

Análise de algoritmos distribuídos com agentes móveis no cenário do projeto de Mineração de dados em tempo real para tomada de decisão no trânsito de uma cidade (Olhos da Cidade).

Lucas Schmidt Corrêa Franco, Ricardo Augusto Rabelo Oliveira
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFOP - Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brasil
email: lucasscf@gmail.com, rrabelo@gmail.com

Resumo—O projeto Olhos da Cidade em execução no DECOM / UFOP visa a mineração de dados em tempo real para tomada de decisão no trânsito de uma cidade. Neste contexto, há a necessidade do desenvolvimento de sistemas que consigam processar os dados colhidos por câmeras distribuídas em campo, em tempo real, recuperando informações importantes para tomada de decisão, com o menor atraso possível. Para recuperação deste tipo de informação, em tempo hábil, é necessário a utilização e abordagem de algoritmos distribuídos e agentes móveis como base para a recuperação de informação distribuída.

Neste trabalho, serão apresentados algoritmos distribuídos abordando agentes móveis que serão apresentados no contexto do projeto Olhos da Cidade. Estes algoritmos serão executados e modelados através de uma ferramenta de simulação de algoritmos distribuídos, chamada DAJ, e posteriormente serão apresentadas as devidas análises de complexidades e resultados.

Keywords—recuperação de informação, algoritmos distribuídos, agentes móveis, análise de complexidade.

I. INTRODUÇÃO

Com o crescimento da frota de veículos e consequentemente o aumento do congestionamento no trânsito das principais capitais do país, houve-se a necessidade e a preocupação no investimento de estudos e pesquisas que pudessem contribuir para a melhoria ou solução de tais problemas de trânsito. Este aumento na frota de veículos deve-se a: facilidade de crédito na venda de automóveis, conforto, suposta sensação de segurança e a facilidade de locomoção.

Os problemas de trânsito, resultantes do aumento da frota de veículos nas principais capitais, só podem ser melhorados ou solucionados com informações importantes capazes de indicar pontos críticos no trânsito da cidade estudada.

Para a realização de pesquisas e estudos, que envolvam recuperação deste tipo de informação, várias áreas são envolvidas, principalmente a área de Ciência da Computação. Esta, permite fornecer sistemas e técnicas que consigam processar os dados colhidos, usando mineração de dados de imagens obtidas de câmeras em campo, em tempo real, descobrindo informações importantes com tempo hábil na

apresentação dos resultados. Esta é a proposta do projeto Olhos da Cidade em execução no DECOM / UFOP.

No contexto deste projeto, pode-se destacar como principais recursos e técnicas da computação as abordagens de recuperação de informação, mineração de dados, processamento e recuperação de imagens e algoritmos distribuídos, este, tema principal deste trabalho, a fim de apresentar resultados para o problema que permitam a tomada de decisão em tempo factível.

Basicamente, os algoritmos distribuídos permitem que vários processos executem um mesmo algoritmo usando por meio da comunicação entre si, realizando trocas de mensagens (informações), entre os processos [1]. O objetivo de um algoritmo distribuído é possibilitar um alto processamento em baixo tempo de execução, usando por meio do compartilhamento de recursos [1]. Esta abordagem é semelhante ao contexto de sistemas distribuídos, porém menos abrangente. Um sistema distribuído pode ser definido como um conjunto de unidades de processamento independentes que podem processar uma aplicação em diferentes localidades, além de assumir elementos de rede diferenciados, como cliente e servidor [2].

Em uma analogia, um algoritmo distribuído pode ser efetivamente representado através de um grafo. Seja um grafo $G(V, E)$, onde V são os nós (vértices), representados por cada tarefa e E os meios de comunicação (arestas), representados pelas trocas de mensagens [1].

Um novo paradigma referente à computação distribuída vem atraindo muito a atenção de diversos pesquisadores. Trata-se do conceito de agentes móveis para aplicações distribuídas. Diversos pesquisadores vem tentando adaptar agentes móveis para diversas aplicações distribuídas, como descoberta de conhecimento, recuperação de informação distribuída, coleta de informação, mineração de dados e redes domésticas de comunicação móvel. Estudos comprovam que os agentes móveis podem ser utilizados de forma eficaz para aplicações distribuídas em um ambiente de sistema com uma baixa largura de banda de rede e frequente desconexão.

De acordo com estas informações, as pesquisas relacionadas, encontradas até o momento, são:

d-Agent: An Approach to Mobile Agent Planning for Distributed Information Retrieval [3], que propõe um novo algoritmo de planejamento de novos agentes chamado *d-Agent* capaz de realizar recuperação distribuída de informações de forma prática e realista. Em analogia aos grafos, para este algoritmo, existem n nós onde a informação distribuída reside. Em cada nó, um tempo de computação (recuperação da informação) é necessário para um agente móvel realizar a tarefa de recuperação naquele nó. Neste caso, o problema de planejamento é encontrar o número de agentes móveis e o itinerário de cada agente de modo que o tempo de rotação não seja maior do que o tempo determinado com o menor gasto de recurso de sistema. Pelo fato deste problema de planejamento ser NP-difícil, a solução é baseada em heurística.

A outra proposta conhecida como *Cost-Effective Planning of Timed Mobile Agents* [4] propõe uma solução ideal para o problema de planejamento de agentes móveis e gerenciamento do tempo de recuperação da informação distribuída. Tal método é aplicável ao tempo limitado de recuperação de informação distribuída para problemas que tem janelas de tempo, que consiste em um limite inferior e um *deadline* como limite superior.

Com isso, este trabalho propõe um estudo das duas propostas citadas acima e suas devidas análises de complexidade, a fim de propor uma solução a nível de algoritmos distribuídos que poderão ser utilizados no projeto Olhos da Cidade. Além disso, os algoritmos serão modelados através de uma ferramenta de algoritmos distribuídos chamada DAJ, que representará o cenário proposto neste trabalho. O DAJ é um pacote para Java que permite a simulação, a nível de grafos, capazes de representar uma estrutura de aplicação distribuída em execução.

Sendo assim, este projeto está organizado da seguinte forma: A Seção II apresenta os trabalhos já estudados relacionados ao tema deste trabalho. A Seção III apresenta os métodos e metodologias estudadas para realização deste projeto. As análises de complexidade estão apresentados na Seção IV. A Seção V apresenta os experimentos realizados para este trabalho, o estudo e comentários dos resultados obtidos. Posteriormente, propostas de estudos e novos trabalhos relacionados ao tema são sugeridos na Seção VI. A Seção VII apresenta as conclusões de análises e estudos deste trabalho.

II. TRABALHOS RELACIONADOS

Recentemente, muitas pesquisas sobre agentes móveis e computação distribuída vem sendo realizadas. No caso de agentes móveis, as pesquisas mais relacionadas abordam sistemas com um baixa largura de banda de rede e frequentes desconexões para recuperação de informação distribuída, como é facilmente encontrado na Internet e computação móvel.

Baseado nisso, outro contexto relacionado é abordado como problema de planejamento de agentes móveis. A idéia do MAP (*Mobile Agent Planning*), é o planejamento para a distribuição de agentes móveis. No caso do MAP, agentes móveis devem ter o conhecimento das condições da rede para que eles possam se adaptar ao ambiente.

Na literatura encontramos algumas referências que tratam de problemas de MAP como: *Mobile Agent Planning Problems* de K. Moizumi e *The traveling agent problem* de K. Moizumi e G. Cybenko.

Outros trabalhos consideráveis e relacionados à este projeto são os que abordam problemas de recuperação de informação através de imagens de câmeras remotas em grandes cidades para reconhecimento de padrão.

III. MÉTODOS

Os algoritmos distribuídos permitem que vários processos executem um mesmo algoritmo usando por meio da comunicação entre si, realizando trocas de mensagens (informações), entre os processos [1]. Os objetivos de um algoritmo distribuído é possibilitar um alto processamento em baixo tempo de execução, usando por meio do compartilhamento de recursos [1].

Em uma analogia, um algoritmo distribuído pode ser efetivamente representado através de um grafo. Seja um grafo $G(V, E)$, onde V são os nós (vértices), representados por cada tarefa e E os meios de comunicação (arestas), representados pelas trocas de mensagens [1].

Os algoritmos distribuídos básicos encontrados na literatura são: *PI* (Propagação de Informação), *Test Connectivity* (Conectividade) e *Compute Distances* (Menor Distância) [1].

Podemos dizer que o algoritmo *PI* representa um grafo G não dirigido e dissemina a informação *inf*. n é o número de nós em G e m é o número de arestas em G [1]. O problema do algoritmo *PI* é difundir em G informações presentes num subconjunto dos nós de G . Como solução para o problema, supondo que n_i seja um nó que tenha *inf*, é necessário calcular uma árvore geradora em G e utilizar esta árvore para fazer a difusão de *inf*. Posteriormente n_i envia *inf* em todas as arestas da árvore geradora que são incidentes a n_i . Este processo é repetido por todos os outros nós, exceto na aresta onde *inf* foi recebida. Com isso, se a árvore geradora foi computada anteriormente, um algoritmo assíncrono baseado nessa estratégia tem complexidade de tempo e mensagem de $O(n)$ [1].

O algoritmo *Test Connectivity* representa um grafo G e seu problema consiste em cada nó (vértice) em N que deve descobrir as identificações de todos os outros nós aos quais está conectado por um caminho em G [1]. No caso de sistemas sujeitos a falhas, é importante saber os identificadores dos nós de um componente conectado. No contexto de solução, este algoritmo não é adequado para casos onde G altera-se dinamicamente. No caso, basicamente, ele mostra o uso da técnica do algoritmo *PI* [1].

Já o algoritmo *Compute Distances* possui como problema determinar a menor distância em G entre todos os pares de nós (vértices), onde esta distância é medida em número de arestas através de pulsos síncronos [1]. Neste caso cada nó (vértice) possui uma identificação onde se $s = 0$, cada nó envia sua identificação para os seus vizinhos, se $s = i$, sendo $i > 0$, cada nó envia para os vizinhos as identificações recebidas durante $s - 1$ e $N_0 = N$ [1].

Estes algoritmos na maioria das vezes podem ser utilizados como complemento para a solução de outros algoritmos distribuídos mais complexos e completos. Estes algoritmos, muitas vezes, são mais específicos a um tipo de problema e podem, as vezes, ser baseados em heurísticas e algoritmos aproximados [1].

Aprofundando no mundo da computação distribuída e móvel, outro paradigma deve ser considerado. Trata-se do conceito de agentes móveis para aplicações distribuídas. Os agentes móveis podem ser adaptados em diversas aplicações distribuídas, como descoberta de conhecimento, recuperação de informação distribuída, coleta de informação, mineração de dados e redes domésticas de comunicação móvel [5].

Basicamente, os agentes móveis podem ser comparados a agentes ou programas que podem migrar entre computadores de uma rede durante a sua execução, carregando consigo o seu estado de execução. No caso, podem viajar de um nó para o outro, diferentemente da computação distribuída tradicional, como o RPC (*Remote Procedure Call*). Para o RPC, a comunicação é feita computador à computador, onde um cliente chama procedimentos em um servidor. Já no caso de agentes móveis, este usa RP (*Remote Programming*), na qual um programa além de poder fazer chamadas de procedimentos em outro computador, também pode executar seus procedimentos sobre outro computador [6] [7].

Muitos pesquisadores têm relatado que os agentes móveis, em comparação com os paradigmas de computação convencional, pode reduzir o tráfego de rede, superar a latência da rede e melhorar a tolerância a falhas e robustez de aplicações distribuídas [6].

Em particular, os agentes móveis podem ser utilizados de forma eficaz para aplicações distribuídas em um ambiente de sistema com uma baixa largura de banda de rede e frequente desconexão [7].

Uma das principais áreas de potencial aplicação para agentes móveis é a de recuperação de informação distribuída, onde uma enorme quantidade de dados podem ser acessadas através de uma rede [8]. Com o rápido crescimento da Internet e da computação móvel, podemos facilmente ver que serviços de informação e computação estão se espalhando em vários nós dispersos geograficamente. Com isso, a utilização de recuperação de informações distribuídas é agora amplamente utilizado [8]. Com base nessa idéia de agentes móveis, em vez de transmitir grandes quantidades de dados através da rede, agentes móveis migram para os nós onde a informação está localizada e o processo de sua

tarefa de recuperação é realizada no nó presente. A partir daí, os agentes móveis finalmente voltam para o nó original, resultando se a sua execução foi bem sucedida [8].

Alguns recursos de recuperação de informação podem gastar muito tempo de espera para carregar grandes quantidades de informação distribuída, e necessitam de conexões de rede permanente. Os agentes móveis, portanto, podem ajudar a aliviar essa sobrecarga na recuperação com boas técnicas de planejamento de distribuição dos agentes [9]. Com isso, entra em questão uma técnica conhecida como MAP (*Mobile Agent Planning*), que é usada para alocar agentes para um caminho eficiente antes de distribuí-los, usando dados estatísticos atuais sobre o tráfego da rede [9] [10] [11]. No MAP, os dois fatores importantes são o planejamento do número de agentes móveis e o itinerário de cada agente móvel. O sistema de agente pode ter um melhor desempenho quando se usa um plano bem estruturado para os agentes móveis. Em outras palavras, com um itinerário cuidadosamente planejado, um agente móvel pode ser migrado para o nó de destino a baixo custo em condições normais da rede [9] [10] [11].

Outro fator importante a ser considerado são as limitações de tempo que podem residir nos nós de repositório de informações, que são servidores de informação, aos quais também precisam ser tratados no mecanismo que realizará o planejamento de agente móveis [12] [13]. Considerando os nós que apresentam informações apenas para algum intervalo de tempo, tais como os serviços de *clipping* de notícias, bolsa de valores, ou de câmbio, se um agente é enviado e chega mais cedo do que um tempo de atualização especificado para coletar informações, este pode recuperar informações inúteis ou corrompidas [12] [13].

Devido aos avanços na rede doméstica, Internet e principalmente na computação móvel, o tempo de resposta à uma recuperação de informação se torna a cada dia um fator de grande importância. Os usuários, assim como sistemas de recuperação de informação, na maioria das vezes, necessitam de respostas rápidas do sistema, enquanto ao mesmo tempo necessitam de menor custo possível [8].

Com isso, duas propostas de algoritmo distribuído baseadas em agentes móveis foram estudadas e analisadas a fim de contribuir para a solução do projeto Olhos da Cidade. Estas duas propostas são conhecidas como: *d-Agent: An Approach to Mobile Agent Planning for Distributed Information Retrieval* [3] e *Cost-Effective Planning of Timed Mobile Agents* [4], e serão o foco de estudo deste trabalho.

Para o entendimento das propostas citadas acima, é necessário o conhecimento de algumas notações [3] [4]:

Símbolo	Descrição
H	Nó origem
n	Número de nós processados
r	Número de agentes móveis
δ	Tempo de computação (recuperação)
h_1, h_2, \dots, h_n	Identificação de nós
A_1, A_2, \dots, A_n	Identificação de agentes móveis
$P(A_j)$	Itinerário (caminho) por agente
$L_S(h_i, h_j)$	Menor latência entre nós h_i e h_j
$C(h_i, h_j)$	Cálculo de tempo (de recuperação)
$TourTime(*)$	Tempo de roteamento de um itinerário
$Dist(h_i, h_j)$	Tempo de distância entre h_i e h_j

A. *d-Agent: An Approach to Mobile Agent Planning for Distributed Information Retrieval* [3]

A proposta do *d-Agent: An Approach to Mobile Agent Planning for Distributed Information Retrieval* [3] é oferecer uma nova técnica de planejamento de agentes, chamada de *d-Agent*, capaz de realizar recuperação distribuída de informações de forma prática e realista.

Em analogia aos grafos, para este algoritmo, existem n nós onde a informação distribuída reside. Em cada nó, um tempo de computação (recuperação da informação), conhecido como *turn-around*(δ), é necessário para um agente móvel realizar a tarefa de recuperação naquele nó [3].

O *d-Agent* possui um modelo de planejamento de agentes, para a recuperação de informação distribuída, que é focada no tempo *turn-around*(δ). Para descrever este modelo de planejamento, é apresentado duas definições importantes [3]:

Definição 1 (O tempo de roteamento, $TourTime(S)$)

$TourTime(S)$ é o tempo de encaminhamento para o especificado itinerário S , onde pode ser calculado conforme abaixo [3]:

$$S = (h_1, h_2, h_3, \dots, h_m) \quad (1)$$

$$TourTime = (h_1, h_2, h_3, \dots, h_m) \quad (2)$$

$$TourTime = L_S(H, h_1) + C(h_1) + \quad (3)$$

$$L_S(h_1, h_2) + \dots + L_S(h_m, H)$$

$$TourTime = L_S(H, h_1) + L_S(h_m, H) + \quad (4)$$

$$\sum_{i=1}^m C(h_i) + \sum_{i=1}^{m-1} L_S(h_i + h_{i+1})$$

Definição 2 (O tempo de distância $Dist(h_i, h_j)$)

$Dist(h_i, h_j)$ é a distância entre h_j e h_i e pode ser representados matematicamente conforme abaixo [3]:

$$Dist(h_i, h_j) = L_S(h_i, h_j) + C(h_j) \quad (5)$$

Basicamente, o problema de planejamento consiste em encontrar o número de agentes móveis e o itinerário de cada agente de modo que o tempo de rotação não seja maior do

que o tempo determinado com o menor gasto de recurso de sistema. Pelo fato deste problema de planejamento ser NP-difícil, a solução é baseada em heurística.

No caso do algoritmo, primeiramente é necessário processar o grafo original para extrair a menor latência da rede ou SNL (*Shortest-Latency Network*). Esta etapa é executada uma única vez em todos os processos de planejamento. Posteriormente, os nós não alcançados, que possuem tempos de encaminhamento maior que o *turn-around*(δ), devem ser identificados e excluídos. Agentes móveis, mesmo com um esquema de planejamento perfeito, podem não processar esses nós dentro do prazo de δ [3]. O terceiro passo, é encontrar o número de agentes móveis e a sequência de nós (itinerários) para cada agente, usando o tempo *turn-around*(δ) dado, e posteriormente é utilizado um algoritmo TSP (*Travel Salesman Problem*), ou mais conhecido como problema do caixeiro viajante, como por exemplo o algoritmo 2OPT, para determinar itinerários individuais de agentes móveis para minimizar o tempo total de roteamento, a fim de otimizar o caminho de roteamento de cada agente móvel [3]. O algoritmo 2OPT é um algoritmo de busca local simples proposta inicialmente por Croes em 1958 para resolver o problema do caixeiro viajante. A idéia principal por trás dele é tomar um caminho que se cruza e reordená-lo a fim de evitar este cruzamento. Finalmente, os agentes móveis são alocados e implementados em cada partição de rede [3].

Tal processo pode ser melhor visualizado no exemplo abaixo [3]:

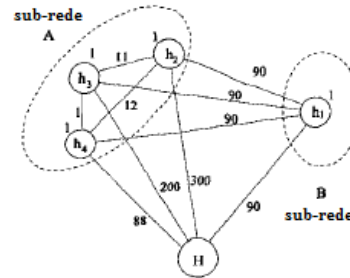


Figura 1. a) Configuração da rede

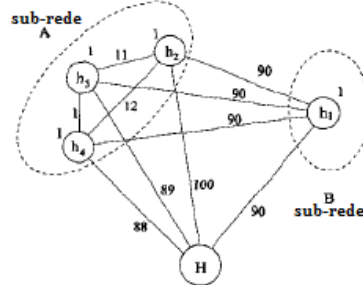


Figura 2. b) Menor latência da rede (SLN)

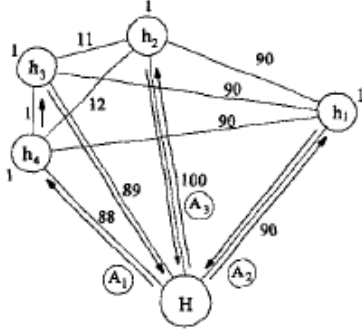


Figura 3. c) Planejamento com $\delta = 201$

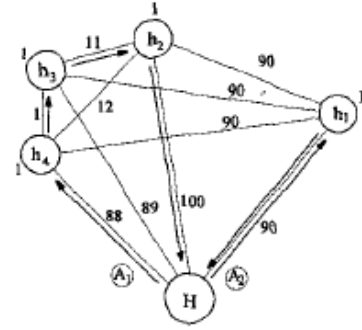


Figura 4. d) Planejamento com $\delta = 203$

Algoritmo de planejamento *d-Agent* [3]:

```

Algoritmo d-Agent
Entrada  $\delta$ 
Saída  $r$  e  $P(A_i)$ 
 $k \leftarrow 0$ ;  $NoAtual \leftarrow ProximoNo \leftarrow H$ 
Função main()
1 determinar status de  $NAO - PROCESSADO$  para todos os nós
2 faça
3    $k \leftarrow K + 1$ 
4    $P(A_k) \leftarrow NULL$ 
5    $P(A_k) \leftarrow Partition()$ 
6 enquanto  $P(A_k)$  for diferente de  $NULL$ 
7   para  $i$  de 1 para  $k$  faça
8      $2OPT(P(A_i))$ 
9   devolva  $k - 1$  e  $P(A_i)$ ,  $1 \leq i \leq k$ 
Função Partition()
1 enquanto (1) faça
2   determinar nós não VISITADOS e que não tenham sido PROCESSADOS;
3    $ProximoNo \leftarrow h_i$  encontrado, o nó com a mínima  $Dist(NoAtual, h_i)$  que não tenha sido PROCESSADO ou VISITADO;
4   se  $h_i$  não for encontrado então
5     devolva  $P(A_i)$ 
6   se  $TempoRoteamento(Add(P(A_i), ProximoNo)) > \delta$  então
7      $Del(P(A_i), ProximoNo)$ 
8     Marca  $ProximoNo$  como VISITADO
9   senão
10    Marca  $ProximoNo$  como PROCESSADO
11     $NoAtual \leftarrow ProximoNo$ 

```

B. Cost-Effective Planning of Timed Mobile Agents [4]

A proposta apresentada no artigo *Cost-Effective Planning of Timed Mobile Agents* [4] oferece métodos conhecidos como TCEMAP1 e uma pequena extensão do algoritmo conhecido como TCEMAP2 que apresentam uma solução

ideal para o problema de planejamento de agentes móveis e gerenciamento do tempo de recuperação da informação. Tal método é aplicável ao tempo limitado de recuperação de informação distribuída para problemas que tem janelas de tempo, que consiste em um limite inferior e um *deadline* como limite superior [4]. Adotando esses métodos de planejamento, os sistemas de recuperação de informação podem manter um ótimo desempenho e atingir o mínimo de sobrecarga da rede com o número mínimo de agentes sob as limitações de tempo atribuído [4].

O TCEMAP em geral parte do pressuposto de algumas suposições características de agentes móveis. Primeiramente, é necessário assumir que os agentes móveis conhecem as estatísticas de rede, tais como, a latência e largura de banda das ligações, a carga computacional e as restrições de tempo da informação em cada nó através de serviços de monitoramento de rede e através de dados históricos [4]. Neste caso, outra consideração importante é que latências de rede e cargas de computação podem variar. Em seguida, é necessário supor que os agentes móveis são criados apenas no nó origem [4].

No caso do TCEMAP que considera a restrição de tempo, também é necessário considerar apenas dois tipos de servidores que existem para sistemas de recuperação distribuída; os servidores ingênuos e os cronometrados. O servidor cronometrado difere do normal na medida em que fornece informações corretas em uma determinada janela de tempo. A janela de tempo é composta de um limite inferior e um *deadline* como limite superior [4].

Para efeitos do processo de recuperação distribuída, um agente móvel deve migrar para algum nó, que tem informações para processar. Se o nó tem janela de tempo, um agente móvel deve estacionar naquele nó antes ou no limite inferior de tempo daquele nó. Naturalmente, o tempo que o agente móvel tem para concluir a execução de sua tarefa não deve exceder o *deadline* ou limite superior [4].

Com isso, o número de agentes móveis e custo de roteamento de cada agente móvel afeta diretamente a largura da banda de rede. Assim, buscou-se determinar o número mínimo de agentes necessários e a quantidade mínima de rotas a serem percorridas pelos agentes móveis, conseguindo enfim, o tempo mínimo para a conclusão de execução de uma tarefa.

Basicamente, para esses tipos de problemas que envolvem planejamento de agentes móveis o algoritmo comum encontrado em diversas aplicações parte da ordenação dos nós em ordem decrescente de tempo de execução de cada nó, conforme abaixo [4]:

$$\max(Ready(h_i), L_S(h_i, H)) + Comp(h_i) + L_S(h_i, H), 1 \leq i \leq n \quad (6)$$

Posteriormente, define o limite mínimo com o tempo de

execução do primeiro nó da lista, conforme pode ser visto na expressão abaixo [4]:

$$\text{Ready}(h_j) \leq \text{Start}(A_i, h_j) \text{ and } \text{End}(A_i, h_j) \leq \text{Dead}(h_j), 1 \leq i \leq r, 1 \leq j \leq n \quad (7)$$

Com isso, o próximo passo é particionar a rede em várias partes, por nós de coleta, para que o tempo de execução de cada parte não exceda o limite permitindo criar um caminho de roteamento para cada partição. Posteriormente, é utilizado um algoritmo TSP (*Travel Salesman Problem*), ou mais conhecido como problema do caixeiro viajante, para otimizar o caminho de roteamento de cada agente móvel. Logo após, os agentes são alocados e distribuídos nas partições [4].

Sendo assim, o TCEMAP1 e o TCEMAP2 executam a mesma tarefa que o algoritmo descrito acima, porém usam um método ligeiramente diferente de particionamento [4]. A diferença entre o TCEMAP1 e TCEMAP2 é que este último possui propriedades mais dinâmicas que o primeiro algoritmo. O TCEMAP1 tenta encontrar a próxima possível partição calculando todas as latências a partir do nó origem, enquanto o TCEMAP2 procura o próximo nó a partir do nó no qual o agente reside [4].

Algoritmo de planejamento TCEMAP1 [4]:

```

Algoritmo TCEMAP1
  // Passo 1: Ordenação dos nós
  1 Ordenação dos nós em ordem decrescente de tempo de execução de cada nó
  2 define o limite mínimo com o tempo de execução do primeiro nó da lista
  // Passo 2: Planejamento de agentes móveis
  3  $k \leftarrow 0$ 
  4 faça
  5    $i \leftarrow i + 1$ 
  6    $\text{Tour}(A_i) \leftarrow \text{AtribuirNo}(0, \delta, H, H)$ 
  7 enquanto  $\text{Tour}(A_i) == \text{NULL}$ 
  // Passo 3: Otimização do caminho do Agentes na rede
  8 para cada agente  $A_i$  faça
  9   se  $\text{PrimeiroNo}$  já foi processado e o valor da latência entre o nó
  origem ( $H$ ) e este nó for menor do que já foi processado então
  10      $\text{Depart}(A_i) \leftarrow \text{Ready}(\text{PrimeiroNo}) -$ 
 $L_S(H, \text{PrimeiroNo})$ 
  11      $2\text{OPT}(\text{Tour}(A_i))$ 
  Função AtribuirNo( $\text{Menor}, \text{Maior}, \text{MinNo}, \text{MaxNo}$ )
  1  $\text{Seq1} \leftarrow \text{Seq2} \leftarrow \text{Seq3} \leftarrow \text{NULL}$ 
  2 para cada  $i_{ak}$  onde  $i_{ak}$  está NAO – PROCESSADO e  $ak$  é o menor
  número faça
  3    $\text{Inicio} \leftarrow \text{MaxReady}(i_{ak}), \text{Menor} + L_S(\text{MinNo}, i_{ak})$ 
  4    $\text{Fim} \leftarrow \text{Inicio} + \text{Comp}(i_{ak})$ 
  5   se  $\text{Fim} + L_S(i_{ak}, \text{MaxNo}) \leq \text{Maior}$  então
  6      $\text{Marca } i_{ak} \text{ como PROCESSADO}$ 
  7      $\text{Seq1} \leftarrow \text{AtribuirNo}(\text{Menor}, \text{Inicio}, \text{MinNo}, i_{ak})$ 
  8      $\text{Seq2} \leftarrow \text{AtribuirNo}(\text{Fim}, \text{Maior}, i_{ak}, \text{MaxNo})$ 
  9      $\text{Seq} \leftarrow \text{Union}(\text{Seq1}, i_{ak}, \text{Seq2})$ 
  10   devolva Seq
  11 senão
  12   devolva Seq

```

Algoritmo de planejamento TCEMAP2 [4]:

```

Algoritmo TCEMAP2
  // Semelhante ao algoritmo TCEMAP1, porém com alteração no trecho ( $ak$ 
  é o menor número) dentro da função AtribuirNo. Este trecho deverá ser trocado
  por:

```

1 Ordenar os nós em termos de menor latência do nó i_{ak} , que é calculado por *AtribuirNo*(), em ordem crescente e o ak é o menor número

Após conhecer as propostas detalhadas acima, um outro contexto relacionado à elas deve ser apresentado. Trata-se do conceito do problema do fluxo máximo distribuído (*Distributed Maximum Flow*), ou DMF. Os métodos apresentados neste trabalho basicamente enquadram-se nos conceitos do problema de fluxo máximo distribuído por possuírem problemas do planejamento dos agentes, baseados na latência da rede [14].

IV. ANÁLISE DE COMPLEXIDADE

Considerando uma rede com n nós, para o problema de planejamento de agentes móveis, tem-se:

O *d-Agent* tem complexidade de tempo $O(n^2 \log n)$ [3].

O TCEMAP1 tem complexidade de tempo $O(n \log n)$ e o TCEMAP2 tem complexidade de tempo de $O(n^2 \log n)$ [4].

Para algoritmos distribuídos, a ordem de complexidade das trocas de mensagem entre os nós de uma rede é o principal fator a ser analisado. Com isso, após a distribuição dos agentes, a execução dos agentes pode ser realizada. Esta execução é caracterizada por trocas de mensagens realizadas por algoritmos conhecidos de Propagação de Informação. Em geral, consideramos que $m = |E|$ onde $|E|$ é a quantidade total de arestas no grafo e $n = |V|$ onde $|V|$ é a quantidade total de vértices no grafo, logo tem-se:

$O(n^3 * m)$ para troca de mensagens e $O(n^2 * m)$ de tempo, para os casos citados acima.

V. EXPERIMENTOS

Abaixo serão apresentadas informações de resultados obtidos através do método de Propagação de Informação com *Feedback*, ou PIF, como é mais conhecido, e através do método conhecido como *Distributed Maximum Flow* ou DMF. Estes métodos foram modelados na ferramenta de algoritmos distribuídos chamada DAJ. O objetivo desta representação é apresentar e comparar os dados referentes de cada método, observando o ganho real de se equilibrar latência da rede no caso de troca de mensagens para sistemas distribuídos. Com isso, os resultados obtidos serão analisados e discutidos.

Nas imagens abaixo são apresentados gráficos de quantidade de nós x quantidade de troca de mensagens. As instâncias executadas foram de 4 a 80 nós.

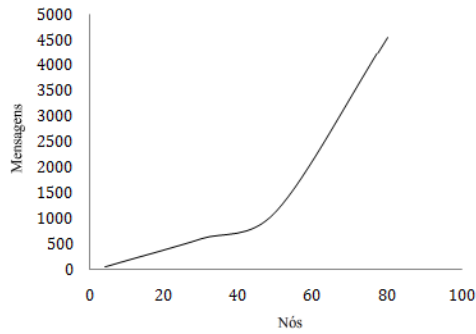


Figura 5. a) DMF

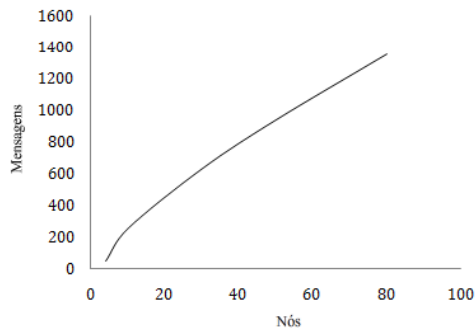


Figura 6. b) PIF

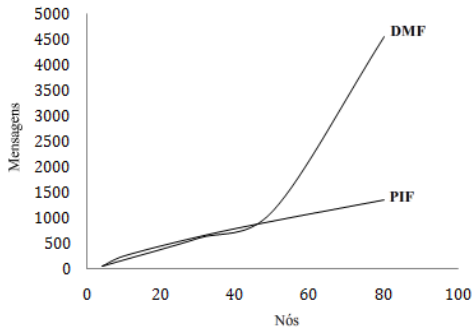


Figura 7. c) DMF X PIF

No primeiro gráfico, é apresentado o desempenho em relação a troca de mensagens do método DMF. O número de mensagens aumenta de acordo com o número de nós.

Já no segundo gráfico, é apresentado o desempenho em relação a troca de mensagens do método PIF. Neste caso, também é possível observar o número de mensagens aumentando de acordo com o número de nós.

A terceira imagem, que corresponde à junção comparativa entre os dois gráficos já apresentados, podemos perceber que, o custo de equilibrar latência da rede no caso de troca de mensagens é consideravelmente alto. Assim, um algoritmo mais simples como o PIF, no cenário apresentado, pode obter um resultado bem mais satisfatório. Pois o número de troca de mensagens é bem menor, comparado ao DMF.

A seguir é apresentado uma tabela com os resultados

do número de mensagens trocadas entre os algoritmos. Podemos perceber pela terceira coluna da tabela, que a diferença do número de troca de mensagens, aumenta consideravelmente com o aumento de nós.

Tabela I
DMF X PIF

Nós	Mensagens PIF	Mensagens DMF	Diferença
4	42	30	12
10	248	158	90
30	625	587	38
50	934	1121	187
80	1351	4563	3212

VI. TRABALHOS FUTUROS

- Aprofundar nos estudos de agentes móveis como solução real para a recuperação de informação distribuída em tempo factível;
- Estudar e propor novas formas eficientes de planejamento de agentes móveis;
- Modelar e aplicar efetivamente as técnicas estudadas para o projeto Olhos da Cidade e outros projetos que detenham desta necessidade.

VII. CONCLUSÃO

Após o estudo e análise das técnicas e métodos apresentados neste trabalho, foi possível aprofundar nos conceitos da computação distribuída e agentes móveis, conhecendo suas principais características. Com este estudo, foi possível conhecer novas técnicas que permitem e contribuem com a recuperação da informação distribuída. Tais técnicas são: *d-Agent*, TCEMAP1 e TCEMAP2. Além disso, as devidas análises de complexidade foram estudadas e apresentadas. No contexto dos algoritmos distribuídos, essas técnicas foram modeladas e implementadas para apresentação de resultados. Com isso, foi possível absorver um conhecimento considerável no assunto permitindo assim, aprofundar em novos métodos e conceitos que permitam evoluir no tema abordado. Espera-se com este trabalho, servir como contribuição para aplicações reais, como o projeto Olhos da Cidade e agregar conhecimento à interessados e principalmente aos alunos e professor da disciplina Projeto e Análise de Algoritmos do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto.

REFERÊNCIAS

- [1] V. C. Barbosa, *An Introduction to Distributed Algorithms*. The MIT Press, 1996, ISBN-13: 978-0-262-51442-2.
- [2] A. S. Tanenbaum, *Distributed operating systems*. Prentice Hall, 1995, ISBN-13: 978-0-132-19908-7.
- [3] J.-W. Baek and H.-Y. Yeom, *d-Agent: An Approach to Mobile Agent Planning for Distributed Information Retrieval*. Consumer Electronics, IEEE Transactions on, pp. 115-122, 2003.

- [4] G.-T. K. Jin-Wook Baek and H.-Y. Yeom, *Cost-Effective Planning of Timed Mobile Agents*. Information Technology: Coding and Computing, 2002. Proceedings. International Conference on, pp. 536, 2002.
- [5] C. Ghezzi and G. Vigna, *Mobile code paradigm and technologies: A case study*. in Intl Workshop on Mobile Agents, April, 1997.
- [6] Y. Aridor and M. Oshima, *Infrastructure for mobile agents: Requirements and design*. in Proc. of Intl Workshop on Mobile Agents, 1998.
- [7] W. R. Cockayne and M. Zyda, *Mobile Agents*. Manning Publications Co., 1998.
- [8] T. S. O. de Krester, A. Moffat and J. Zobel, *Methodologies for distributed information retrieval*. in Proc. of the Eighteenth International Conference on Distributed Computing Systems, pp. 26-29, 1998.
- [9] K. Moizumi, *Mobile Agent Planning Problems*. Ph.D. thesis, Dartmouth College, 1998.
- [10] K. M. W. Caripe, G. Cybenko and R. Gray, *Network awareness and mobile agent systems*. IEEE Communications Magazine, pp. 44-49, 1998.
- [11] K. Moizumi and G. Cybenko, *The traveling agent problem*. Mathematics of Control, Signals and Systems, 1998.
- [12] S. Bandyopadhyay and K. Paul, *Evaluating the performance of mobile agent-based message communication among mobile hosts in large ad hoc wireless network*. In Proc. of 2nd ACM Intl Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 1999.
- [13] K. M. D. K. G. C. B. Brewington, R. Gray and D. Rus, *Mobile agents in distributed information retrieval*. In Intelligent Information Agents, pp.355395, 1999.
- [14] M. B. Thuy Lien PHAM, Ivan LAVALLEE and S. H. DO, *A Distributed Algorithm for the Maximum Flow Problem*. In Proceedings of the 4th International Symposium on Parallel and Distributed Computing, pp. 131-138, 2005.