ALINE MARQUES DEL VALLE

PATRULHA MULTIAGENTE DE ARESTAS

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS – BRASIL
2008

ALINE MARQUES DEL VALLE

PATRULHA MULTIAGENTE DE ARESTAS

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de concentração:

Otimização

Orientador:

Prof. Dr. Ricardo Martins de Abreu Silva

LAVRAS
MINAS GERAIS – BRASIL
2008

Ficha Catalográfica preparada pela Divisão de Processo Técnico da Biblioteca Central da UFLA

Del Valle, Aline Marques

Patrulha Multiagente de Arestas / Aline Marques Del Valle. Lavras – Minas Gerais, 2008.

Monografia de Graduação – Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Problema da Patrulha. 2. Algoritmo Multiagente. 3. Grafo Euleriano Dinâmico. I. DEL VALLE, A. M. II. Universidade Federal de Lavras. III. Título.

ALINE MARQUES DEL VALLE

PATRULHA MULTIAGENTE DE ARESTAS

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 20 de maio de 2008

_____
Prof. Dr. Cláudio Fabiano Motta Toledo

_____
Prof. Dr. Thiago de Souza Rodrigues

_____
Prof. Dr. Ricardo Martins de Abreu Silva
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL
2008

Dedico este trabalho a meus pais Celina Aparecida Marques Del Valle e João Carlos Del Valle.

Aos meus pais pelo constante apoio e amor dados durante toda a graduação e, principalmente, na minha vida.
Aos meus irmãos, Carla e Tiago, companheiros de toda hora.
Aos amigos que caminharam lado a lado comigo e dividiram várias conquistas e alegrias.
Ao meu professor e orientador Ricardo pela orientação, disponibilidade e confiança depositada em mim.
A todos que de alguma forma me ajudaram a chegar aqui. Obrigada!

PATRULHA MULTIAGENTE DE ARESTAS

RESUMO

Este trabalho tem como objetivo desenvolver um novo algoritmo multiagente, denominado Multi_Agent_Ciclic_Edge_Walk, para patrulhar arestas de grafos Eulerianos dinâmicos que minimize a ociosidade das arestas, tornando-a mais uniforme. O algoritmo é uma extensão do algoritmo de Yanovski et. al. (2002). Objetiva-se também estabelecer métricas relativas à ociosidade para medir o desempenho do algoritmo e a partir delas fazer uma comparação com os resultados não apenas de Yanovski, mas de uma variação deste último baseado em partição, denominado Multi_Agente_Partition_Edge_Walk.
Verificou-se que o algoritmo proposto obteve resultados melhores com relação às métricas estudadas neste trabalho.

Palavras-chave: Problema da Patrulha, Algoritmo Multiagente, Grafos Eulerianos Dinâmicos.

MULTI-AGENT EDGE PATROLLING

ABSTRACT

This paper aims to develop a new multiagente algorithm called Multi_Agent_Ciclic_Edge_Walk, to patrol edges of dynamic Eulerians graphs that minimize the idleness of the edges making it more uniform. The algorithm is an extension of the Yanovski's et. al. algorithm (2002). It also aims to establish metrics on the idleness measuring the performance of the algorithm which will be compare at with the Yanovski's results and variation of then based on partition, called Multi_Agente_Partition_Edge_Walk.
It was found that the proposed algorithm obtained better results with respect to the metric studied in this work.

Key-Words: Patrolling Problem, Algorithm Multi-agent, Dynamic Eulerian Graph.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

EAW – Edge_Ant_Walk

AEI – *Average Edge Idleness* (Ociosidade Média de uma Aresta)

GI – *Graph Idleness* (Ociosidade do Grafo)

EWI – *Edge Worst Idleness* (Pior Ociosidade da Aresta)

WI – *Worst Idleness* (Pior Ociosidade do Grafo)

AWI – *Average Worst Idleness* (Média da Pior Ociosidade da Aresta)

1. INTRODUÇÃO

Patrulhar é, "o ato de andar ou viajar por áreas, em intervalos regulares, para protegê-la ou supervisioná-la" (Abate, 1996). Esta tarefa é de natureza multiagente.

Agentes de patrulha são úteis em domínios em que vigilância, inspeção ou controle distribuído são necessários. Por exemplo, em jogos onde o patrulhamento de regiões é importante para os jogadores; em uma *intranet* para ajudar administradores de redes a prevenir ou detectar falhas; na indexação de páginas *web* para detectar novas páginas ou modificações (Machado et. al., 2002).

Algumas abordagens para a solução deste problema já foram pesquisadas pelo Centro de Informática da Universidade Federal de Pernambuco - UFPE (Geber Ramalho, Almeida, Santana, Machado, Leitão) e por Chevaleyre (2004), ambos abordando a patrulha em nós. Há também a pesquisa de Yanovski et. al. (2002) que aborda a patrulha em arestas. Este trabalho aborda a patrulha em aresta por ser um tema ainda pouco estudado na literatura.

1.1. Objetivos

O objetivo do trabalho é desenvolver um novo algoritmo multiagente, Multi_Agent_Ciclic_Edge_Walk, para patrulhar arestas em grafos Eulerianos dinâmicos que minimize a ociosidade e mantenha freqüência uniforme de visita nas arestas. O algoritmo é uma extensão do algoritmo de Yanovski et. al. (2002).

Objetiva-se também estabelecer métricas relativas à ociosidade para medir o desempenho do algoritmo.

Para fins de comparação foram implementados, além do algoritmo proposto no trabalho dois outros algoritmos: o Edge_Ant_Walk de Yanovski et. al. (2002) e o Multi_Agent_Partition_Edge_Walk que foi apresentado em um trabalho anterior e cuja proposta encontra-se no Apêndice A (Silva et. al., 2007).

1.2. Estrutura do Trabalho

O presente trabalho divide-se em sete capítulos, os quais visam a abordagem de questões relacionadas ao Problema da Patrulha, ao algoritmo Edge_Ant_Walk, à proposta e às métricas utilizadas. Assim sendo, os capítulos seguintes estão estruturados da seguinte forma:

- O segundo capítulo faz uma revisão bibliográfica sobre o Problema da Patrulha.
- O terceiro capítulo faz uma revisão bibliográfica sobre o algoritmo Edge_Ant_Walk para um agente e multiagente de Yanovski et. al. (2002).
- O quarto capítulo apresenta o algoritmo proposto: funcionamento, pseudocódigo e exemplos.
- O quinto capítulo identifica a metodologia utilizada, o tipo de pesquisa e apresenta as métricas.
- O sexto capítulo apresenta os resultados alcançados através de simulações e comentários sobre eles.
- E o sétimo capítulo apresenta as conclusões do trabalho e sugestões para trabalhos futuros nesta área.

2. O PROBLEMA

A tarefa da patrulha é definida como ato de percorrer constantemente determinados ambientes com objetivo de supervisioná-los, inspecioná-los e/ou controlá-los. Tal ambiente pode ser representado por um grafo (Machado, 2002).

Um grafo G = (V, E) é um conjunto não-vazio V, cujos elementos são chamados vértices ou nós, e o conjunto E de arestas é um par não-ordenado (V_i, V_j), onde V_i e V_j são elementos de V. A cada aresta E(V_i, V_j) pode estar associado um custo C_{ij}, representando a distância entre V_i e V_j (Chevaleyre, 2004).

A Figura 2.1 apresenta um grafo obtido através do Simulador desenvolvido por Machado (2002).

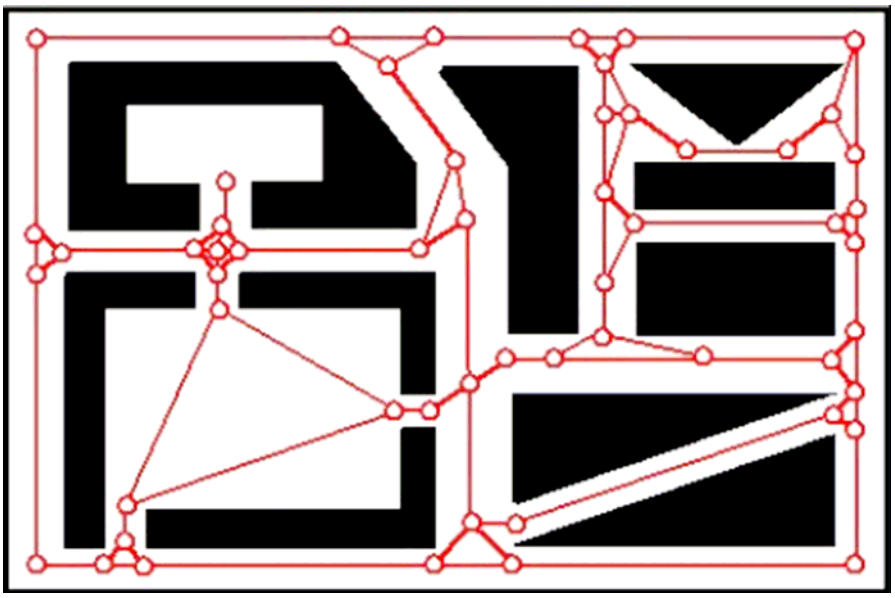

Figura 2.1: Exemplo de uma abstração de um grafo (Fonte: Machado, 2002)

Para Chevaleyre (2004) dado um grafo G, representando o ambiente, a patrulha consiste na visita contínua dos nós de G de forma a minimizar o intervalo de tempo entre duas visitas consecutivas a um mesmo nó.

Para Yanovski et. al. (2002) dado um grafo G, representando o ambiente, a patrulha consiste, além da completa exploração do mesmo, em manter a mesma freqüência de visitas a todas as arestas de G.

As definições feitas do problema por Chevaleyre (2004) e Yanovski et. al. (2002) se diferem em visitas a nós e a arestas, respectivamente. Este trabalho terá foco na patrulha de arestas.

Um exemplo de patrulha de aresta segundo Yanovski et. al. (2002) é a medida de segurança adotada em uma galeria de artes. A galeria pode ser representada por um grafo, onde os corredores são as arestas e os *halls* os nós. Os seguranças devem patrulhar a galeria garantindo que os corredores são percorridos com a mesma freqüência, garantindo que não haverá corredores mais ociosos.

3. O ALGORITMO EDGE_ANT_WALK (EAW)

O trabalho de Yanovski et. al. (2002) propõe o algoritmo Edge_Ant_Walk para resolver o problema da patrulha. De acordo com o algoritmo, os agentes em um vértice *v* sempre escolhem a aresta *e* que foi visitada há mais tempo para o seu próximo movimento. O algoritmo busca manter uma freqüência de visitas uniforme às arestas.

O algoritmo desenvolvido por Yanovski et. al. (2002) aplica-se a grafos Eulerianos direcionados. Segundo Wilson (1996), um grafo *G(E, V)* é Euleriano se for conexo e possuir um circuito que visita cada aresta de *E* exatamente uma vez.

A seguir é apresentado o algoritmo de Yanovski et. al. (2002) para o caso de um agente e para o caso multiagente.

3.1. Edge_Ant_Walk para um Agente

O algoritmo EAW de Yanovski et. al. (2002) assume que a rede de informação é um grafo e a comunicação entre os agentes é atribuída à *labels* em cada aresta que os agentes podem ler e modificar. Os *label* podem também ser chamados de *flags,* que são bandeiras colocadas no ambiente com informações relevantes, acessíveis por todos os agentes para que haja comunicação entre eles (Almeida, 2003).

Quando situado em um vértice *v*, um agente sempre escolhe uma aresta *e* e move-se por ela. Ao visitar esta aresta deixa uma marcar $\sigma(e)$ proporcional ao tempo corrente. A aresta escolhida sempre é aquela que foi percorrida há mais tempo.

O algoritmo de Yanovski et. al. (2002) pode ser formalizado pelo seguinte pseudocódigo:

```
00:  Initialization:
01:          t←0;
02:          for every edge e=u→v in graph G
03:                       set σ(e)←0
04:
05:          Find the edge e = u → v with a minimal value of σ(e)
06:          among the edges emanating from u; break ties randomly.
07:          σ(e) ← t;
08:          t ← t + 1;
09:          u ← v;
10:          Go to (05)
```

Figura 3.1: Algoritmo Edge_Ant_Walk para um agente (Fonte: Yanovski et. al., 2002)

Pode-se observar que, em um vértice v, o algoritmo nunca escolhe uma aresta já visitada se ainda há arestas que não foram visitadas, ou seja, sempre escolhe arestas com $\sigma(e) = 0$. As arestas são inicialmente visitadas em uma ordem específica e no futuro elas são revisitadas exatamente na mesma ordem (Yanovski et. al., 2002).

A Figura 3.2 mostra um exemplo do algoritmo EAW percorrendo todas as arestas de um grafo. O esboço (1) da figura mostra o grafo. Os esboços (2) a (15) mostram as fases do algoritmo. As arestas mais grossas correspondem às arestas que sofreram mais visitas e os nós sombreados ao local em que o agente se encontra.

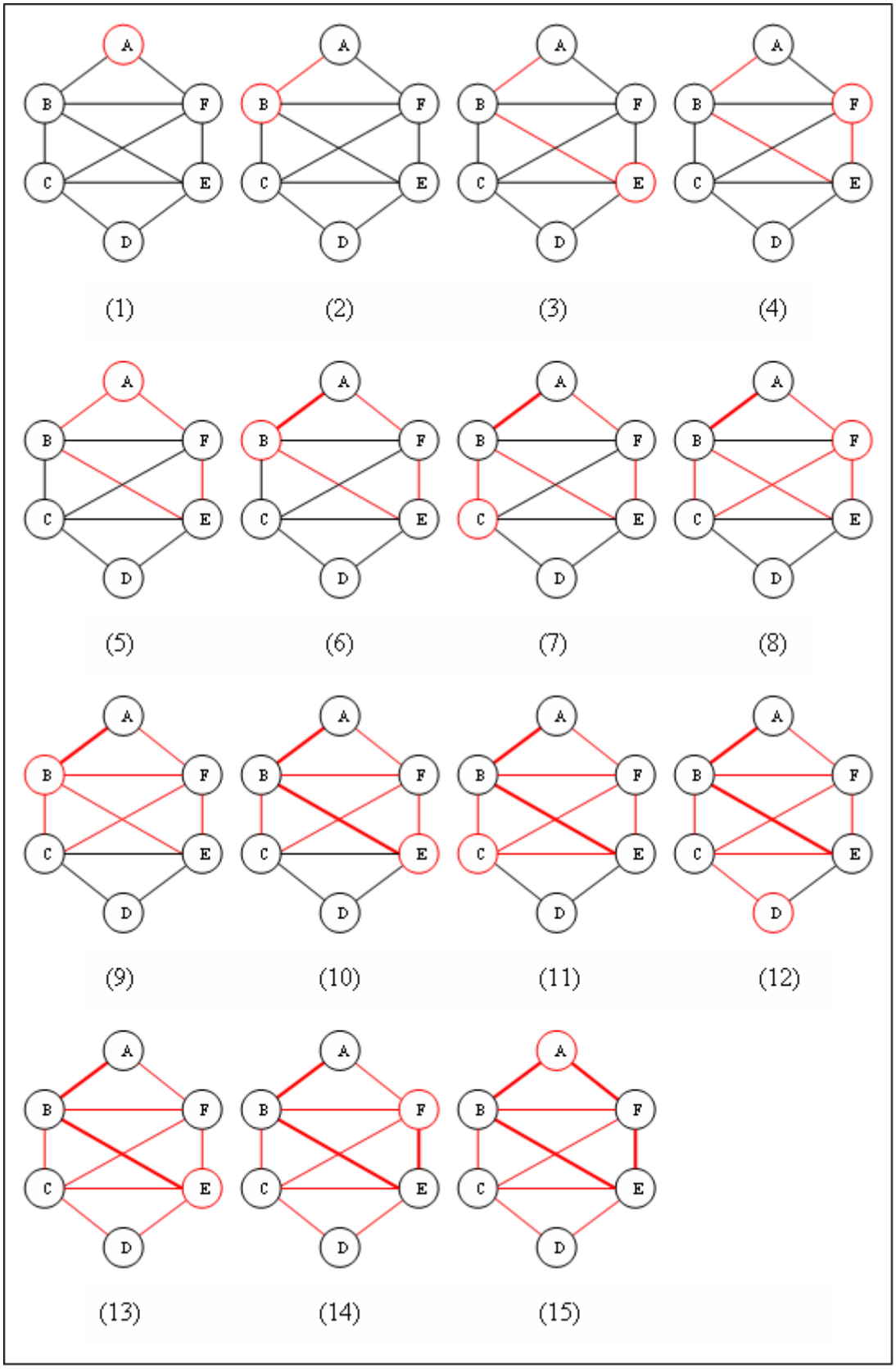

Figura 3.2: Um Exemplo do algoritmo Edge_Ant_Walk explorando um grafo (Fonte: Yanovski et. al., 2002)

De acordo com Yanovski et. al. (2002), o conjunto de arestas visitadas pelo algoritmo EAW pode ser imaginado como uma serpente, com o agente iniciando da cabeça da serpente. Em um instante t_0, quando a cabeça da serpente está em um vértice que contém arestas não visitadas, o agente vai por uma delas e a serpente estende-se. Em um instante t_1, este processo de crescimento termina, pois já não há mais arestas que não foram visitadas ($\sigma(e) = 0$). Após t_1 os agentes voltam a visitar arestas que já foram visitadas. Assumindo que o crescimento começa em t_0, pode-se chamar o intervalo de tempo $[t_0 ... t_1]$ de estágio exploratório em que todas as arestas eram novas, ou seja, com $\sigma(e) = 0$. Pode haver diversos estágios exploratórios e entre eles estágios em que partes do grafo que já havia sido explorado são visitadas novamente. Diz-se que t está no estágio exploratório $[t_0 ... t_1]$ se $t_0 \leq t \leq t_1$. Denomina-se $\bar{E}_i$ o conjunto de arestas exploradas em i-ésimos estágios exploratórios e E_t o conjunto das arestas exploradas em t.

A Figura 3.3 mostra todos os estágios exploratórios do grafo e as revisitas de arestas. No instante t_0, o agente encontra-se no nó A e começa a percorrer arestas que ainda não foram visitadas, o agente visita as arestas AB, BE, EF e FA, quando retorna ao nó A. Neste instante, t_4, não há arestas "novas" e as arestas visitadas no intervalo de tempo $[t_0 ... t_4]$ fazem parte do primeiro estágio exploratório. No nó A, o agente agora deve escolher a aresta que foi visitada há mais tempo, AB, e segui-la. Em B há ainda arestas "novas" e inicia-se um novo estágio exploratório.

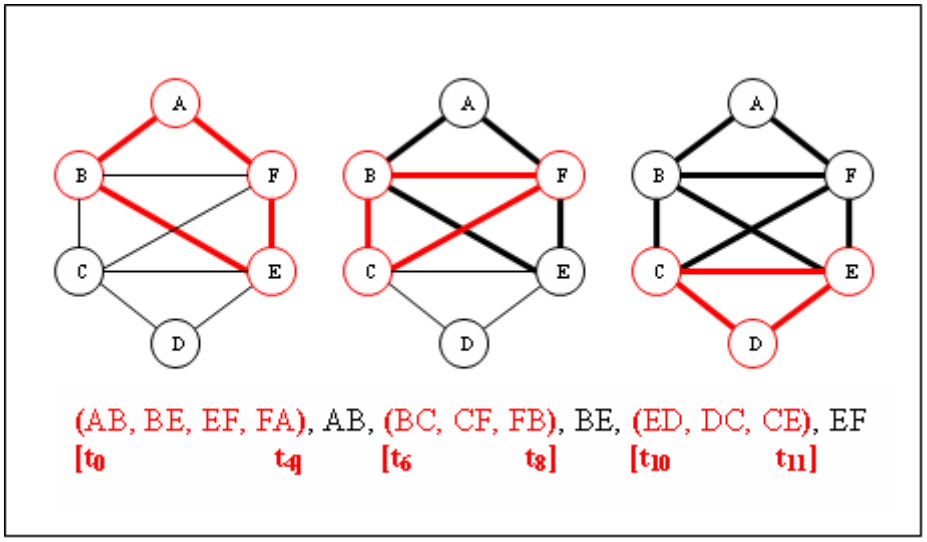

Figura 3.3: Estágios Exploratórios Intercalados por não Exploratórios

A seguir são apresentados lemas, corolários e teoremas propostos por Yanovski et. al. (2002):

Lema 1: *(Subgrafos Eulerianos) Em algum momento t, fora de um estágio exploratório, o grafo induzido por E_t é um Grafo Euleriano.*

Em um instante t, temos *m* estágios exploratórios, a união dos *m* estágios exploratórios ($\bar{E}_m$) é um subgrafo Euleriano. Quando um agente parte de um vértice *v* e está percorrendo arestas "novas" toda chegada do agente a um vértice qualquer corresponde a uma saída até que o agente retorne a *v*, disto conclui-se que os vértices de um estágio exploratório têm grau par. A união dos estágios exploratórios resulta em um subgrafo com vértices de grau par e de acordo com o Teorema de Euler um grafo é Euleriano se e somente se todos os seus vértices têm grau par (Jurkiewicz).

Corolário 1: *O conjunto $\bar{E}_m$ de arestas exploradas durante m-ésimos estágios exploratórios, ordenados de acordo com o tempo de exploração, é um circuito Euleriano em um grafo não-direcionado pelo conjunto $\bar{E}_m$.*

Em conseqüência do Lema 1, temos que o conjunto de arestas $\bar{E}_m$ forma um grafo Euleriano. Segundo Yanovski et. al. (2002) um grafo Euleriano é um grafo que possui um circuito Euleriano.

Corolário 2: *Um circuito Euleriano é um circuito limite de um grafo – uma vez completo, será repetido indefinidamente.*

Quando o agente percorre arestas que já foram visitadas, relembrando a analogia com a serpente, a serpente não se estende mais, apenas segue sua calda. Uma vez que o grafo inteiro já foi explorado, o agente seguirá a calda indefinidamente pelo circuito (Yanovski at. al., 2002).

Teorema 1: *Se G = (V, E) é um grafo Euleriano direcionado, então em tempo O(|E||V|) o agente EAW percorrerá todas as arestas do grafo e entrará em circuito Euleriano*.

O tempo gasto em todos estágios exploratório é igual a |*E*|, pois cada aresta é percorrida exatamente uma vez durante algum estágio exploratório. Após um estágio exploratório, o agente percorre arestas já exploradas. O intervalo de tempo entre dois estágios exploratórios não pode ser maior que |*E*|.

Um estágio exploratório inicia-se em um vértice *u* e termina somente após todas as arestas adjacentes a *u* forem percorridas. Assim, há no máximo, um estágio exploratório por vértice e o número total de estágios exploratórios é delimitado por |*V*|. Assim:

$$t \leq |E| + |E| * |V| = O(|E|\,|V|)$$

3.2. Edge_Ant_Walk Multiagente

Segundo Yanovski et. al. (2002) para acelerar a exploração de um grafo, aumentando a freqüência de visitas nas arestas, um caminho é ter vários agentes patrulhando ao mesmo tempo.

Em seu trabalho, Yanovski et. al. (2002), citou dois modelos de sincronização de agentes: o sincrônico e o assincrônico. No modelo sincrônico todos os agentes se atualizam simultaneamente, ou seja, há um *clock* comum para todos agentes e, no modelo assincrônico a atualização dos agentes depende de incentivos, ou seja, os agentes movem-se quando querem. Yanovski et. al. (2002) adotou o modelo sincrônico.

Os agentes então se movem seguindo o mesmo *clock*. Quando vários agentes estão em um nó e querem sair simultaneamente, eles podem escolher uma ordem aleatória de saída ou podem usar uma ordem predefinida por seus ID`s. O conjunto de arestas visitadas em cada etapa são as mesmas, independente da ordem de saída dos agentes. Assim, Yanovski et. al. (2002) assumiu, em seu trabalho, agentes idênticos, ou seja, sem a necessidade de diferenciação mediante uma identificação.

A seguir é apresentado um lema que mostra que adicionando um agente a performance do algoritmo não cai (Yanovski et. al., 2002).

Lema 3: *Adicionar um agente aumenta o número total de arestas percorridas sem diminuir o número de visitas de qualquer aresta e o número de visitas de qualquer vértice em um dado momento*.

Em adicional ao Lema 3, o seguinte lema garante que o intervalo de tempo entre duas visitas a uma aresta não pode ser muito grande. No caso de um único agente, quando o agente entra em circuito Euleriano, cada aresta é revisitada em um período de |E|. Para o caso multiagente (Yanovski et. al., 2002):

Lema 4: *Considere k agentes percorrendo uma rede. Assuma u→v foi visitada por um agente em um momento t. Esta aresta pode ser revisitada pelo menos mais uma vez até o tempo t = |E| + 1, por algum agente.*

Denotado por $d_t(u \rightarrow v)$ o número de vezes que a aresta $u \rightarrow v$ foi visitada em um momento t. Denotado por $d_t(u) = \min(d_t(u \rightarrow v))$ o menor valor de $d_t(u \rightarrow v)$ dentre todas arestas adjacentes a u. O próximo teorema fornece um limite para $|d_t(u \rightarrow v) - d_t(v)|$ (Yanovski et. al., 2002).

Teorema 2: *Suponha k agentes executando o algoritmo Edge_Ant_Walk em um grafo Euleriano direcionado. Então, em algum momento t para qualquer aresta e = u→v é correto:*

$$-k \leq d_t(e) - d_t(v) \leq k + 1$$

Corolário 3: *Suponha que k agentes executam o algoritmo Edge_Ant_Walk em uma rede. Então em algum momento t para duas arestas quaisquer $u_1 \rightarrow v$ e $v \rightarrow u_2$ é correto:*

$$|d_t(u_1 \rightarrow v) - d_t(v \rightarrow u_2)| \leq k + 1$$

Corolário 4: *Suponha que k agentes executam o algoritmo Edge_Ant_Walk em uma rede. Então em algum momento t para dois vértices u e v, é correto:*

$$|d_t(u) - d_t(v)| \leq d(u,v)(k + 1)$$

Onde, d(u, v) é o tamanho do menor caminho entre os vértices u e v.

O Teorema 3 mostra a propriedade de que o algoritmo da patrulha garante uniformidade de visitas nas arestas (Yanovski et. al., 2002).

Teorema 3: *Suponha k agentes executando o algoritmo Edge_Ant_Walk em uma rede. Para duas arestas quaisquer e_1 e e_2, é correto:*

$$\lim_{t \rightarrow \infty} \frac{d_t(e_1)}{d_t(e_2)} = 1$$

Uma questão interessante para k agentes executando o algoritmo EAW é saber se eles entram em movimento periódico para $k > 1$. Uma movimentação é periódica se a seqüência de arestas visitadas pelo grupo de agentes é periódica. O Lema 5 responde esta questão (Yanovski et. al., 2002):

Lema 5: *Depois de um período de tempo suficiente para k agentes executarem o algoritmo Edge_Ant_Walk em uma rede sincrônica, o algoritmo entra em movimento periódico.*

Durante um período cada aresta é percorrida o mesmo número de vezes. O tamanho T de um período pode ser um múltiplo de |E|/k.

4. PROPOSTA

Este trabalho propõe um novo algoritmo multiagente para patrulhar arestas de grafos Eulerianos dinâmicos.

Para Eppstein et. al. (1997), um *update* em um grafo é uma operação que insere ou remove arestas ou vértices do grafo ou troca atributos associados com arestas ou vértices, tal como custo e cor. Um grafo dinâmico é um grafo que passa por uma seqüência de *updates*.

Um problema típico com grafos dinâmicos é responder a consultas em grafos dinâmicos, tal como, se o grafo é conexo ou qual o menor caminho entre dois vértices. O objetivo de um algoritmo para grafo dinâmico é atualizar de forma eficaz a solução do problema depois das mudanças dinâmicas, sem ter que partir do zero outra vez (Eppstein et. al.,1997).

Grafos dinâmicos podem ser classificados quanto ao tipo de *update*. Um grafo dinâmico é dito plenamente dinâmico se as operações de *update* incluem inserção e remoção de arestas ou vértices. Um problema com grafos dinâmicos é dito parcialmente dinâmico se somente um tipo de *update*, inserção ou remoção, pode ser feito. Um problema com grafos parcialmente dinâmicos é dito incremental se somente inserções podem ocorrer e decremental se somente remoções podem ocorrer (Eppstein et. al.,1997).

O algoritmo de Yanovski et. al. (2002), Edge_Ant_Walk, é naturalmente um algoritmo dinâmico para o problema da patrulha, uma vez que é capaz de se adaptar a inserção ou remoção de nós ou arestas no grafo. A Figura 4.1 mostra o algoritmo EAW percorrendo as arestas de um grafo dinâmico. O esboço (1) da figura mostra o grafo. Os esboços (2) a (8) mostram as fases do algoritmo percorrendo as arestas até que no esboço (9) o agente percorre a aresta FB e o grafo sofre um *update,* com inserção do vértice G e das arestas DG e EG e remoção da aresta DE. É importante lembrar que qualquer inserção ou remoção de arestas ou nós deve garantir que o grafo continuará sendo Euleriano. Os esboços (10) a (16) mostram as fases do algoritmo após o *update*. As arestas mais grossas correspondem arestas que sofreram mais visitas e os nós sombreados ao local em que o agente se encontra.

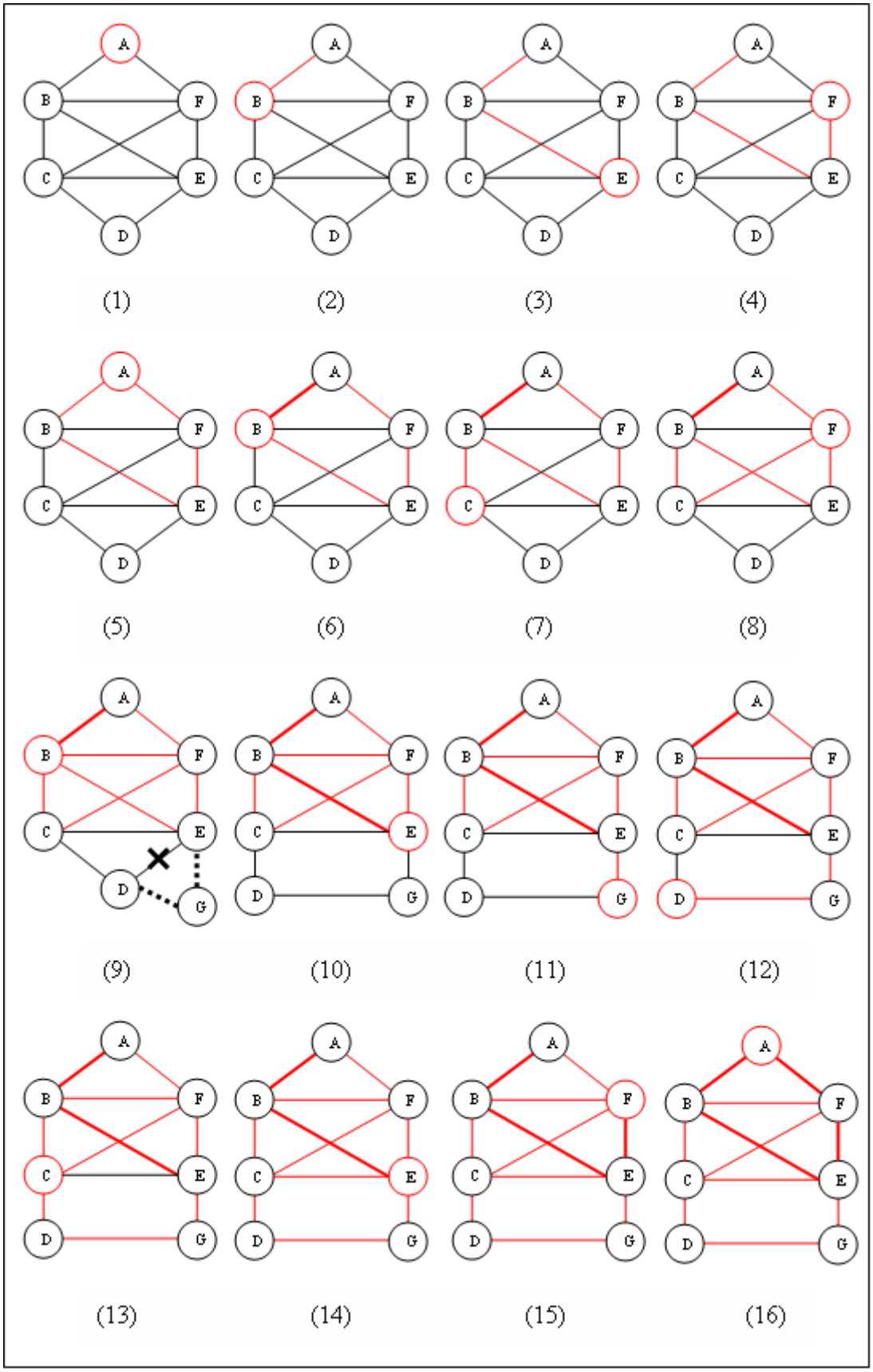

Figura 4.1: Um Exemplo do algoritmo Edge_Ant_Walk explorando um grafo dinâmico

Devido ao caráter dinâmico que o algoritmo de Yanovski et. al. (2002) pode assumir, este trabalho faz uma extensão do algoritmo EAW para um agente para o caso multiagente.

O algoritmo usa o Corolário 1 que provou que cada estágio exploratório por si só é um circuito Euleriano e o Corolário 2 que provou que um circuito Euleriano é um circuito limite do grafo, uma vez completo, será repetido indefinidamente para propor tal extensão. A idéia é percorrer todos os estágios exploratórios e depois encontrar um circuito Euleriano que cubra todo grafo e distribuir os agentes pelo circuito. O algoritmo pode ser formalizado com o seguinte pseudocódigo (Figura 9):

```
00 Multi_Agent_Ciclic_Edge_Walk(G(V,E), k)
01 Inicialização:
02   t ← 0; flag ← true; ok1 ← true; ok2 ← true;
03   NV(e) ← 0, σ(e) ← 0, para ∀e ∈ E;
04   Escolher aleatoriamente u ∈ V;
05
06   enquanto (ok)
07     Encontrar aresta e = (u,v), adjacente a u, com menor valor de σ(e)
08     se (NV(e) ≤ 3) então
09       se (NV(e)= 3) ou (!flag) então
10         flag = false;
11         Adicionar v ao circuito S;
12       fim_se;
13       σ(e) ← t;
14       NV(e) ← NV(e) + 1;
15       t ← t + 1;
16       u ← v;
17     senão
18       Definir nó de partida de cada agente;
19       ok ← falso;
20     fim_se;
21   fim_enquanto
22
23    enquanto (ok2)
24      Fase da patrulha, onde os agentes percorrem o circuito Euleriano
25      de forma cíclica;
26
27      se (G(E,V) mudou) então
28        ok2 ← false;
29      fim_se;
30    fim_enquanto;
```

Figura 4.2: Algoritmo Multi_Agent_Ciclic_Edge_Walk

O algoritmo recebe como parâmetros o grafo (*G(V,E)*) e a quantidade de agentes (*k*).

Na fase inicial (linhas 02 a 04 – Figura 4.2), todas as arestas $e \in E$ do grafo Euleriano G são inicializadas com $\sigma(e)$ e $NV(e)$ iguais a zero, já que nenhuma visita ocorreu ainda. $\sigma(e)$ é a última iteração em que a aresta e foi visitada e $NV(e)$ o número de visitas da aresta e. Em seguida, um vértice $u \in V$ é escolhido randomicamente.

Durante a segunda fase do algoritmo (linhas 06 a 21 – Figura 4.2) há dois momentos distintos. No primeiro (linhas 08 a 16 – Figura 4.2), o algoritmo encontra um circuito Euleriano que cobre todo grafo. Se $NV(e) \leq 2$ o algoritmo ainda encontra-se em um estágio exploratório, ou seja, ainda há arestas que não foram exploradas. Se $NV(e) \geq 3$, não há estágios exploratório e a partir daí encontra-se o circuito Euleriano do grafo. Para determinar o início do circuito a ser percorrido pelos agentes a variável *flag* é definida como falso.

No segundo momento (linhas 18 e 19 – Figura 4.2), os agentes são distribuídos pelo circuito. Quando o número de agentes é maior que o número de arestas, o número de agentes que participam da patrulha restringe-se ao número de arestas. O vértice de partida é calculado de forma que os agentes fiquem distribuídos pelos vértices do circuito Euleriano o mais eqüidistantes possível. Para $m = 1$ até qa agentes que participam efetivamente da patrulha, o vértice de partida dos agentes pode ser calculado pelas fórmulas:

(1) $start = m * (quocient + 1)$; ou

(2) $start = (rest * (quocient + 1)) + ((m - rest) * quocient)$.

Onde *quocient* é o inteiro da divisão do número de arestas ($|S|$) pela quantidade de agentes (qa) e *rest* é o resto dessa divisão. Se m for menor ou igual à *rest* aplica-se a equação (1) e caso contrário a equação (2).

Em seguida, na terceira fase (linhas 24 e 25 – Figura 4.2) acontece a patrulha, onde os agentes percorrem o circuito de forma cíclica, até que um novo grafo Euleriano seja criado (linha 27) e todo processo comece novamente. Entretanto, para evitar o aumento da ociosidade das arestas, que aconteceria até um novo estágio da patrulha começar novamente, o algoritmo foi implementado de forma paralela, de modo que o processo da patrulha possa continuar executando enquanto o novo circuito Euleriano é construído. Assim, somente depois que o circuito Euleriano é construído é que a patrulha mais antiga será substituída pela "nova".

Segundo Grötschel et. al. (2001), casos como este em que decisões têm que ser tomadas antes que informações completas sobre o problema estejam disponíveis são denominados como problemas *online*.

A Figura 4.3 mostra o grafo após seus estágios exploratórios, como ilustrado na Figura 3.2. A Figura 4.3.2 a Figura 4.3.11 mostra o circuito Euleriano proposto pelo algoritmo Multi_Agent_Ciclic_Edge_Walk, quando aplicado ao grafo da Figura 4.3.1.

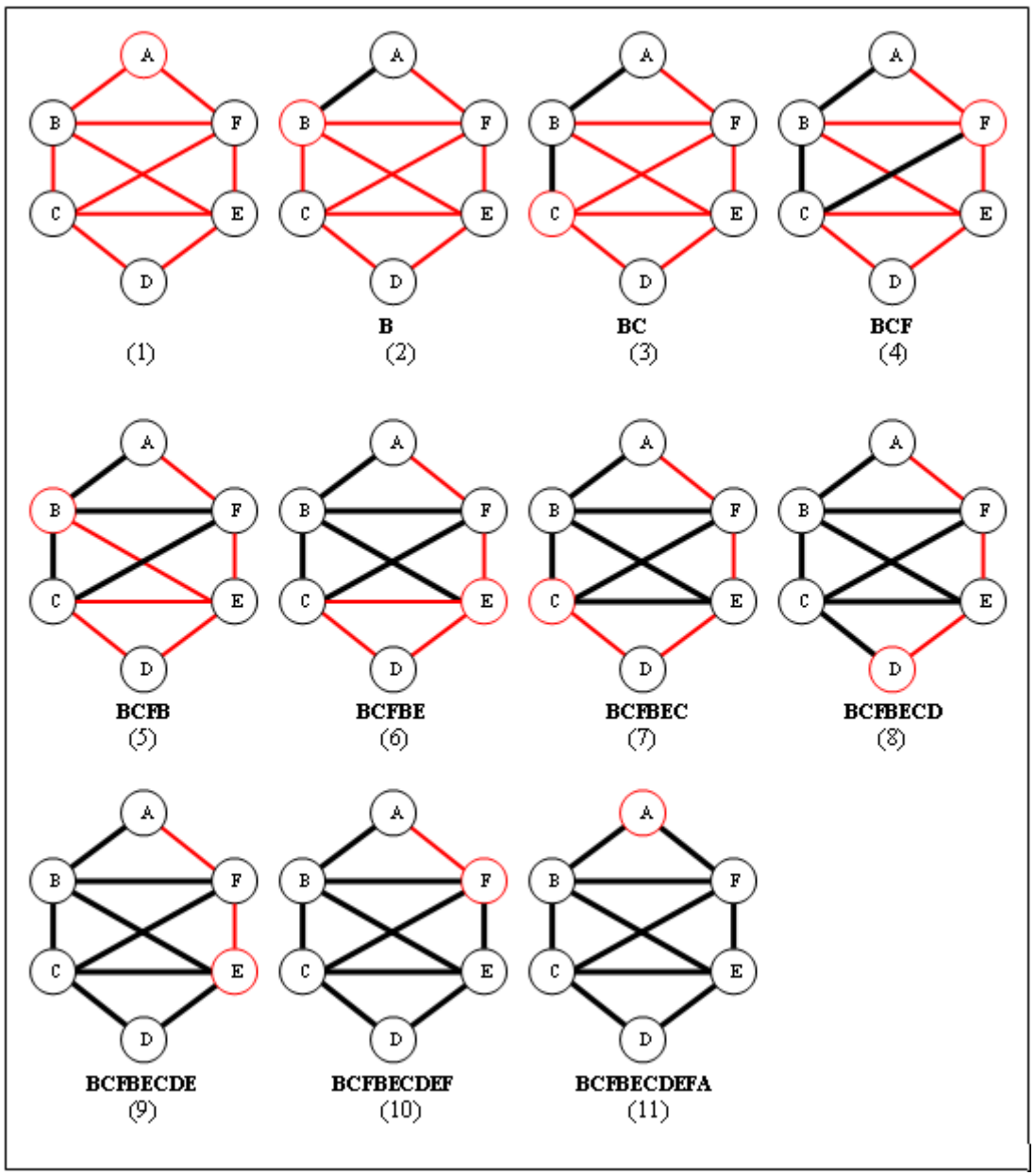

Figura 4.3: Um exemplo do Circuito Euleriano do algoritmo Multi_Agent_Ciclic_Edge_Walk

A Figura 4.4 apresenta os k = 4 agentes patrulhando o circuito Euleriano proposto na Figura 4.3. Em t_0 temos o circuito e em destaque os vértices de partida dos agentes. Em t_1, t_2 e t_3 os agentes estão patrulhando o grafo, ou seja, visitando suas arestas. Em cada unidade de tempo os agentes percorrem uma aresta, pois considera se que as arestas têm custo 1.

Analisando a Figura 4.4 pode-se observar que os agentes percorrem as arestas de grafo de forma cíclica, ou seja, os agentes partem de um vértice e retornam ao vértice de origem, e repetem o circuito indefinidamente. Em um determinado vértice, o próximo nó a ser visitado pelo agente é sempre o próximo vértice do circuito. Pode-se observar também que todos os k agentes participam da patrulha, pois o número de agentes é menor que o número de arestas do grafo.

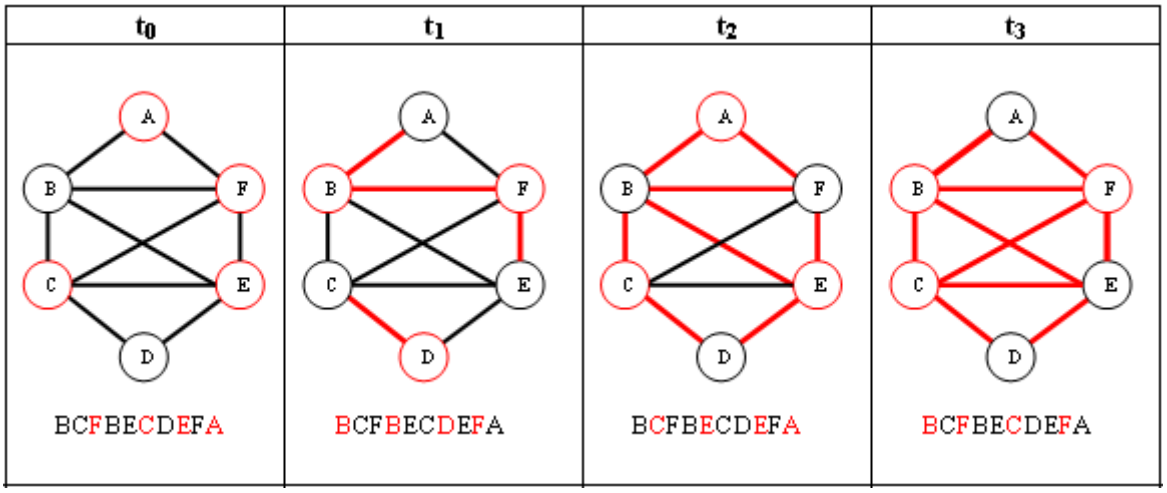

Figura 4.4: Um exemplo de k = 4 agentes patrulhando o grafo segundo o algoritmo Multi_Agent_Ciclic_Edge_Walk

5. METODOLOGIA

5.1. Tipo da Pesquisa

Bazzo e Pereira (2000) citado por Jung (2004) reforçam que "Pesquisar é um conjunto de investigações, operações e trabalhos intelectuais e práticos, que objetiva a descoberta de novos conhecimentos, a investigação de novas técnicas e a exploração ou criação de novas realidades".

Segundo Jung (2004) a pesquisa pode ser classificada quanto a natureza como básica quando tem por objetivo a aquisição de conhecimentos fundamentais a partir de estudos de fenômenos ou como tecnológica quando utiliza conhecimentos básicos, tecnologias existentes, conhecimentos tecnológicos e tem como objetivo um novo produto ou processo.

Quanto aos objetivos pode ser classificada como exploratória quando visa a descoberta, o achado, a elucidação de fenômenos ou a explicação daqueles que não eram aceitos apesar de evidentes. Como descritiva quando visa a identificação, registro e análise das características, fatores ou variáveis que se relacionam com o fenômeno ou processo. Ou como explicativa quando exige maior investimento em síntese, teorização e reflexão do objeto em estudo (Jung, 2004).

E, quanto aos procedimentos pode ser classificada como operacional quando trata do uso de ferramentas estatísticas e métodos matemáticos da otimização para a seleção do meio mais adequado para se obter o melhor resultado. Como experimental quando adquire conhecimentos por meio de experiências. Ou como estudo de caso quando é um procedimento que investiga um fenômeno dentro do contexto local, real e especialmente quando os limites entre fenômeno e o contexto não estão claramente definidos (Jung, 2004).

Sendo assim esta pesquisa é tecnológica, pois objetiva a implementação de um algoritmo, exploratória, pois visa conhecer fatos e fenômenos relacionados ao tema e operacional porque se utiliza de métodos matemáticos da otimização.

5.2. Procedimentos Metodológicos

A pesquisa foi realizada no período de março a maio de 2008. Inicialmente foi realizada uma revisão bibliográfica sobre as soluções propostas na literatura para o

problema da patrulha, o trabalho de Yanovski et. al. (2002) e técnicas computacionais para desenvolvimento e implementação dos algoritmos propostos. Foram consultados livros, monografias, teses e dissertações disponibilizadas na Internet e na literatura de modo geral.

Posteriormente foi implementado o algoritmo EAW de Yanovski et. al. (2002), o algoritmo Multi_Agent_Ciclic_Partition_Edge_Walk proposto neste trabalho e o algoritmo Multi_Agent_Partition_Edge_Walk para comparações dos resultados obtidos pelos três algoritmos.

Os algoritmos foram implementados utilizando Java Development Kit, versão 1.5.0, *copyright* 1994-2007 Sun Microsystems; e a biblioteca Colt, versão 1.2.0, uma biblioteca *open source* de alta performance de computações científicas e técnicas em Java, *copyright* (c) 1999 CERN - European Organization for Nuclear Research (Colt). A biblioteca foi utilizada devido ao seu gerador de números aleatórios baseados no método *Mersenne Twister* (Matsumoto e Nishimura, 1998) ser um dos melhores e mais rápidos geradores existentes.

5.3. Métricas

Segundo Almeida (2003) o objetivo do problema da patrulha é minimizar o intervalo de tempo entre duas visitas a um mesmo lugar, no caso as arestas, para todos os lugares do ambiente. Esta afirmação nos leva ao conceito de ociosidade.

Consideramos uma visita o tempo necessário para que um agente percorra uma aresta de modo completo. Tal visita tem sempre o mesmo custo, pois consideramos por questão de simplicidade, que todas as arestas têm distância constante ou que a transição é sempre feita em um mesmo intervalo de tempo.

Chamamos de ociosidade instantânea da aresta (o_t) a quantidade de tempo que a aresta permanece sem ser visitada.

As métricas utilizadas foram:

- Ociosidade Média de uma Aresta (*Average Edge Idleness* - AEI):

Definida como o somatório da ociosidade instantâneas da aresta em *t* dividido pelo número de visitas (*NV*) que a aresta sofreu até *t*.

$$AEI = \frac{\sum_{i=1}^{t} o_i}{NV}$$

- Ociosidade do Grafo (*Graph Idleness* - GI):

  Definida como a média da Ociosidade Média de uma Aresta (AEI).

$$GI = \sum_{i=1}^{|E|} \frac{AEI_i}{|E|}$$

- Pior Ociosidade da Aresta (*Edge Worst Idleness* - EWI):

A pior ociosidade é a maior ociosidade ocorrida na aresta. Definida como a maior ociosidade ocorrida na aresta em *t*.

$$EWI = max(o_i), \text{ para } i = 1 \ldots t$$

- Pior Ociosidade do Grafo (*Worst Idleness* – WI):

Definida como a maior ociosidade ocorrida no grafo, ou seja, a maior ociosidade dentre todas as arestas.

$$WI = max(EWI_i, \text{ para } i = 1 \ldots |E|)$$

- Média da Pior Ociosidade da Aresta (*Average Worst Idleness* - AWI):

  Definida como a média da pior ociosidade da aresta.

$$AWI = \sum_{i=1}^{|E|} \frac{EWI_i}{|E|}$$

No próximo capítulo são apresentados alguns resultados para as métricas definidas neste capítulo.

6. RESULTADOS E DISCUSSÕES

Para gerar os resultados foram feitas simulações utilizando-se grafos de 50 nós, com 10, 20 e 30 agentes patrulhando. Foram aplicados os mesmo grafos para o algoritmo proposto, Multi_Agente_Ciclic_Edge_Walk, e para os algoritmos Edge_Ant_Walk e Multi_Agente_Partition_Edge_Walk.

As métricas analisadas foram a Pior Ociosidade da Aresta e a Média da Pior Ociosidade da Aresta. A partir dessas métricas calculou-se a variância utilizando a seguinte fórmula (Oliveira et. al., 2004):

$$S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2$$

Onde, *n* foi adotado como número de arestas, x_i como a métrica Pior Ociosidade da Aresta e $\overline{x}$ como a métrica Média da Pior Ociosidade da Aresta.

Para efeito de comparação, calculou-se a média da Média da Pior Ociosidade da Aresta (*AWI*) e da variância (S^2) de cada grafo como mostra a tabela:

Tabela 6.1: Cálculo dos Resultados para Média da Pior Ociosidade e para Variância

Grafos	Multi_Agente_Ciclic_Edge_Walk	Multi_Agente_Partition_Edge_Walk	Edge_Ant_Walk
G_1	AWI_1, S_1^2	AWI_1, S_1^2	AWI_1, S_1^2
G_2	AWI_2, S_2^2	AWI_2, S_2^2	AWI_2, S_2^2
G_3	AWI_3, S_3^2	AWI_3, S_3^2	AWI_3, S_3^2
…	…	…	…
G_k	AWI_k, S_k^2	AWI_k, S_k^2	AWI_k, S_k^2
Média	$\sum_{i=1}^{k} \frac{AWI_i}{k}$ e $\sum_{i=1}^{k} \frac{S_i^2}{k}$		

A seguir são apresentados estes resultados:

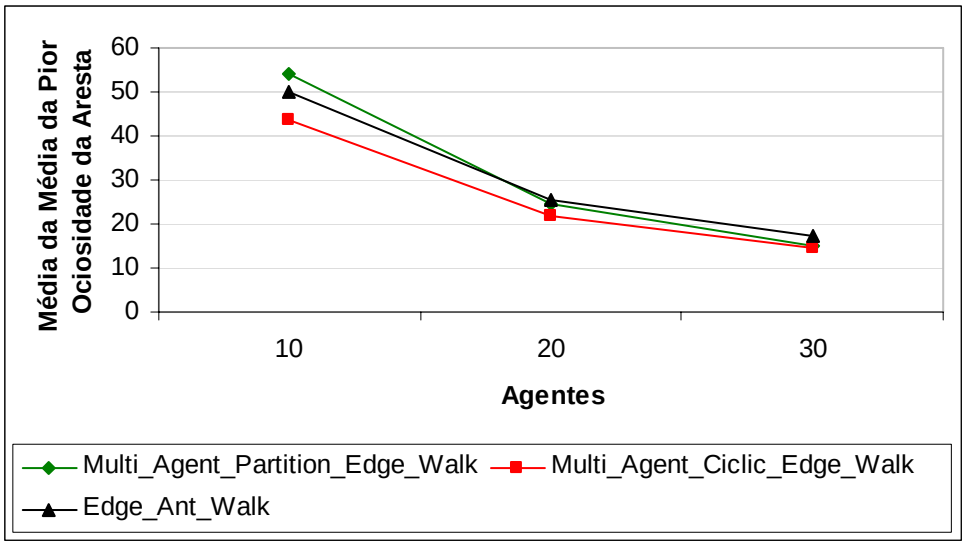

Figura 6.1: Média da Média da Pior Ociosidade da Aresta

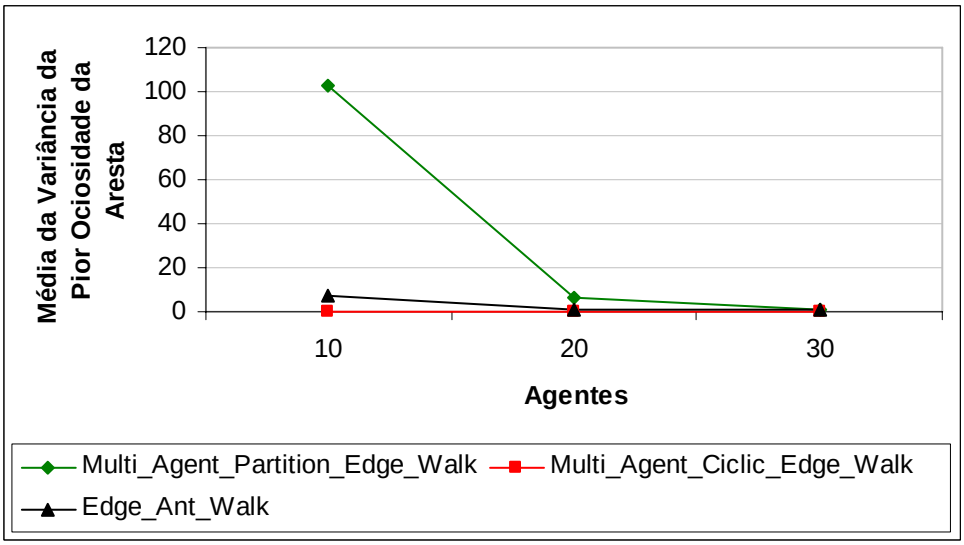

Figura 6.2: Média da Variância da Pior Ociosidade da Aresta

O algoritmo Multi_Agente_Ciclic_Edge_Walk obteve resultados melhores para a média da Média da Pior Ociosidade da Aresta como mostra a Figura 6.1 pois foi o que apresentou os menores valores de ociosidade. Os resultados para o algoritmo Multi_Agente_Partition_Edge_Walk foram os piores devido a possibilidade das partições geradas serem de tamanhos muito heterogêneos.

A partir dos dados da média da Variância da Pior Ociosidade da Aresta (Figura 6.2) observa-se que o algoritmo Multi_Agente_Ciclic_Edge_Walk é o que possui melhores resultados, uma vez que o valor da Pior Ociosidade da Aresta praticamente não varia.

As métricas Ociosidade Média de uma Aresta e Ociosidade do Grafo (média da Ociosidade Média de uma Aresta) também foram analisadas. A partir dessas métricas calculou-se a variância utilizando a fórmula anterior. Onde, *n* foi adotado como número de arestas, x_i como a métrica Ociosidade Média de uma Aresta e $\overline{x}$ como a métrica Ociosidade do Grafo (média da Ociosidade Média de uma Aresta).

Para efeito de comparação, calculou-se a média da Ociosidade do Grafo (*GI*) e da Variância (S^2) de cada grafo como mostra a tabela:

Tabela 6.2: Cálculo dos Resultados para Ociosidade do Grafo e para Variância

Grafos	Multi_Agente_Ciclic_Edge_Walk	Multi_Agente_Partition_Edge_Walk	Edge_Ant_Walk
G_1	GI_1, S_1^2	GI_1, S_1^2	GI_1, S_1^2
G_2	GI_2, S_2^2	GI_2, S_2^2	GI_2, S_2^2
G_3	GI_3, S_3^2	GI_3, S_3^2	GI_3, S_3^2
…	…	…	…
G_k	GI_k, S_k^2	GI_k, S_k^2	GI_k, S_k^2
Média	$\sum_{i=1}^{k} \frac{GI_i}{k}$ e $\sum_{i=1}^{k} \frac{S_i^2}{k}$		

A seguir são apresentados estes resultados:

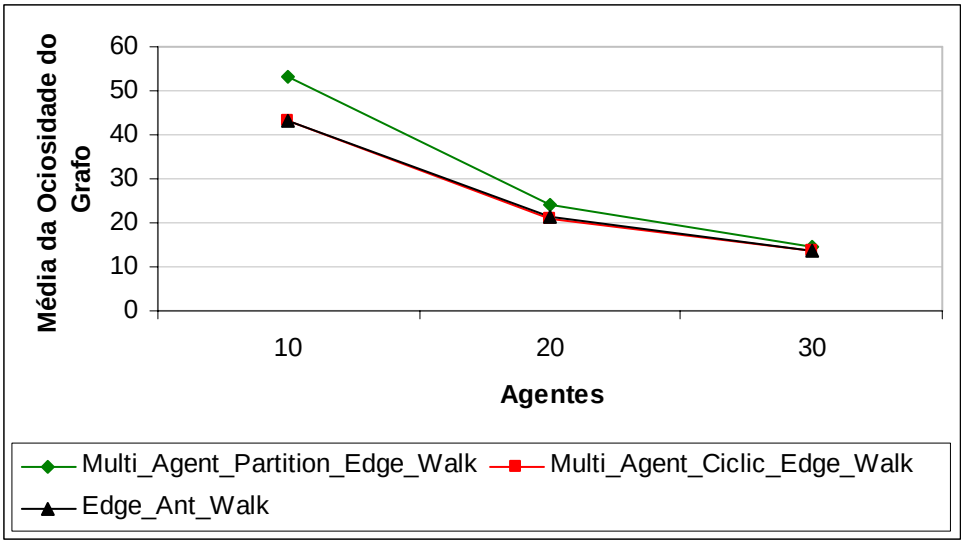

Figura 6.3: Média da Ociosidade do Grafo

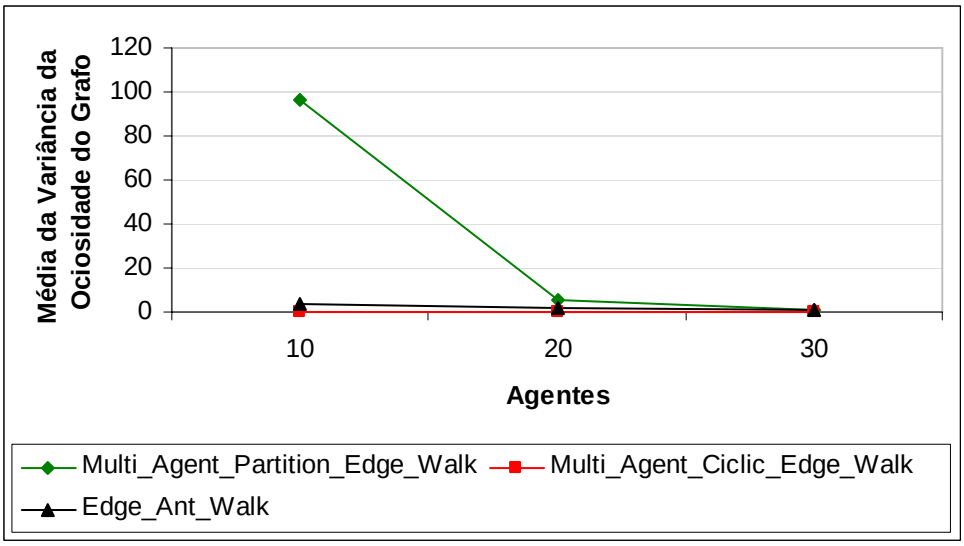

Figura 6.4: Média da Variância da Ociosidade do Grafo

Os algoritmos Multi_Agente_Ciclic_Edge_Walk e Edge_Ant_Walk obtiveram resultados semelhantes para a média da Ociosidade do Grafo como mostra a Figura 6.3. Os resultados para o algoritmo Multi_Agente_Partition_Edge_Walk foram piores devido a possibilidade das partições geradas serem de tamanhos muito heterogêneos.

A partir dos dados da média da Variância da Ociosidade do Grafo (Figura 6.4) observa-se que o algoritmo Multi_Agente_Ciclic_Edge_Walk é o que possui melhores resultados, uma vez que os resultados da média da Ociosidade do Grafo praticamente não varia.

7. CONCLUSÕES

Este trabalho propôs um novo algoritmo multiagente para patrulhar grafos Eulerianos Dinâmicos usando um circuito Eureliano obtido através da extensão do algoritmo de Yanovski et. al. (2002).

Foram implementados três algoritmos, Multi_Agente_Ciclic_Edge_Walk, Edge_Ant_Walk e Multi_Agente_Partition_Edge_Walk para fins de comparação.

Como medidas de desempenho foram adotadas as métricas: Ociosidade Média de uma Aresta, Ociosidade do Grafo, Pior Ociosidade da Aresta, Pior Ociosidade do Grafo e Média da Pior Ociosidade da Aresta. Em seguida, a partir destas métricas calculou-se também a variância para se estimar o quão uniforme são as ociosidades nas arestas.

Observando-se os resultados apresentados no capítulo anterior conclui-se que para os valores médios da métrica Média da Pior Ociosidade da Aresta (*AWI*), que o algoritmo proposto obteve melhores resultados, pois apresentou os menores valores de ociosidade e a variância foi praticamente nula, ou seja, a Média da Pior Ociosidade da Aresta foi praticamente a mesma para todas as arestas. Para os valores médios da métrica Ociosidade do Grafo (média da Ociosidade Média de uma Aresta - GI) o algoritmo proposto e o algoritmo de Yanovski obtiveram resultados semelhantes, salvo no que se refere à variância que apresentou resultados melhores no algoritmo proposto.

Os resultados para o algoritmo Multi_Agente_Partition_Edge_Walk foram os piores, devido ao fato do algoritmo particionar o grafo e de não haver garantias de que as partições tenham tamanhos o mais semelhante possível.

Assim fica como proposta para trabalhos futuros tentar combinar as partições geradas pelo algoritmo Multi_Agente_Partition_Edge_Walk de forma que minimize a diferença de tamanho entre as partições. No Apêndice A encontra-se a proposta para este algoritmo. Por fim, como trabalhos futuros sugerimos também estender o algoritmo proposto para trabalhar com grafos com pesos distintos, pois melhor representam problemas reais.

REFERÊNCIAS BIBLIOGRÁFICAS

ABATE, F. R. **The Oxford Dictionary and Thesaurus: The Ultimate Language for American Readers**, Oxford Univ. Press. 1996.

ALMEIDA, A. L.: **Patrulhamento Multiagente em Grafos com Pesos**. Dissertação de Mestrado. Recife, outubro/2003.

BAZZO, W. A. E PEREIRA, L. T. V. **Introdução à Engenharia**. 6 ed. Florianópolis: UFSC, 2000, p. 34.

CHEVALEYRE, Y. **Theoretical Analysis of the Multi-Agent Patrolling Problem**. Em IEEE/WIC/ACM International Conference on Intelligent Agent Techonology. Beijing, China, 2004.

COLT. **Colt.** Disponível em: http://dsd.lbl.gov/~hoschek/colt/index.html. Acessado em: 05/05/2008.

EPPSTEIN, D.,GALIL Z., e ITALIANO G. F. **Dynamic graph algorithms**. In CRC Handbook of Algorithms and Theory of Computation, Chapter 22. CRC Press, 1997.

GRÖTSCHEL, M., KRUMKE, S. O., WINTER, T., ZIMMERMANN, U. **Combinatorial Online Optimization in Real Time**. Berlin, 2001.

JUNG, C. F. **Metodologia para Pesquisa & Desenvolvimento** – Rio de Janeiro, RJ : Axcel Books do Brasil Editora, 2004.

JURKIEWICZ , S. **Grafos: Uma Introdução**. Disponível em: http://www.obmep.org.br/obmepcontent/estagio_bolsistas/apostila_estagio/mainColumnParagraphs/07/document/GrafosZ2-versao-atualizada25-06_FINAL_SemMarcasCorte.pdf. Acesso em: 02/04/2008.

LEITÃO, A. R. G. A. **Detecção em Sistemas Multiagentes para Patrulhamento**. Monografia de graduação. Recife, abril/2004.

MACHADO, A. P.: **Patrulha Multiagente: Uma Análise Empírica e Sistemática**. Dissertação de Mestrado. Recife, setembro/2002.

MATSUMOTO, M. and NISHIMURA T. **Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator**, ACM Transactions on Modeling and Computer Simulation, Vol. 8, Nº. 1, Janeiro 1998, pp 3--30.

OLIVEIRA, M. S., BEARZOTI, E., VILAS BOAS, F. L., NOGUEIRA, D. A., NICOLAU, L. A. **Introdução à Estatística**. Lavras: UFLA, 2004.

SANTANA, H. P. **Patrulha Multiagente com Aprendizagem por Reforço.** Dissertação de Mestrado. Recife, fevereiro/2005

SILVA, R. M. A.; SOUZA, D. J.; CHAVES, L. M.; RAMALHO, G. L.; MENEZES, F. S.; COELHO, T. A.; DEL VALLE, A. M. e PIRES, D. M. **A multi-agent edge patrolling on dynamic edge insertions/deletions eulerian graphs**. Icord VI, Fortaleza, 2007.

YANOVSKI, V.; WAGNER, I. A.; BRUCKSTEIN, A. M. **A Distributed Ant Algorithm for Efficiently Patrolling a Network**. Technical Report, 2002

WISON, R. J. **Introduction to Graph Theory.** 4 ed. Inglaterra: E. Longman, 1996.

Apêndice A – Algoritmo com Estratégia Baseada em Partição

O algoritmo usa o Corolário 1 que provou que cada estágio exploratório por si só é um circuito Euleriano para propor uma extenção do algoritmo de Yanovski et. al. (2002). A idéia é considerar cada partição como um circuito Euleriano induzido pelo conjunto $\bar{E}_m$ (com m = [1, ..., p], p é o último estágio exploratório), e depois, atribuí-los ao grupo de agentes, com número de agentes proporcional ao comprimento do circuito Euleriano. O algoritmo pode ser formalizado com o seguinte pseudocódigo (Figura A.1):

```
00 Multi_Agent_Partition_Edge_Walk(G(V,E), k)
01 Inicialização:
02   t ← 0; ok ← true; ok2 ← true;
03   NV(e) ← 0, σ(e) ← 0, para ∀e ∈ E;
04   Escolher aleatoriamente u ∈ V;
05
06   enquanto (ok)
07     Encontrar aresta e = (u,v), adjacente a u, com menor valor de σ(e)
08     se (NV(e) ≤ 2) então
09       se (NV(e)= 0) então
10         Adicionar v a partição;
11       senão
12         Adicionar partição ao conjunto das partições S;
13         σ(e) ← t;
14         NV(e) ← NV(e) + 1;
15         t ← t + 1;
16         u ← v;
17       fim_se;
18     senão
19       Para cada partição de S faça
20         qa ← ⌊1 + (k - |S|)*(|Si|/|E|)⌋;
21         Definir partição para cada agente;
22         Definir nó de partida de cada agente;
23       fim_para;
24       ok ← falso;
25     fim_se;
26   fim_enquanto;
27
28   enquanto (ok2)
29      Fase da patrulha, onde as partições atribuídas ao grupo de
30      agentes são executadas ciclicamente;
31
32      se (G(E,V) mudou) então
33        go to (01);
34      fim_se;
35    fim_enquanto;
```

Figura A.1 : Algoritmo Multi_Agent_Partition_Edge_Walk

Na fase inicial (linhas 02 a 04 – Figura A.1), todas as arestas $e \in E$ do grafo Euleriano G são inicializadas com $\sigma(e)$ e $NV(e)$ iguais a zero, já que nenhuma visita ocorreu ainda. $\sigma(e)$ é a última iteração em que a aresta e foi visitada e $NV(e)$ o número de visitas da aresta e. Em seguida, um vértice $u \in V$ é escolhido randomicamente.

Durante a segunda fase do algoritmo (linhas 06 a 26 – Figura A.1) há dois momentos distintos. No primeiro (linhas 08 a 17), o algoritmo constrói as partições, equivalente a cada circuito Euleriano encontrado durante os estágios exploratórios do algoritmo de Yanovski et. al. (2002). Para determinar se um agente está em um estágio exploratório, é suficiente verificar se o número de visitas (NV) sofridas pela aresta e atual é zero (linha 9). Se $NV > 2$, não há mais estágios exploratórios, é hora de começar o segundo momento: distribuir partições propostas entre os agentes.

No segundo momento (linhas 19 a 23 – Figura A.1), as partições propostas são distribuídas a k agentes, juntamente com o vértice de partida dos agentes. O vértice de partida é calculado de forma que os agentes que dividem a mesma partição fiquem distribuídos de forma eqüidistante pelos vértices da partição (subgrafo).

A quantidade de agentes qa que compartilham a mesma partição é calculada proporcionalmente ao comprimento da partição pela seguinte fórmula $qa \leftarrow \lfloor 1 + (k - |S|) * (|S_i|/|E|) \rfloor$ (linha 20).

Em seguida, na terceira fase (linhas 28 a 35 – Figura A.1) acontece a patrulha, onde as partições distribuídas ao grupo de agentes são executadas de forma cíclica, até que um novo grafo Euleriano seja criado (linha 32) e todo processo comece novamente. Entretanto, para evitar o aumento da ociosidade das arestas, que aconteceria até um novo estágio da patrulha começar novamente, o algoritmo foi implementado de forma paralela, de modo que o processo da patrulha possa continuar executando enquanto as novas partições estão sendo construídas e distribuídas entre os grupos de agentes. Assim, somente depois que as partições são construídas e distribuídas pelos agentes é que a patrulha mais antiga será substituída pela "nova".

Segundo Grötschel et. al. (2001), casos como este em que decisões têm que ser tomadas antes que informações completas sobre o problema estejam disponíveis são denominados como problemas *online*.

A Figura A.2.1 a Figura A.2.4 mostra os circuitos Eulerianos propostos como partições pelo algoritmo Multi_Agent_Partition_Edge_Walk durante suas fases

exploratórias, quando aplicado ao grafo ilustrado pela Figura A.2: **Um exemplo dos Circuitos Eulerianos do algoritmo Multi_Agent_Partition_Edge_Walk**

.1 e segue a mesma seqüência da Figura 3.2.2 a Figura 3.2.15.

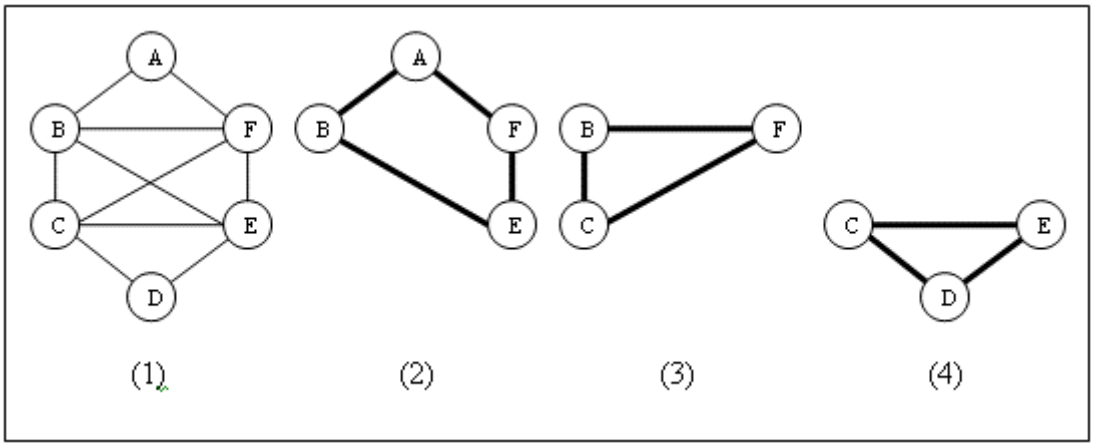

Figura A.2: Um exemplo dos Circuitos Eulerianos do algoritmo Multi_Agent_Partition_Edge_Walk

A Figura A.3 apresenta os agentes $k = 4$ agentes patrulhando os circuitos Eulerianos propostos como partições ou subgrafos na Figura A.3. Em t_0 temos as partições, a quantidade de agentes por partição e em destaque os vértices de partida dos agentes. Em t_1, t_2 e t_3 os agentes estão patrulhando o grafo, ou seja, visitando suas arestas. Em cada unidade de tempo os agentes percorrem uma aresta, pois considera se que as arestas têm custo 1.

Analisando a Figura A.3 pode-se observar que os agentes percorrem as arestas de grafo de forma cíclica, ou seja, os agentes partem de um vértice e retorna ao vértice de origem, e repetem este circuito indefinidamente. Pode-se observar também que apesar de inicialmente termos k = 4 agentes apenas 3 participam efetivamente da patrulha, isso ocorre porque a fórmula da quantidade de agentes acaba eliminando agentes.

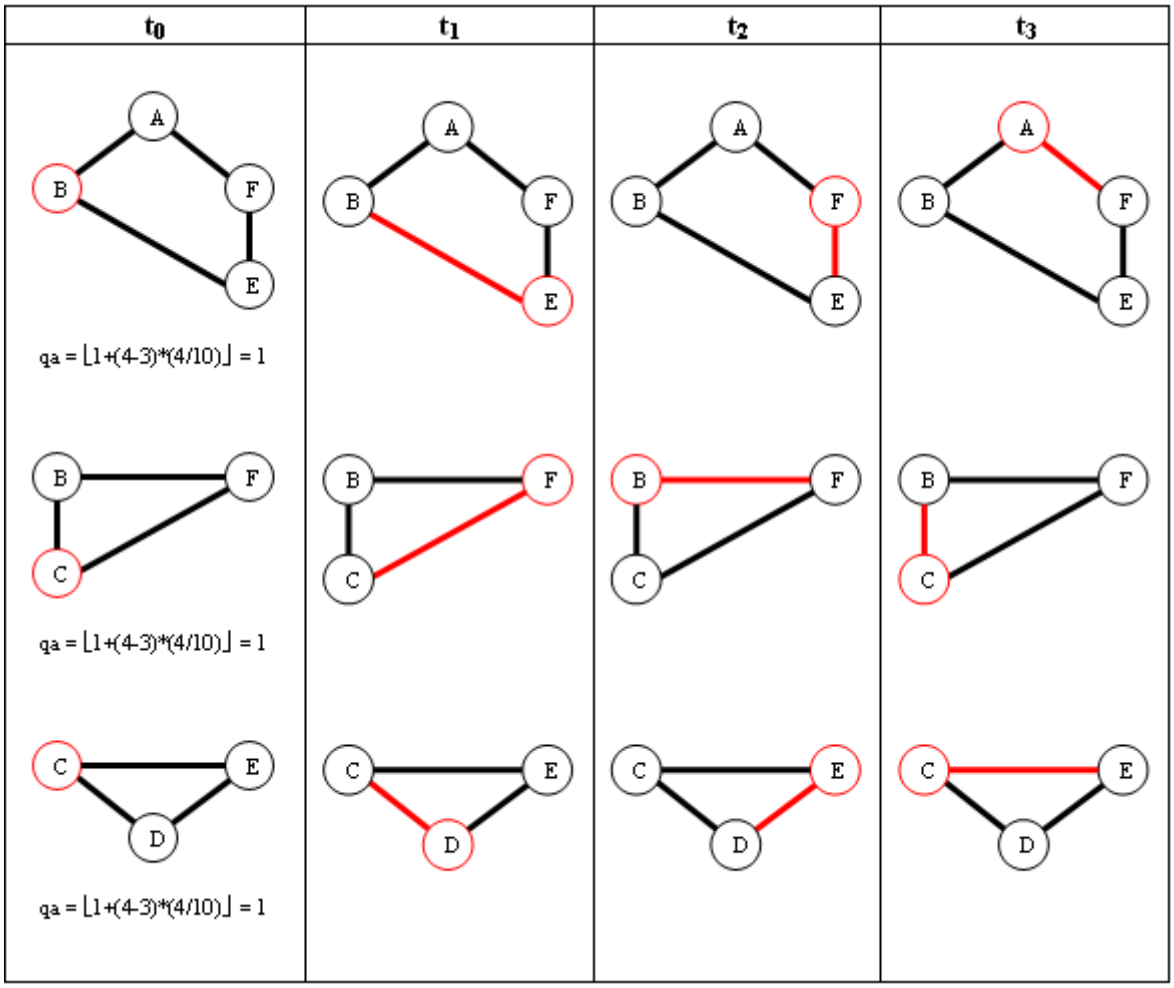

Figura A.3: Um exemplo de k = 4 agentes patrulhando o grafo segundo o algoritmo Multi_Agent_Partition_Edge_Walk