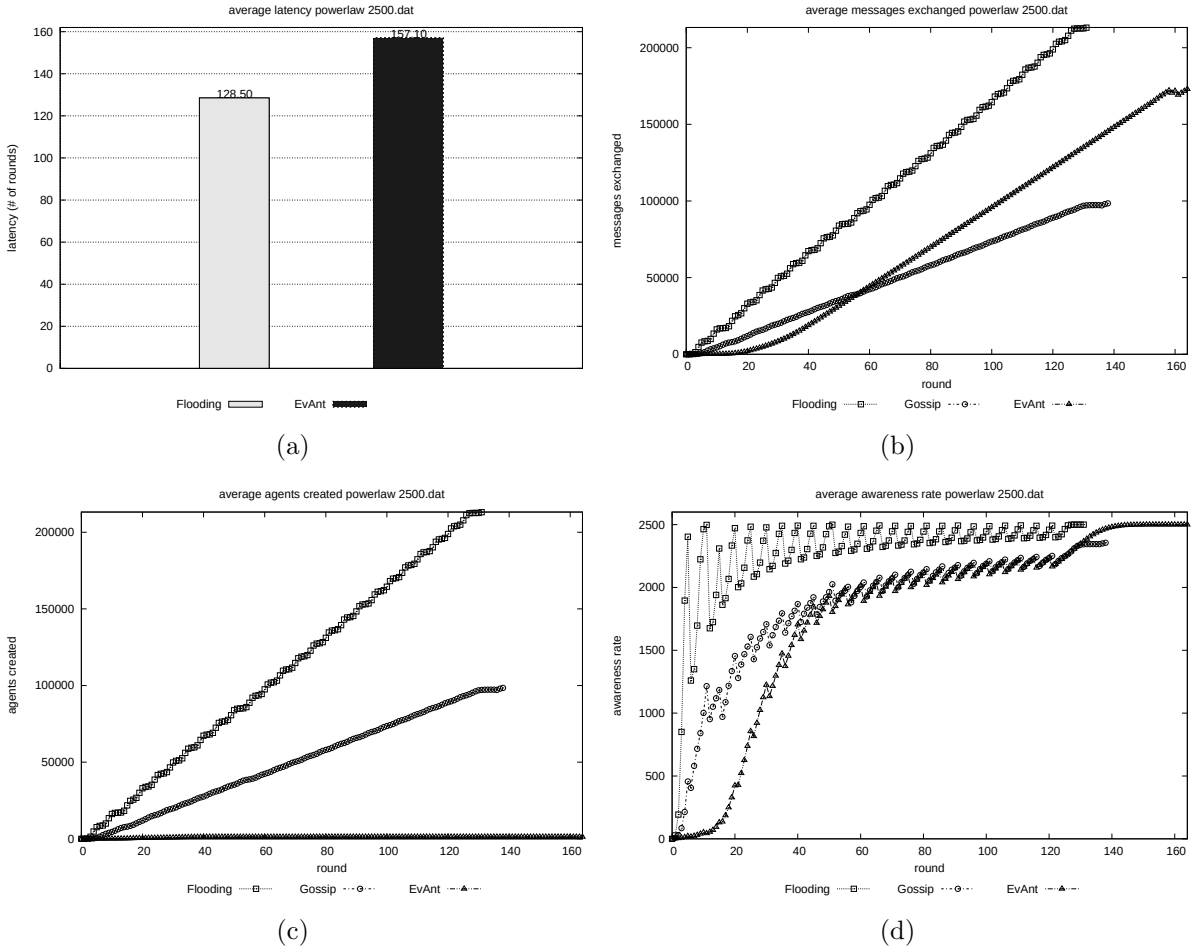


Figura 6.9: Gráficos referentes às redes *powerlaw* de 2500 nodos.

ter conhecimento de todas as novidades. Em relação ao número de mensagens trocadas, novamente o *EvAnt* comporta-se melhor, trocando em média 29.5% mensagens a menos do que o *flooding*. Isso representa uma redução de aproximadamente 62300 mensagens, conforme observado em 6.10(b). O *gossip*, mesmo sendo incapaz de disseminar todos os eventos por todas as redes, trocou cerca de 36000 mais mensagens em comparação ao *EvAnt*, uma diferença de 19.4%. A diferença na quantidade de mensagens criadas continua a favorecer o algoritmo *EvAnt*. De acordo com a figura 6.10(c), o *flooding* cria, ao longo do processo, 99.5% mensagens a mais em relação ao *EvAnt*. O *gossip* também é desfavorável nessa métrica, criando 99.4% mensagens a mais que o algoritmo *EvAnt*.

Um resumo de todos os resultados obtidos é mostrado nas tabelas 6.2, 6.3 e 6.4. Os valores mostrados na parte superior de cada campo da tabela referem-se à média dos dados obtidos nas simulações, enquanto os dados apresentados na parte inferior de cada

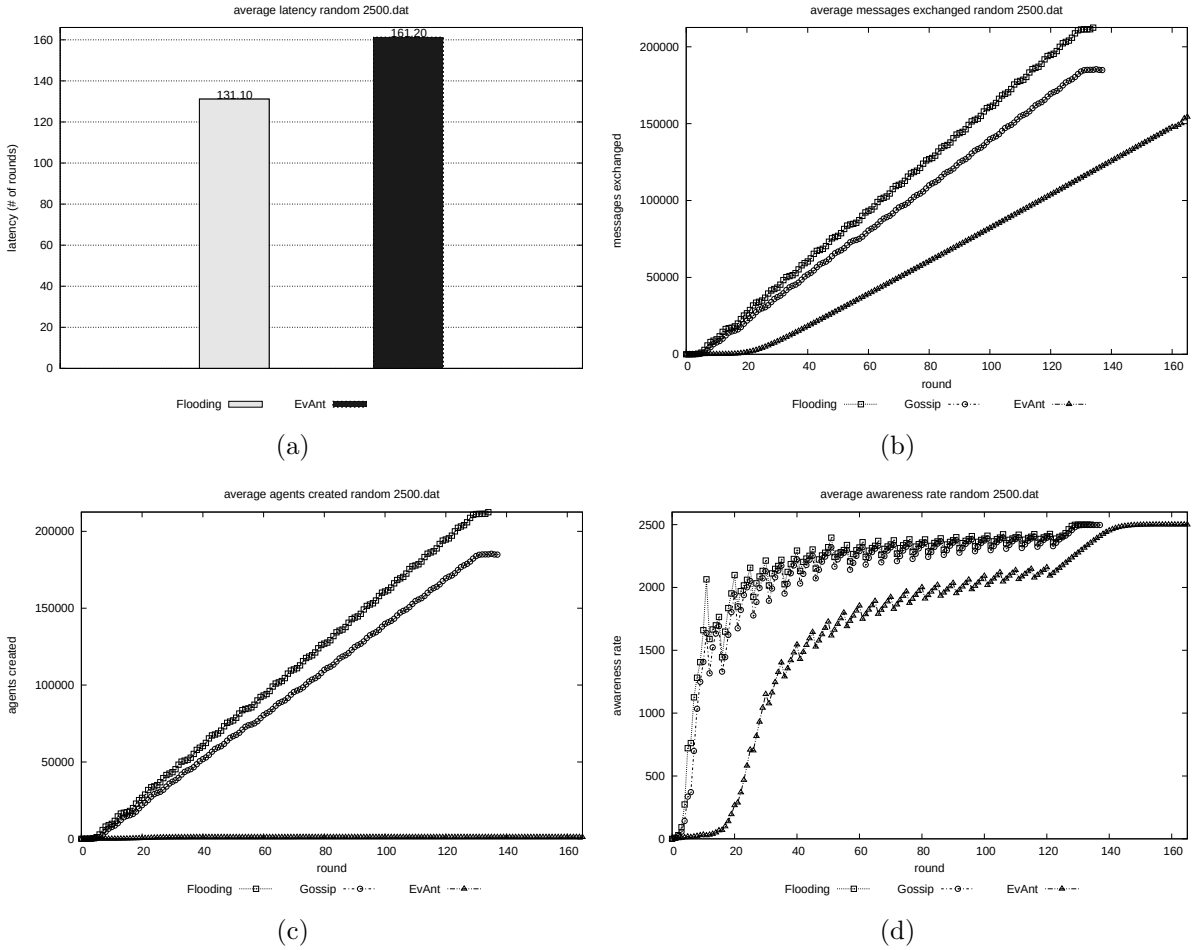


Figura 6.10: Gráficos referentes às redes de topologia aleatória de 2500 nodos.

campo dizem respeito ao desvio padrão.

Alguns valores referentes à *awareness rate* são maiores do que o tamanho inicial da rede, como por exemplo o valor 1002 na tabela 6.2, que apresenta resultados das redes de 1000 nodos. Isso ocorre pois houve a criação de novos nodos na série de eventos simulados para aquelas redes e, assim, o tamanho final da rede é maior do que o inicial.

6.4 Discussão dos resultados

Para todos os tamanhos de rede, tanto na topologia *powerlaw* quanto nas redes de topologia aleatória, pode ser observado que tanto no *flooding* quanto no *EvAnt* toda a rede consegue ficar ciente de todas as novidades. O *flooding*, como um algoritmo de disseminação confiável, sempre consegue tornar a rede ciente de todos os acontecimentos. Uma vez que a *awareness rate* do algoritmo *EvAnt* é a mesma obtida pelo *flooding*, conclui-se que o

algoritmo proposto consegue também deixar toda a rede ciente dos eventos. O *gossip* em nenhum caso conseguiu disseminar os eventos para todos os nodos. A latência obtida pelo *flooding*, como era de se esperar, é menor que a obtida pelo *EvAnt*, em todos os casos.

Ao se analisar o número de mensagens trocadas, percebe-se uma diferença considerável entre os métodos. O algoritmo *EvAnt* sempre consegue disseminar todas as informações pela rede com um menor número de mensagens que o *flooding*, essa diferença chegando a ser de mais de 60%, conforme pode ser observado nas redes de topologia aleatória de 1000 nodos. Em alguns casos, como para as redes *powerlaw* de 1000, 2000 e 2500 nodos, o *gossip* troca menos mensagens que as outras duas estratégias. Porém este algoritmo não atingiu o objetivo de disseminar as informações para todos os nodos, como explicado acima. Ainda, em outras situações, como nas redes de topologia aleatória de 1000, 2000 e 2500 nodos, o *gossip*, mesmo não sendo capaz de tornar toda a rede ciente das novidades, trocou mais mensagens do que o algoritmo *EvAnt*.

Esse é um forte indício da eficiência do algoritmo *EvAnt*, uma vez que ele é capaz de alcançar a mesma taxa de ciência do *flooding* com uma menor quantidade de mensagens disseminadas ao longo do processo.

Por fim, quanto ao número de mensagens criadas ao longo das simulações, há novamente um ganho significativo por parte do *EvAnt*. Isso ocorre pois, a cada vez que o *flooding* ou o *gossip* recebem uma mensagem que carrega novidades, eles necessitam criar *cópias* dessa mensagem para então espalhá-las pela rede. No algoritmo *EvAnt*, é criada apenas uma formiga por nodo, a cada rodada em que a condição $C_i(t) \leq L_{min}(t)$ for atendida.

6.5 Validação de resultados

A fim de verificar estatisticamente se os resultados podem ser considerados relevantes, foi utilizado o *teste de Wilcoxon* [34]. Esse teste é utilizado para verificar se um determinado conjunto de dados apresenta resultados significativamente diferentes em relação a outro.

O teste de Wilcoxon retorna um valor p (p *value*). Caso esse p *value* seja menor que 0.05 (5%), então os valores de um conjunto de dados são considerados significativamente

	powerlaw			topologia aleatória		
	flooding	*gossip*	*EvAnt*	*flooding*	*gossip*	*EvAnt*
latência	50.9	–	78.8	53.3	–	81.5
	0.56	–	3.88	0.82	–	2.99
mensagens trocadas	46981.6	15372.9	25226.4	60285.7	37938	22850.5
	5715.21	2167.52	1536.84	16869.89	6725.84	1323.09
mensagens criadas	46981.6	15373.3	468.2	60285.7	37938.1	392.9
	5715.21	2167.48	16.90	16869.89	6726.02	8.79
awareness rate	1002	860.66	1002	1000.4	995.5	1000.4
	2.16	20.85	2.16	2.83	3.91	2.83

Tabela 6.2: Resultados para as redes de 1000 nodos.

	powerlaw			topologia aleatória		
	flooding	*gossip*	*EvAnt*	*flooding*	*gossip*	*EvAnt*
latência	103.2	–	134.1	105.8	–	138.1
	0.63	–	3.03	0.78	–	4.04
mensagens trocadas	136079.1	42887.2	105979.1	135362.2	95906.3	94791.2
	1886.34	678.27	3440.67	1968.49	619.71	3973.89
mensagens criadas	136079.1	42887.2	989.9	135362.2	95906.3	849.8
	1886.34	678.27	28.77	1968.49	619.71	12.55
awareness rate	2000	1728.5	2000	2000	1989.97	2000
	0	14.42	0	0	1.43	0

Tabela 6.3: Resultados para as redes de 2000 nodos.

diferentes dos valores presentes no outro conjunto. A comparação foi realizada apenas entre os algoritmos *EvAnt* e *flooding*, uma vez que, conforme já mostrado, o *gossip* não consegue tornar toda a rede ciente dos eventos.

Nas tabelas 6.5, 6.6 e 6.7 são apresentados os *p values* obtidos ao comparar os resultados obtidos pelos algoritmos *flooding* e *EvAnt* para as redes de 1000, 2000 e 2500 nodos, respectivamente. As tabelas são divididas em 2 partes: a primeira parte apresenta os *p values* resultantes da comparação entre os métodos *flooding* e *EvAnt* em redes de topologia *powerlaw*. A outra parte mostra os *p values* relativos aos resultados das simulações em redes de topologia aleatória. Para cada uma das duas topologias, dois conjuntos de 10 valores foram comparados, um composto pelos resultados obtidos pelo *flooding* e outro pelos resultados do algoritmo *EvAnt*. Quando há "–" na tabela significa que cada conjunto é todo composto por valores iguais e os dois conjuntos de dados também são iguais entre si.

Na tabela 6.8 são apresentados os *p values* obtidos ao comparar todos os resultados

	powerlaw			topologia aleatória		
	flooding	*gossip*	*EvAnt*	*flooding*	*gossip*	*EvAnt*
latência	128.5	–	157.1	131.1	–	161.2
	0.52	–	3.17	0.73	–	1.98
mensagens trocadas	212591.8	97304.1	170525.2	211328.4	185029	148998.3
	2351.7	1942	4232.6	2377.9	1516.7	2820.33
mensagens criadas	212591.8	97304.1	1311.2	211328.4	185029	1098
	2351.7	1942	25.7	2377.9	1516.7	15.8
awareness rate	2500	2344.9	2500	2500	2497	2500
	0	10.5	0	0	0.44	0

Tabela 6.4: Resultados para as redes de 2500 nodos.

obtidos pelos algoritmos *flooding* e *EvAnt*. Para cada algoritmo e topologia, 30 valores por métrica foram utilizados, 10 referentes às redes de 1000 nodos, 10 para redes de 2000 nodos e outros 10 valores para as redes de 2500 nodos.

Pela análise das tabelas 6.5, 6.6, 6.7 e 6.8, pode-se notar uma igualdade quanto aos p *values* referentes à métrica *awareness rate*. Como já foi discutido anteriormente, ambos os algoritmos conseguem tornar toda a rede ciente dos acontecimentos, fato que explica a igualdade de valores. Em relação à latência, observa-se p *values* inferiores a 0.05, confirmando que ambos os métodos possuem conjuntos de dados que diferem nesse quesito. Quanto às outras duas métricas, número de mensagens trocadas e número de mensagens criadas, todos os p *values* também são inferiores a 0.05, o que corrobora a conclusão de que os resultados obtidos pelos algoritmos *flooding* e *EvAnt* são realmente distintos.

Através da análise conjunta dos resultados do teste de Wilcoxon com as tabelas 6.2, 6.3 e 6.4, pode-se observar um claro caso de escolha conflitante (*trade-off*) entre velocidade e carga imposta à rede. De um lado, o *flooding* oferece rápida detecção de eventos porém com um alto número de mensagens trocadas e criadas ao longo do processo. Do outro lado há o algoritmo *EvAnt*, que, embora demore algumas rodadas a mais para tornar toda a rede ciente das novidades, realiza essa tarefa com uma menor quantidade de mensagens trocadas e com uma diferença muito expressiva no número de mensagens criadas.

	powerlaw	topologia aleatória
awareness rate	1	1
mensagens trocadas	1.083e-05	1.083e-05
latência	0.0001262	0.0001442
mensagens criadas	0.0001806	0.0001817

Tabela 6.5: *p values* obtidos ao utilizar o teste de Wilcoxon para as redes de 1000 nodos.

	powerlaw	topologia aleatória
awareness rate	–	–
mensagens trocadas	1.083e-05	1.083e-05
latência	0.0001366	0.0001593
mensagens criadas	1.083e-05	1.083e-05

Tabela 6.6: *p values* obtidos ao utilizar o teste de Wilcoxon para as redes de 2000 nodos.

	powerlaw	topologia aleatória
awareness rate	–	–
mensagens trocadas	1.083e-05	1.083e-05
latência	0.0001433	0.0001503
mensagens criadas	0.0001817	0.0001806

Tabela 6.7: *p values* obtidos ao utilizar o teste de Wilcoxon para as redes de 2500 nodos.

	powerlaw	topologia aleatória
awareness rate	1	1
mensagens trocadas	0.02633	0.02073
latência	0.0002184	0.0002209
mensagens criadas	3.014e-11	3.014e-11

Tabela 6.8: *p values* obtidos ao utilizar o teste de Wilcoxon com os valores relativos às redes de 1000, 2000 e 2500 nodos.

CAPÍTULO 7

CONCLUSÃO

Este trabalho apresentou o algoritmo *EvAnt*, uma estratégia para a disseminação de informações em redes dinâmicas e descentralizadas. Através de formigas que viajam pela rede, as informações são repassadas aos nodos. Cada nodo possui um depósito de feromônios relativo a cada enlace. Quanto maior a concentração de feromônios desse depósito, maior a frequência com que o enlace foi utilizado. A escolha dos destinos das formigas é tomada com base apenas em informações locais ao nodo no qual ela se localiza. Para que as formigas sejam distribuídas de maneira adequada na rede, os enlaces menos utilizados possuem maior probabilidade de serem escolhidos. Os depósitos de feromônios também são utilizados para controlar a população de formigas, de modo que não haja excesso ou falta de formigas.

Com o objetivo de não gastar recursos desnecessários da rede, a movimentação das formigas ocorre apenas quando há novidades a serem disseminadas. A estratégia mostrou-se eficiente e eficaz na tarefa a qual se propõe, uma vez que as simulações mostraram que o algoritmo *EvAnt*, apesar de demorar algumas rodadas a mais em relação ao *flooding*, é capaz de deixar toda a rede ciente dos eventos com uma menor quantidade de mensagens do que os métodos com os quais foi comparado.

Trabalhos futuros incluem considerar alterações em determinados pontos do modelo de sistema apresentado, como por exemplo considerar uma rede com enlaces direcionados. Uma implementação do algoritmo *EvAnt* para redes *peer-to-peer* deverá ser desenvolvida.

BIBLIOGRAFIA

[1] `http://si.cs.up.ac.za/?page=bib_other_swarm`.

[2] A. Ahmed e B. Far. Performance of mobile agent based network topology discovery. *Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on*, páginas 66–69, 2007.

[3] M. Aissani, M. Fenouche, H. Sadour, e A. Mellouk. Ant-dsr: Cache maintenance based routing protocol for mobile ad-hoc networks. *Telecommunications, 2007. AICT 2007. The Third Advanced International Conference on*, páginas 35–35, May de 2007.

[4] N.T.J. Bailey. *The mathematical theory of infectious diseases and its applications*. London, 1975.

[5] C. Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353–373, 2005.

[6] B. Bollobas. *Random graphs*. Cambridge Univ. Press, 2001.

[7] E. Bonabeau, M. Dorigo, e G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, USA, 1999.

[8] Amos Brocco, Apostolos Malatras, e Béat Hirsbrunner. Proactive information caching for efficient resource discovery in a self-structured grid. *BADS '09: Proceedings of the 2009 workshop on Bio-inspired algorithms for distributed systems*, páginas 11–18, New York, NY, USA, 2009. ACM.

[9] T. Bu e D. Towsley. On distinguishing between Internet power law topology generators. *IEEE INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, volume 2, 2002.

[10] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, e D. Terry. Epidemic algorithms for replicated database maintenance. *Proce-*

edings of the sixth annual ACM Symposium on Principles of distributed computing, páginas 12. ACM, 1987.

[11] Chao Dong, Xiao rong Cheng, e Ming quan Zhang. Research of mobile agent based network topology discovery. *Innovative Computing, Information and Control, 2006. ICICIC '06. First International Conference on*, volume 1, páginas 733–736, 30 2006-Sept. 1 de 2006.

[12] M. Dorigo. Optimization, learning and natural algorithms. *Italian) Ph. D. dissertation, Politecnico di Milano, Milan, Italy, 1992*.

[13] M. Dorigo e L.M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66, 1997.

[14] M. Dorigo, V. Maniezzo, e A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(1):29–41, 1996.

[15] M. Dorigo e K. Socha. An introduction to ant colony optimization. *Handbook of Approximation Algorithms and Metaheuristics*, páginas 26–1.

[16] PT Eugster, R. Guerraoui, A.M. Kermarrec, e L. Massoulie. Epidemic information dissemination in distributed systems. *Computer*, 37(5):60–67, 2004.

[17] M. Faloutsos, P. Faloutsos, e C. Faloutsos. On power-law relationships of the internet topology. *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, páginas 251–262. ACM, 1999.

[18] L.M. Gambardella e M. Dorigo. An ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, 12(3):237–255, 2000.

[19] N.P. Gopalan, C. Mala, R. Shriram, e S. Agarwal. Multicast tree computation for group communication in mobile networks using optimization techniques. *Ad Hoc and*

Ubiquitous Computing, 2006. ISAUHC '06. International Symposium on, páginas 88–93, Dec. de 2006.

[20] R. Guerraoui e Rodrigues L. *Introduction to Reliable Distributed Programming*. 2006.

[21] I. Gupta, T.D. Chandra, e G.S. Goldszmidt. On scalable and efficient distributed failure detectors. *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, páginas 170–179. ACM New York, NY, USA, 2001.

[22] S. Johnson. *Emergence: The connected lives of ants, brains, cities, and software*. Scribner, 2001.

[23] J. Kennedy, R.C. Eberhart, e Y. Shi. *Swarm intelligence*. Springer, 2001.

[24] Kenneth N. Lodding e Paul Brewster. Multi-agent organisms for persistent computing. *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, páginas 1038–1045, Washington, DC, USA, 2004. IEEE Computer Society.

[25] M.H. MacDougall. *Simulating computer systems*. MIT press, 1987.

[26] S. Mullender. *Distributed Systems*. Addison Wesley Publishing Company, 1993.

[27] B.T. Nassu, T. Nanya, e E.P. Duarte. Topology discovery in dynamic and decentralized networks with mobile agents and swarm intelligence. *Proceedings of 7th ISDA, IEEE Computer Society, Washington*, páginas 685–690, 2007.

[28] P. Norvig. *Artificial intelligence: a modern approach*. Pearson Education, 2003.

[29] B. Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47(1):213–223, 1987.

[30] T. Stiitzle e H.H. Hoos. Max-min ant system. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[31] T. Stutzle e H. Hoos. The MAX–MIN ant system and local search for the traveling salesman problem. *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC&# 039; 97)*, 1997.

[32] Jing Wei, Wei Guo, Jian Su, e Wei Tang. Mobile agent based topology discovery in mobile ad hoc networks. *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, páginas 1–4, Sept. de 2009.

[33] T. White e B. Pagurek. Towards multi-swarm problem solving in networks. *Proceedings of Third International Conference on Multi-Agent Systems (ICMAS'98)*, páginas 333–340, 1998.

[34] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[35] Fernando J. Von Zuben e Romis R. F. Attux. Inteligência de enxame. Relatório técnico, CA/FEEC/Unicamp and DECOM/FEEC/Unicamp, 1999.