Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Guilherme Camilo Amantea

Análise comparativa de duas formulações de Programação Linear para o roteamento em Redes Tolerantes a Atrasos

São Paulo
2013

Guilherme Camilo Amantea

Análise comparativa de duas formulações de Programação Linear para o roteamento em Redes Tolerantes a Atrasos

Dissertação de Mestrado apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Computação.

Data da aprovação ___/___/_____

Prof. Dr. Alfredo Goldman (Orientador)
IME-USP – Instituto de Matemática e Estatística da Universidade de São Paulo

Membros da Banca Examinadora:

Prof. Dr. Alfredo Goldman (Orientador)
IME-USP – Instituto de Matemática e Estatística da Universidade de São Paulo

Prof. Dr. Célio V. N. Albuquerque (Membro)
IC-UFF – Instituto de Computação da Universidade Federal Fluminense

Prof. Dr. Paulo J. S. Silva (Membro)
IMECC-UNICAMP – Instituto de Matemática e Estatística e Computação Científica da Universidade Estadual de Campinas

Guilherme Camilo Amantea

Análise comparativa de duas formulações de Programação Linear para o roteamento em Redes Tolerantes a Atrasos

Dissertação de Mestrado apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Computação.
Área de Concentração: Engenharia de Software

Orientador: Prof. Dr. Alfredo Goldman

São Paulo
Julho/2013

Ficha Catalográfica
Elaborada pelo Departamento de Acervo e Informação Tecnológica – DAIT
do Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT

A484a **Amantea, Guilherme Camilo**
Análise comparativa de duas formulações de programação linear para o roteamento em redes tolerantes a atrasos. / Guilherme Camilo Amantea. São Paulo, 2013.
56p.

Dissertação (Mestrado em Engenharia de Computação) - Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Área de concentração: Engenharia de Software

Orientador: Prof. Dr. Alfredo Goldman Vel Lejbman

1. Programação linear 2. Geração de colunas 3. Roteamento 4. Redes tolerantes a atrasos 5. Tese I. Vel Lejbman, Alfredo Goldman, orient. II. IPT. Coordenadoria de Ensino Tecnológico III. Título

13-54 CDU 004.715(043)

RESUMO

Redes tolerantes a atrasos (*delay-tolerant networks* ou DTNs) são caracterizadas por alta latência, conectividade limitada, baixa taxa de transferência de dados, alta probabilidade de perda de pacotes e por não haver nenhuma garantia da existência de caminhos fim-a-fim. Entre os algoritmos de roteamento nessas redes que utilizam informações tais como agendamento dos contatos entre nós, ocupação dos *buffers* de armazenamento dos nós e demanda de tráfego de mensagens, encontra-se um roteamento bastante conhecido baseado em Programação Linear (PL) com objetivo de minimizar atraso médio dos pacotes. Esse algoritmo, porém, pode precisar de muito tempo para ser executado e sua formulação pode ser muito grande para exemplos simples de DTNs. Como ilustração, uma instância de rede com 70 nós, 1035 arestas e 1 mensagem sendo roteada durante um período de 1 hora e 30 minutos produz uma formulação com 11,5 milhões de restrições lineares e 11 milhões de variáveis.

Este trabalho descreve uma formulação PL equivalente à supramencionada com uma estrutura voltada à aplicação da técnica Geração de Colunas. Essa técnica permite que, em cada iteração, somente um subconjunto próprio das variáveis do problema original seja considerado. Se a solução obtida com esse subconjunto satisfizer as condições de otimalidade dadas pela teoria de dualidade em programação linear, ela é solução do problema original. Caso contrário, um novo subconjunto de variáveis é criado por meio da inclusão, no subconjunto anterior, de uma variável do problema original ignorada até então. Esse novo subconjunto é usado na iteração subsequente, o que permite que o processo se repita.

Os algoritmos baseados nas duas formulações são implementados e executados com exemplos sintéticos de redes ou geradas aleatoriamente de forma que os tamanhos das formulações e tempos de execução possam ser observados em cada caso e comparados entre si. Resultados promissores são apresentados. Um roteamento ótimo é obtido pela implementação de Geração de Colunas mais rapidamente que pela formulação original. A diferença no tempo de execução pode chegar até a três ordens de magnitude.

Palavras-chave: Redes Tolerantes a Atrasos, Roteamento, Programação Linear

ABSTRACT

Comparison of two Linear Programming formulations for routing in Delay-Tolerant Networks

Delay-tolerant networks (or DTNs) are characterized by high latency, limited connectivity, low bandwidth, high packet drop count and the inexistence of guarantees that end-to-end paths exist. Among routing algorithms in these networks that use information such as contact scheduling between nodes, node buffer occupancy and message traffic demand, there is a well-known routing based on Linear Programming (LP) with the objective of minimizing the average packet delay. This algorithm, however, may need a large amount of time to be executed and its formulation may be too big for simple DTN examples. To illustrate, a network instance with 70 nodes, 1035 edges and 1 message being routed during a 1 hour and 30 minute period produces a formulation with 11.5 million linear constraints and 11 million variables.

This work describes an LP formulation that is equivalent to the aforementioned with a structure suitable to the application of Column Generation. This technique, in each iteration, considers only a proper subset of the original problem variables. If the solution obtained with this subset satisfies the optimality conditions given by duality theory and linear programming, it is a solution to the original problem. Otherwise, a new subset of variables is created through the inclusion, in the previous subset, of a variable of the original problem that has been ignored so far. This new subset is used in the subsequent iteration, which allows the process to be repeated.

Algorithms based on the two formulations are implemented and executed with artificial network examples or examples generated randomly in such a way that the formulation sizes and execution times can be observed in each case and compared with each other. Promising results are presented. The Column Generation implementation produces an optimal routing faster than the original formulation. The difference in execution time can reach up to three orders of magnitude.

Key Words: Delay-Tolerant Networks, Routing, Linear Programming

SUMÁRIO

1 INTRODUÇÃO **5**
1.1 Motivação 5
1.2 Objetivo 7
1.3 Contribuições 7
1.4 Método de Trabalho 8
1.5 Organização do Trabalho 9

2 REVISÃO BIBLIOGRÁFICA **11**
2.1 Contexto 11
2.2 Algoritmos com Atraso Mínimo 15
2.3 Grafo Expandido 16
2.4 Autoclassificação 17

3 FORMULAÇÕES **18**
3.1 Formulação Original 18
3.2 Grafo Expandido 21
3.3 Formulação Alternativa 24
3.4 Tamanhos das Formulações 27

4 GERAÇÃO DE COLUNAS **29**
4.1 Formulação de Fluxo por Caminhos 29
4.2 Geração de Colunas 30

5 EXPERIMENTOS **33**

6 CONCLUSÃO **36**

REFERÊNCIAS **38**

A TAXONOMIA **42**
A.1 Falta de Informações (I) 42
A.2 Desperdício de Energia (II) 43
A.3 Sobrecarga de *Buffers* (III) 44
A.4 Grandes Atrasos (IV) 45
A.5 Limitações 46

B REVIEW AND ANALYSIS OF CITATIONS **47**

1 INTRODUÇÃO

1.1 Motivação

Redes subaquáticas (Heidemann et al., 2006; Akyildiz et al., 2006; Guo et al., 2010), redes rurais (Chen et al., 2009; Du et al., 2006; Mishra et al., 2005), redes veiculares (Cheng et al., 2010; Soares et al., 2009; Tatchikou et al., 2005), internet com sub-redes em diferentes planetas (Burleigh et al., 2003; Wang et al., 2011) e redes de sensores (Pathirana et al., 2005; Yun e Xia, 2010; Nayebi et al., 2010; Feng et al., 2010) são exemplos de sistemas que frequentemente apresentam alta latência, conectividade limitada, baixa taxa de transferência de dados, falta de caminhos fim-a-fim e alta probabilidade de perda de pacotes. Consequentemente, podem ser considerados instâncias de redes tolerantes a atrasos, ou *Delay-Tolerant Networks* (DTNs) (Fall, 2003).

Essas redes podem ter padrões de conectividade conhecidos, como comunicação entre satélites de órbita baixa e estações terrestres, ou conectividade imprevisível ou oportunista, como transferência de informações entre dispositivos móveis transportados por pessoas ao se aproximarem uns dos outros. Recentemente, traços de redes reais (*network traces*) têm sido estudados para a criação de algoritmos de roteamento (Ristanovic et al., 2012; Karaliopoulos e Rohner, 2012). Traços são arquivos que descrevem trajetórias de mensagens em redes reais durante um período de tempo, além dos movimentos dos nós da rede e os intervalos de tempo em que um nó se aproxima de outro.

A importância dos padrões de mobilidade dos nós e outras características de redes tolerantes a atrasos tornam os algoritmos de roteamento tradicionais inadequados por fazerem suposições que não se aplicam a DTNs, como existência de caminhos fim-a-fim. Algoritmos podem atingir melhores desempenhos em métricas como atraso médio das mensagens se fizerem suposições não-tradicionais sobre a rede. Por exemplo, realização de roteamento ao longo do tempo para atingir entrega eventual.

Jain et al. (2004) apresentam seis algoritmos de roteamento em redes tolerantes

a atrasos, especialmente naquelas com conectividade conhecida. Um desses algoritmos é baseado em uma formulação de Programação Linear (PL), o qual, entre os algoritmos apresentados, utiliza a maior quantidade de informações sobre a rede, entre as quais se encontra a previsão de todos os **contatos** (oportunidades de transmissão de dados) entre os nós, ocupação dos *buffers* em cada nó em cada momento e demanda de tráfego de mensagens. Esse algoritmo produz o roteamento das mensagens de forma a minimizar o atraso médio. Porém, sua execução é muito custosa. De acordo com os autores, dois cenários de simulação foram utilizados para comparar os desempenhos dos seis algoritmos, mas o cenário que modela 20 ônibus municipais com capacidade de comunicação baseada em rádio fazendo viagens agendadas em São Francisco, Califórnia, gerou uma formulação PL grande demais e os autores não conseguiram testá-la nesse cenário.

A emergência de traços de redes reais possibilita a aplicação de um algoritmo de otimização para a identificação de limites superiores de desempenho sobre métricas como atraso médio das mensagens. Esses limites podem servir de referência a futuros trabalhos com algoritmos de roteamento. O tamanho da formulação PL apresentada por Jain et al. (2004), porém, pode impedir sua aplicação a grande parte desses traços. Na ausência de outros algoritmos de minimização de atraso médio, isso pode implicar a inviabilidade de identificação desses limites.

Além da formulação exata, Pasztor et al. (2007), Spyropoulos et al. (2007), Burns et al. (2008) e Burgess et al. (2006) propõem algoritmos de roteamento em redes tolerantes a atrasos que não assumem conhecimento prévio sobre as alterações de conectividade do sistema e utilizam múltiplas cópias de cada mensagem com o objetivo de que uma delas atinja seu destino. Monteiro et al. (2007), Bui Xuan et al. (2002), Jones et al. (2007), Xian et al. (2011) e Dang e Wu (2010) sugerem protocolos baseados no roteamento de somente uma cópia de cada mensagem. Porém, os protocolos propostos não têm como objetivo encontrar um roteamento ótimo de um conjunto de mensagens levando em consideração a ocupação de *buffers* intermediários, demanda de tráfego de mensagens e contatos entre nós.

1.2 Objetivo

A formulação de Programação Linear para roteamento em redes tolerantes a atrasos apresentada por Jain et al. (2004) contém um número de variáveis e restrições lineares proporcional à multiplicação do número de mensagens que serão roteadas, do número de arestas do grafo que representa a rede e do número de intervalos de tempo relevantes para todo o período observado. Portanto, a formulação pode ser muito grande para redes de tamanho não trivial.

Por esse motivo, este trabalho descreve uma formulação de Programação Linear equivalente, mas com uma estrutura adequada à aplicação da técnica Geração de Colunas, a qual permite que, em cada iteração, apenas um subconjunto das variáveis do problema original seja considerado. Se a solução obtida satisfizer as condições de otimalidade, então é solução do problema original. Caso contrário, um algoritmo de precificação seleciona uma variável ignorada até então, embora não consulte todas as variáveis ignoradas, para entrar no conjunto de variáveis consideradas. Isso cria um novo conjunto, que é usado na iteração seguinte, o que permite que o processo se repita. Uma implementação dessa variante de Programação Linear é apresentada e executada com exemplos de rede gerados aleatoriamente, para que seu desempenho possa ser comparado ao da formulação de Jain et al. (2004).

1.3 Contribuições

Trabalhos anteriores considerados não oferecem investigações de melhorias de desempenho de algoritmos de minimização de atraso médio baseados em Programação Linear como os estudados por Jain et al. (2004) e Alonso e Fall (2003). Este trabalho apresenta tal investigação ao aplicar Geração de Colunas a uma formulação de Programação Linear equivalente à original e comparar os desempenhos de ambas as implementações.

1.4 Método de Trabalho

Formulação equivalente - A formulação de Programação Linear apresentada por Jain et al. (2004), chamada fluxo por arestas, contém uma restrição principal para cada vértice do grafo que representa a rede e uma variável por aresta. A primeira etapa deste trabalho é criar uma formulação equivalente, na qual há uma restrição linear por aresta (representando a soma do fluxo de dados transmitidos por ela, que deve estar abaixo da capacidade da aresta), e variáveis indexadas por elementos do conjunto de todos os caminhos possíveis desse grafo. Cada variável representa a quantidade de fluxo que passa através de cada caminho. Essa formulação equivalente, chamada fluxo por caminhos, apresenta um número exponencial de variáveis. Porém, tem uma estrutura adequada para aplicação da técnica chamada Geração de Colunas, em que subproblemas são resolvidos progressivamente, e, caso a solução de um deles satisfaça as condições de otimalidade, não é necessário usar todas elas no algoritmo de otimização.

Implementação - Após a criação da formulação aresta-caminho do problema, a segunda etapa deste trabalho é a implementação, usando a linguagem de programação C e o pacote de otimização IBM CPLEX, do algoritmo de otimização usando como base a formulação. A formulação original apresentada por Jain et al. (2004) também é implementada usando o CPLEX para que uma comparação dos desempenhos de ambas possa ser realizada.

Simulação - A terceira etapa deste trabalho consiste no desenvolvimento de *scripts* na linguagem Perl com o objetivo de produzir exemplos de redes. Esses exemplos incluem alguns sintéticos e outros aleatórios. Ou seja, representações de redes com diferentes números de vértices são produzidas por esses *scripts* de forma que os contatos entre esses vértices (i.e. oportunidades de transmissão de dados, ou períodos de tempo em que a capacidade de transmissão de dados é maior que zero) sejam criados aleatoriamente de acordo com uma distribuição uniforme.

Levantamento de dados - A quarta etapa do trabalho consiste nas execuções dos algoritmos de roteamento usando como entrada as redes produzidas na etapa

três. Os tempos de execução, tamanhos das formulações e números de iterações de cada algoritmo serão coletados para sua execução em cada rede exemplo.

Avaliação - A quinta etapa deste trabalho se resume à análise dos dados coletados para a comparação dos tempos de execução, tamanhos das formulações e números de iterações do algoritmo baseado na formulação original (Jain et al., 2004) com os daquele baseado na formulação derivada, apresentada neste trabalho.

1.5 Organização do Trabalho

A Seção 2, Revisão Bibliográfica, contém um levantamento de importantes algoritmos de roteamento em redes tolerantes a atrasos e apresenta uma avaliação das diferenças entre suas contribuições e a apresentada neste trabalho.

A Seção 3, Formulações, apresenta a formulação de Programação Linear proposta por Jain et al. (2004) para modelar o roteamento em redes tolerantes a atrasos. Além disso, apresenta a formulação derivada, do tipo aresta-caminho, e discute as diferenças e semelhanças entre os dois modelos.

A Seção 4, Geração de Colunas, mostra a representação de fluxo por caminhos da formulação de fluxo por arestas desenvolvida na Seção 3 e discute sua adequação à aplicação da técnica Geração de Colunas, a qual é também explicada.

A Seção 5, Experimentos, descreve os exemplos de redes que servem como entrada para os algoritmos implementados a partir das formulações apresentadas na Seção 3 e Seção 4. Além disso, descreve as condições computacionais do ambiente em que serão executados os algoritmos, seleciona os critérios de comparação entre os algoritmos que serão observados e justifica sua escolha. Dados obtidos após as simulações são descritos e interpretados para se obter uma comparação de desempenhos significativa.

A Seção 6, Conclusão, resume a hipótese deste trabalho, o método de pesquisa e os resultados da análise dos dados para a breve compreensão das observações viabilizadas por este trabalho.

O Apêndice A, Taxonomia, mostra uma classificação dos algoritmos de roteamento

considerados na Seção 2 de acordo com os problemas que têm o objetivo de resolver.

O Apêndice B contém um estudo sobre as citações do trabalho seminal de Jain et al. (2004) e conclui que a grande maioria delas tem como objetivo mencionar a existência de um trabalho na mesma área de pesquisa do artigo que contém a citação em vez de utilizar o conteúdo do citado. Além disso, diferentes ferramentas de busca de citações produzem resultados que não aparecem entre os resultados das outras, o que implica que nenhuma ferramenta é suficiente para se obter uma lista completa de citações de um trabalho.

2 REVISÃO BIBLIOGRÁFICA

Os protocolos de roteamento em redes tolerantes a atrasos são desenvolvidos com propósitos explícitos de resolver limitações dos protocolos pre-existentes. Todos eles, por exemplo, têm o objetivo de encaminhar mensagens com sucesso na ausência de caminhos fim-a-fim em qualquer dado momento na rede. Protocolos de roteamento em redes tradicionais falham nessas condições. Porém, os algoritmos criados para redes com essa característica também apresentam limitações importantes. Por exemplo, alguns requerem que os nós tenham informações que não são realistas para a grande maioria dessas redes, como previsão exata de todo o tráfego futuro de mensagens (Jain et al., 2004). Em outros protocolos, cada mensagem é enviada, a partir da origem, para o maior número possível de nós da rede na tentativa de atingir seu destino (Vahdat et al., 2000). Apesar disso garantir atraso mínimo e máxima taxa de entrega dos pacotes em condições ideais que ignoram colisões, contenção e limitações de *buffer*, apresenta um grande desperdício de energia (muitas transmissões desnecessárias). Nesta seção, importantes protocolos de roteamento em DTNs são apresentados e avaliados quanto a suas limitações com relação ao tipo de roteamento que é objeto de estudo deste trabalho.

2.1 Contexto

Jones e Ward (2006) classificam importantes algoritmos de roteamento em DTNs de acordo com duas propriedades: replicação e conhecimento. As estratégias que assumem disponibilidade de informações úteis para decisões de roteamento configuradas previamente nos nós (conhecimento) podem fazer uso muito eficiente dos recursos da rede como energia e espaço em *buffer* enquanto atingem alta taxa de entrega de pacotes e baixo atraso. O lado negativo é que esses esquemas não são adaptáveis a várias redes em condições diferentes. Por outro lado, redes com componentes não confiáveis e imprevisíveis inviabilizam a aplicação de algoritmos que requerem tantas informações pré-configuradas. Nesses casos, há benefício em criar múltiplas cópias de cada mensagem e espalhá-las pelos nós da rede para aumentar a probabilidade

de que uma delas encontrará seu destino (replicação).

O algoritmo *Context-Aware Routing* (Musolesi et al., 2005), classificado como conhecimento, mistura dois métodos para roteamento: síncrono e assíncrono. Se um nó-origem de uma mensagem se encontra no mesmo componente conexo da rede que o destino, o protocolo síncrono, por exemplo, DSDV (Perkins e Bhagwat, 1994), encaminha a mensagem. Caso contrário, a mensagem é encaminhada para o nó presente no mesmo componente conexo que apresenta maior probabilidade de entrega, o qual a armazena. Quando esse nó entra em contato com outro componente conexo contendo o nó destino ou um nó com maior probabilidade de entrega, encaminha a mensagem. Limitações das informações disponíveis a cada nó para o cálculo da probabilidade de entrega e o fato de que interferências entre múltiplas mensagens são ignoradas evitam que o algoritmo produza sempre roteamentos ótimos.

Com o algoritmo baseado em *erasure-coding* (Wang et al., 2005), classificado como replicação, quando configurado com fator r, cada mensagem de tamanho M é dividida em kr blocos de código, para uma constante k pré-definida. Os kr blocos de código de uma mensagem são divididos igualmente entre os primeiros kr nós encontrados pelo nó-origem da mensagem. Esses nós são denominados **portadores**. Dessa maneira, *erasure-coding* usa um número k vezes maior de portadores do que protocolos mais simples de replicação, que criam r cópias da mensagem e as distribuem entre os primeiros r contatos. Porém, cada portador transporta uma fração $1/k$ do número de *bytes* quando comparado com esses algoritmos mais simples. Dessa forma, o número de *bytes* transmitidos é o mesmo: rM. Além disso, *erasure-coding* garante que, se $1/r$ dos blocos de código forem recebidos pelo destino, a mensagem original pode ser recuperada. Um grande número de caminhos, entretanto, é percorrido por blocos de código que não atingem seu destino, embora sua transmissão desperdice energia e espaço em *buffers*. Falta de espaço em *buffer* pode levar a altas taxas de deleção de pacotes para acomodar novas mensagens, o que prejudica o desempenho.

No protocolo de conhecimento *First Contact* (Jain et al., 2004), cada mensagem é

encaminhada por uma aresta do grafo que representa a rede escolhida aleatoriamente entre todas as arestas disponíveis no momento para o nó-origem. Se todas as arestas estão momentaneamente indisponíveis, a mensagem espera até que uma delas fique com capacidade de transmissão maior que zero e é transmitida através do primeiro contato. Embora esse método não apresente bons resultados, pode ser usado como referência para comparação de desempenho entre outros protocolos.

O algoritmo baseado em *minimum estimated expected delay* (Jones et al., 2007), ou atraso esperado estimado mínimo, usa o histórico de contatos entre nós para construir o número que representa o MEED. Cada nó grava os tempos de conexões e desconexões em cada contato durante uma janela de tempo. Quando a tabela de estado de conexões é alterada, uma atualização é propagada para todos os nós da rede. Além disso, cada nó calcula suas decisões de roteamento (i.e. o próximo nó para o qual cada mensagem será transmitida) a cada novo contato, em vez de na origem ou no momento do recebimento da mensagem. Isso garante que as decisões sejam feitas com informações mais recentes. Esse método, que não leva fluxos de mensagens em consideração, atinge desempenho sub-ótimo mesmo na ausência de fluxos. Classifica-se como conhecimento por encaminhar somente uma cópia de cada mensagem. Consequentemente, pode ser adequado a ambientes com fortes restrições de recursos como energia.

No protocolo MaxProp (Burgess et al., 2006), classificado como replicação, cada nó mantém uma lista de probabilidades de encontros entre todos os nós. Essas probabilidades são utilizadas para o cálculo do custo do envio de uma mensagem para seu destino, que se trata do custo do caminho de menor custo entre todos os caminhos possíveis até o destino. O custo de um caminho se dá pela soma das probabilidades de que cada aresta do caminho não ocorra na rede. Essa informação é utilizada para duas coisas. As mensagens com menor custo são as primeiras a serem transmitidas. Aquelas com maior custo são as primeiras a serem abandonadas para liberar espaço em *buffer*. Isso ocorre com duas exceções. As mensagens mais **jovens**, ou seja, que passaram por um baixo número de nós, são transmitidas antes daquelas com menor

custo. Outra exceção é que os nós que recebem confirmações de entrega de certa mensagem abandonam essa mensagem antes das mensagens com baixa probabilidade de entrega. Como não há computação de caminhos propriamente ditos entre nó origem e destino, mensagens podem ser encaminhadas a becos sem saída ou a nós já visitados anteriormente. Além disso, a existência de múltiplas cópias, tanto de dados quanto de confirmações de entrega, degrada desempenho.

Usuários de uma rede normalmente não se movem de forma completamente aleatória. Portanto, padrões de movimentação podem ser previsíveis. Por exemplo, um lugar frequentemente visitado no passado tem alta probabilidade de ser visitado novamente no futuro. Por isso, a proposta do protocolo PROPHET (Lindgren et al., 2004), classificado como replicação, é que os nós troquem entre si, a cada encontro, uma métrica chamada previsibilidade de entrega, definida em cada nó para cada destino conhecido, indicando a chance prevista de que aquele nó entregue uma mensagem para aquele destino. Em cada encontro, os nós atualizam suas próprias previsibilidades de entrega usando como base as do vizinho. Baseada nisso, a decisão é tomada sobre encaminhar ou não uma certa mensagem para esse vizinho. O objetivo é limitar a degradação de desempenho observada em algoritmos tradicionais epidêmicos dada pelo desperdício de recursos resultante de transmissões desnecessárias.

Com o protocolo RAPID (Balasubramanian et al., 2007), classificado como replicação, o administrador da rede tem a possibilidade de selecionar uma métrica a ser otimizada. Essa métrica é traduzida para uma função de utilidade para cada pacote. Em cada oportunidade de transmissão entre os nós X e Y, os nós trocam informações de controle. Depois disso, os pacotes de X destinados a Y são transmitidos em ordem decrescente de utilidade. Para cada pacote de X não presente no *buffer* de Y, uma cópia é transferida para Y se sua função utilidade tiver uma variação suficientemente grande ao passar para Y. Ao executar o protocolo RAPID, cada nó na rede não tem acesso a informações como fluxos de mensagens ou ocupação de *buffers* de outros nós, o que limita sua capacidade de se aproximar de uma solução ótima. Apesar disso, os autores observaram desempenho consistentemente entre 90% e 100% do ótimo.

O algoritmo *Spray and Wait* (Spyropoulos et al., 2005), de replicação, é dividido em duas fases. Na fase *spray*, cada mensagem originada tem L cópias encaminhadas aos primeiros L nós da rede que entrarem em contato com o nó-origem (portadores). Na fase *wait*, se o nó destino não foi atingido na fase *spray*, cada um dos L portadores da mensagem realiza transmissão direta. Isto é, encaminham a mensagem para seu destino quando o encontrarem. Esse método pode ter um desempenho muito ruim em alguns casos, especialmente naqueles em que a mensagem tipicamente precisa passar por vários nós para chegar até seu destino.

Uma classificação desses algoritmos de acordo com os problemas de DTNs que têm o objetivo de resolver se encontra no Apêndice A.

2.2 Algoritmos com Atraso Mínimo

O protocolo baseado em roteamento epidêmico (Vahdat et al., 2000), ao distribuir cópias de cada mensagem para todos os nós atingíveis a partir do nó-origem, garante que todos os caminhos possíveis entre o nó-origem e o nó-destino sejam percorridos por alguma cópia. Em particular, o caminho ótimo, ou seja, aquele que minimiza o atraso, é percorrido, o que garante que a mensagem atinge seu destino com atraso mínimo. Contudo, esse algoritmo não leva em consideração contenção e concorrência de mensagens nos *buffers* de armazenamento dos nós e nas arestas do grafo que representa a rede. Portanto, em redes reais, que apresentam recursos limitados, o desempenho desse algoritmo pode ser severamente degradado por esses fatores.

Jain et al. (2004) descrevem um protocolo baseado em Programação Linear (PL) que também garante o roteamento de mensagens de forma a minimizar o atraso médio. Porém, não apresenta a limitação do algoritmo epidêmico de ter seu desempenho reduzido em redes com recursos limitados. Em vez disso, esse algoritmo calcula um roteamento ótimo para todas as mensagens levando em consideração as ocupações dos *buffers* a cada momento, capacidade e atraso de cada aresta e a topologia da rede em cada momento. Para que isso seja possível, portanto, o algoritmo baseado em PL precisa ter à disposição informações globais presentes e futuras, o que impede sua

aplicação a redes nas quais essas informações não estão disponíveis para os nós.

Apesar disso, aplicações do algoritmo ótimo baseado em PL incluem a criação de limites superiores de desempenho, com os quais os desempenhos de outros algoritmos podem ser comparados. De acordo com Jain et al. (2004), porém, até mesmo um cenário simples, que contém somente uma vila remota que troca mensagens com uma cidade por meio de três opções de comunicação, produziu uma formulação PL com 500.000 restrições lineares e 550.000 variáveis. A execução da implementação utilizando a ferramenta CPLEX exigiu 16.000 iterações. Por isso, instâncias de tamanhos realistas são impraticáveis.

2.3 Grafo Expandido

A formulação PL descrita por Jain et al. (2004) não se trata de um problema padrão de fluxo de custo mínimo de múltiplos produtos porque a rede é dinâmica. Isso significa que arestas podem estar disponíveis durante alguns intervalos de tempos e indisponíveis em outros. O texto clássico de fluxos em redes de Ford e Fulkerson (2010), porém, sugere uma maneira de expandir o modelo da rede original criando cópias de cada nó para cada instante de tempo. Dessa forma, cópias das arestas originais podem ser transportadas para o grafo expandido entre os nós associados a instantes de tempo em que as arestas estão disponíveis. Usando essa técnica para modelar as propriedades dinâmicas da rede, uma solução de fluxo de custo mínimo de vários produtos pode ser usada como se a rede fosse estática. Isso permite a aplicação de técnicas de fluxos em rede bem conhecidas para melhorar o desempenho.

Hay e Giaccone (2009) descrevem um grafo expandido similar para roteamento em DTNs. Embora esse grafo represente propriedades dinâmicas da rede, ele pode ser usado em um contexto estático, o que é explorado pelos autores. Eles se concentram em um modelo de problema de uma única fonte e um único destino para cada mensagem (apesar de que, como eles mencionam, o modelo pode ser usado para computar fluxos de vários produtos com a aplicação de Programação Linear). Algoritmos são apresentados os quais computam um caminho de atraso mínimo (com ou sem número

mínimo de saltos), caminho de máxima taxa de transferência de dados (com ou sem atraso mínimo), entre outros. Tais procedimentos são baseados em teoria dos grafos, como, por exemplo, o algoritmo de caminhos mínimos de Dijkstra, busca em largura e algoritmos de fluxo máximo.

Malandrino et al. (2012b) também aplicam o grafo expandido em um contexto de DTN. Eles modelam um sistema composto de veículos munidos de equipamento de comunicação e estações estacionárias conectadas à Internet como um grafo expandido. Eles então usam esse grafo em um algoritmo para encontrar roteamento ótimo de dados que são obtidos de servidores na Internet e repassados aos veículos. Tais dados incluem notícias, mapas de navegação ou arquivos multimídia. Os autores aplicam esse algoritmo baseado em Programação Linear para avaliar o desempenho da obtenção de conteúdo a partir da Internet por meio da análise do impacto de diferentes configurações do sistema, tais como a taxa de penetração de tecnologia de comunicação veicular.

2.4 Autoclassificação

Este trabalho apresenta uma variação, na forma fluxo por caminhos, da formulação de Programação Linear proposta por Jain et al. (2004). Além disso, analisa o desempenho, nos aspectos de tempo de execução e tamanho da formulação, da implementação da técnica Geração de Colunas para este problema. Como esse algoritmo produz cópia única de cada mensagem, pode ser classificado, nos termos da seção 2.1, como conhecimento.

3 FORMULAÇÕES

Nesta Seção, o modelo do problema é apresentado como descrito por Jain et al. (2004). Algumas de suas características são discutidas posteriormente, assim como a motivação para a criação da formulação alternativa, a qual também é mostrada nesta Seção.

3.1 Formulação Original

Jain et al. (2004) e Alonso e Fall (2003) propuseram uma formulação de Programação Linear que associa mensagens a arestas e instantes de tempos em um multigrafo de DTN $G(V, E)$ de forma a minimizar o atraso médio de entrega de mensagens. Por conveniência de referência, pois o modelo é o resultado da combinação de dois artigos, e para estabelecer a nomenclatura usada neste trabalho, a formulação, como descrita por Jain et al. (2004), é descrita.

Múltiplas arestas podem existir entre vértices pois múltiplas opções podem existir na rede para transferência de dados entre nós. As variáveis nesta formulação são definidas em termos do conjunto de mensagens, do conjunto de arestas e do conjunto de intervalos disjuntos de tempo que particionam o período total de observação. O conjunto considerado de intervalos de tempo $\mathcal{I}$ é determinado da maneira descrita por Alonso e Fall (2003).

Dados de entrada: Valores dos seguintes conjuntos e funções são obtidos de cada definição de rede e são considerados como entrada para o algoritmo.

- V, o conjunto de nós da rede;
- E, o conjunto de arestas da rede;
- $\mathcal{I}$, o conjunto de intervalos de tempo, obtido da maneira descrita por Alonso e Fall (2003). Cada intervalo de tempo é definido em termos de suas fronteiras (i.e. $I_q \in \mathcal{I} \implies I_q = [t_{q-1}, t_q)$);
- $c : E \times \mathbb{R}^+ \to \mathbb{R}^+$, onde $c_{e,t}$ é a capacidade da aresta $e \in E$ no instante $t \in \mathbb{R}^+$;

- $d : E \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$, onde $d_{e,t}$ é o atraso da aresta $e \in E$ no instante $t \in \mathbb{R}^+$;
- b_v é a capacidade de armazenamento em *buffer* do nó $v \in V$;
- I^v é o conjunto de arestas $(w, v) \in E$ cujo nó destino é $v \in V$;
- O^v é o conjunto de arestas $(v, w) \in E$ cujo nó origem é $v \in V$;
- K é o conjunto de mensagens;
- $w(k)$, $s(k)$, $d(k)$, $m(k)$, para $k \in K$, são, respectivamente, o instante de injeção, o nó origem, o nó destino e o tamanho da mensagem k.

Variáveis: Estas são variáveis usadas na formulação de Programação Linear.

- $N^k_{v,t}$ é a quantidade da mensagem k ocupando *buffer* no nó v no instante $t \in \mathcal{I}$;
- $X^k_{e,I}$ é a quantidade da mensagem k transmitida na fonte da aresta e durante o intervalo $I \in \mathcal{I}$;
- $R^k_{e,I}$ é a quantidade da mensagem k recebida no destino da aresta e durante o intervalo $I \in \mathcal{I}$.

Função objetivo: o objetivo é minimizar o atraso médio de entrega, o que é equivalente a minimizar a soma dos atrasos para todas as mensagens:

$$\text{Min} \sum_{v \in V} \sum_{k \in K} \sum_{I_q \in \mathcal{I}} (t_{q-1} - \omega(k))(\sum_{e \in I^v} R^k_{e,I_q} - \sum_{e \in O^v} X^k_{e,I_q}) \quad (1)$$

Restrições lineares:

$$\sum_{e \in I^v} R^k_{e,I_q} - \sum_{e \in O^v} X^k_{e,I_q} = \begin{cases} N^k_{v,t_q} - N^k_{v,t_{q-1}} + m(k) & \text{se } s(k) = v,\, \omega(k) = t_q \\ N^k_{v,t_q} - N^k_{v,t_{q-1}} & \text{caso contrário} \end{cases} \qquad k,v,I_q \qquad (2)$$

$$R^k_{e,I_q \oplus d_{e,t_{q-1}}} = X^k_{e,I_q} \qquad k,e,I_q \qquad (3)$$

$$\sum_{k \in K} N^k_{v,t_{q-1}} \leq b_v \qquad v,I_q \qquad (4)$$

$$\sum_{k \in K} X^k_{e,I_q} \leq c_{e,t_{q-1}} \cdot |I_q| \qquad e,I_q \qquad (5)$$

$$N^k_{v,t_0} = \begin{cases} m(k) & \text{se } v = s(k),\, t_0 = \omega(k) \\ 0 & \text{caso contrário} \end{cases} \qquad k,v \qquad (6)$$

$$N^k_{v,t_h} = \begin{cases} m(k) & \text{se } v = d(k) \\ 0 & \text{caso contrário} \end{cases} \qquad k,v \qquad (7)$$

Nessas restrições, $\oplus$ representa a operação *deslocamento*. I.e. $I_q \oplus s = I_p \iff [t_{p-1}, t_p) = [t_{q-1} + s, t_q + s)$. O conjunto de restrições (3) indica que cada parte da mensagem k transmitida no começo da aresta e durante o intervalo I_q deve ser recebido no final da mesma aresta após seu atraso durante esse intervalo. Para certos problemas viáveis, porém, essas restrições podem produzir uma formulação inviável. Como o atraso de uma aresta pode mudar com o tempo, é possível que haja intervalos $I_p, I_q \in \mathcal{I}$ tais que $I_p \oplus d_{e,t_{p-1}} = I_q \oplus d_{e,t_{q-1}}$. Neste caso, se $t_q > t_p$ e $d_{e,t_{q-1}} = 0$ e $d_{e,t_{p-1}} > 0$, a formulação se torna inviável, pois $R^k_{e,I_q \oplus d_{e,t_{q-1}}}$ está restrita a ter dois valores potencialmente diferentes. Por esse motivo, a implementação usada neste trabalho substitui (3) pelo seguinte sem prejudicar a intenção original:

$$R^k_{e,I_p} = \sum_{I_q \in S_{e,p}} X^k_{e,I_q} \text{ onde } \quad S_{e,p} = \{I_q \in \mathcal{I} : I_q \oplus d_{e,t_{q-1}} = I_p\} \quad k,e,I_p \qquad (8)$$

Outra modificação é realizada, a qual se baseia na formulação apresentada por Alonso e Fall (2003) (equações de número (6) no artigo deles, página 5). As equações

(2) são substituídas pelo seguinte.

$$\sum_{e \in I^v} R^k_{e,I_q} - \sum_{e \in O^v} X^k_{e,I_q} = \begin{cases} N^k_{v,t_q} - N^k_{v,t_{q-1}} - m(k) & \text{se } s(k) = v,\ \omega(k) = t_{q-1} \\ N^k_{v,t_q} - N^k_{v,t_{q-1}} + m(k) & \text{se } d(k) = v,\ t_q = t_h \\ N^k_{v,t_q} - N^k_{v,t_{q-1}} & \text{caso contrário} \end{cases} \tag{9}$$

Essas equações representam o fato de que, se v é a origem da mensagem k, a qual é gerada no instante t_{q-1}, então o **fluxo líquido** do nó v (i.e. $\sum_{e \in O^v} X^k_{e,I_q} + N^k_{v,t_q} - (\sum_{e \in I^v} R^k_{e,I_q} + N^k_{v,t_{q-1}})$) é o tamanho da mensagem k. Em outras palavras, todo o fluxo de k transmitido para fora ($\sum_{e \in O^v} X^k_{e,I_q}$) de v mais o fluxo que permanece armazenado (N^k_{v,t_q}) em v menos o que é recebido ($\sum_{e \in I^v} R^k_{e,I_q}$) por v menos o que estava previamente armazenado ($N^k_{v,t_{q-1}}$) em v é o tamanho da mensagem k. Isso significa que $m(k)$ unidades de fluxo são introduzidas no sistema. Se v é o destino de k, então o fluxo líquido de v para a mensagem k deve ser o tamanho negativo de k no instante t_h (o fim da simulação), o que significa que a mensagem é consumida por v (removida do sistema). Se v não é destino ou origam de k, então seu fluxo líquido para k deve ser zero.

A criação de um intervalo de tempo em $\mathcal{I}$ tal que, $\forall k \in K$, $w(k) > t_0$, torna as equações (6) desnecessárias. As equações (7) também podem ser ignoradas após a introdução das novas equações (9).

3.2 Grafo Expandido

A principal diferença entre a formulação da Seção 3.1 e a do problema tradicional de fluxo de custo mínimo de vários produtos é a dependência dos intervalos de tempo. O livro clássico de fluxos em redes de Ford e Fulkerson (2010) descreve uma maneira de modelar o fator tempo sem a criação de tipos diferentes de variáveis além daquelas que representam a quantidade de fluxo transmitida em cada aresta do grafo da rede. Em vez disso, se $T = |\mathcal{I}|$, o grafo da rede original $G = G(V, E)$ é expandido e produz $G(T) = G(V(T), E(T))$ de tal forma que cada vértice e aresta tem uma cópia para cada instante de tempo. Desta forma, o problema dinâmico é reduzido a fluxo tradicional

(estático) de custo mínimo de vários produtos no grafo expandido $G(T)$.

Considere o exemplo da Figura 1. Cada aresta neste exemplo é anotada com sua capacidade c e atraso d no formato (c, d). A aresta $(0, 1)$, por exemplo, tem capacidade 2 e atraso 1. Suponha que essa rede é observada durante um período de tempo composto de quatro intervalos: $\{[0, 1), [1, 2), [2, 3), [3, 4)\}$. Deve ser representado o fato de que fluxo transmitido no começo de uma aresta nesta rede chega em seu destino após o atraso da aresta. Também deve ser representado fluxo armazenado em nós por algum período de tempo até que uma boa oportunidade de transmissão aconteça.

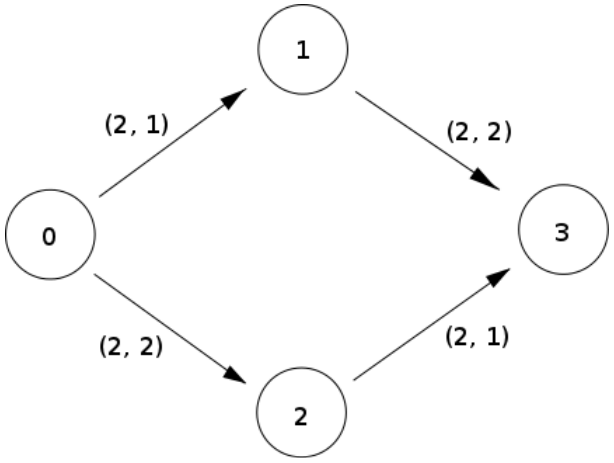

Figura 1: Exemplo de rede

A rede expandida para o exemplo da Figura 1 é mostrado na Figura 2. Cada um dos quatro nós tem quatro cópias, uma para cada instante de tempo. Cada aresta também tem cópias para instantes de tempo. A aresta $(0, 2)$, por exemplo, tem duas cópias: uma que conecta o nó 0 no instante 0 ao nó 2 no instante 2 (pois seu atraso é 2) e outra que conecta o nó 0 no instante 1 ao nó 2 no instante 3. Uma aresta da rede original tem uma cópia para cada instante de tempo exceto aqueles que implicariam fluxo chegando a seu destino após o último instante de tempo. Por exemplo, fluxo só pode ser transmitido a partir do nó 0 para o nó 2 nos instantes 0 ou 1 através da aresta $(0, 2)$. Se fosse transmitido no instante 2, chegaria a seu destino após um atraso de 2, o que significa instante 4, o qual está fora do período de observação.

Outra característica importante desse grafo expandido são as arestas entre pares de nós com o mesmo rótulo. Cada cópia do nó 0 se conecta com outra de suas

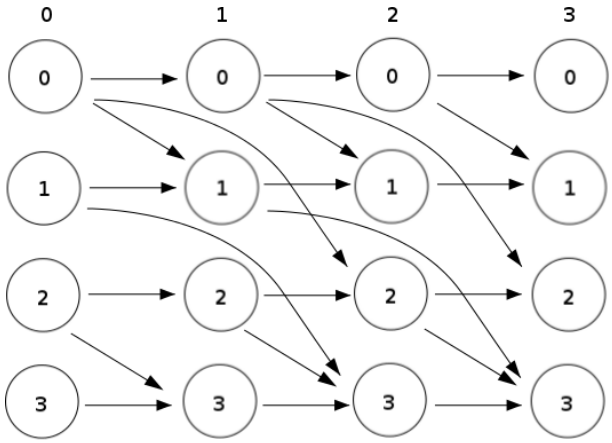

Figura 2: Exemplo de rede expandida

cópias associada ao próximo instante de tempo. Essas arestas de *buffer* representam o armazenamento de um nó durante um intervalo de tempo. Se fluxo passa por uma aresta que conecta o nó 0 no instante 0 com o nó 0 no instante 1, isso significa que esse fluxo foi armazenado no nó 0 durante o intervalo $[0, 1)$.

A solução de um problema tradicional (estático) de fluxo de múltiplos produtos no grafo expandido pode ser interpretada em termos das propriedades dinâmicas da rede original.

O grafo expandido $G(T)$ é construído da seguinte maneira. Suponha que $I_{\mathcal{I}}$ é o conjunto de instantes de tempo formado pelos pontos extremos dos intervalos em $\mathcal{I}$. Além disso, para $p = 0, 1, ..., T$, t_p é o valor do $(p + 1)$-ésimo instante de tempo em $I_{\mathcal{I}}$, $i(t) \in \{0, 1, ..., T\}$ é o índice do instante $t \in I_{\mathcal{I}}$, $d_{e,p} := d_{e,t_p}$ e $c_{e,p} := c_{e,t_p}$.

Cada vértice $v \in V$ tem uma cópia $v(p) \in V(T)$ para $p \in \{0, 1, ..., T\}$. Cada aresta $e = (v, w) \in E$ tem uma cópia $e(p) = (v(p), w(i(t_p+d_{e,p}))) \in E(T)$ para $p \in \{0, 1, ...i(t_T - d_{e,p})\}$. Além disso, para cada vértice $v \in V$, aresta de *buffer* $(v(p), v(p + 1)) \in E(T)$ são criadas para $p \in \{0, 1, ..., T - 1\}$. Quando fluxo passa por uma dessas arestas de *buffer*, significa que dados são armazenados em v por um passo de tempo. A capacidade da aresta $e(p) = (v(p), w(i(t_p+d_{e,p}))) \in E(T)$ é definida como $c(e(p)) = c_{e,p}$. Similarmente, o atraso da aresta é $d(e(p)) = d_{e,p}$. A capacidade da aresta de *buffer* $(v(p), v(p + 1)) \in E(T)$ é b_v, que representa a capacidade de armazenamento do nó $v \in V$. O atraso dessa aresta é 0.

3.3 Formulação Alternativa

Com um grafo expandido como descrito na Seção 3.2, podemos ver que a formulação seguinte é equivalente àquela descrita na Seção 3.1.

$$\text{Min} \sum_{k \in K} dx^k$$

$$\sum_{k \in K} x_e^k \leq c(e) \qquad \forall e \in E(T) \tag{10}$$

$$\mathcal{N} x^k = a^k \qquad \forall k \in K \tag{11}$$

$$0 \leq x_e^k \leq c(e) \qquad \forall e \in E(T), \forall k \in K$$

Aqui, x^k é um vetor de dimensão $|E(T)|$ tal que x_e^k é a quantidade da mensagem $k \in K$ transmitida através da aresta $e \in E(T)$. d também é um vetor de dimensão $|E(T)|$ tal que d_e é o atraso da aresta $e \in E(T)$. $\mathcal{N}$ é a matriz de incidência de $G(T)$, tal que sua dimensão é $|V(T)| \times |E(T)|$ e:

$$\mathcal{N}_{v,e} = \begin{cases} +1 & \text{se } e = (v, w) \text{ para algum } w \in V(T) \\ -1 & \text{se } e = (w, v) \text{ para algum } w \in V(T) \\ 0 & \text{caso contrário} \end{cases}$$

O vetor a^k tem dimensão $|V(T)|$ e é definido da seguinte maneira.

$$a^k_{v(t_q)} = \begin{cases} m(k) & \text{se } v = s(k) \text{ e } t_q = \omega(k) \\ -m(k) & \text{se } v = d(k) \text{ e } t_q = t_h \\ 0 & \text{caso contrário} \end{cases} \tag{12}$$

Por simplicidade, defina, para cada $t \in I_{\mathcal{I}}$ e cada $e = (v, w) \in E$ do grafo original G, $e(t) := e(i(t)) \in E(T)$ e $v(t) := v(i(t)) \in V(T)$. Dessa forma, $e(t) = (v(t), w(t + d_{e,t}))$. Para ver que esta formulação é equivalente à descrita na Seção 3.1, interpretamos os

valores de x^k como as seguintes igualdades.

$$X^k_{e,I_q} = x^k_{e(t_{q-1})} \qquad \forall I_q \in \mathcal{I} \qquad (13)$$

$$R^k_{e,I_q} = \sum_{t_p \in S_{e,q-1}} x^k_{e(t_p)} \text{ onde } S_{e,q} = \{t_p \in I_{\mathcal{I}} : t_p + d_{e,t_p} = t_q\} \qquad \forall I_q \in \mathcal{I} \qquad (14)$$

$$N^k_{v,t_q} = x^k_{(v(t_{q-1}),v(t_q))} \qquad \forall q \in \{1, ..., T\} \qquad (15)$$

A primeira igualdade decorre diretamente da definição das variáveis X^k_{e,I_q} e x^k. A segunda igualdade decorre da primeira igualdade e do conjunto de restrições (9). A última igualdade decorre da definição das arestas de *buffer* $(v(p), v(p+1))$ em $G(T)$. Temos, também, para $i = v(t_{q-1})$:

$$\sum_{\{j:(j,i)\in E(T)\}} x^k_{(j,i)} = \sum_{e \in I^v} R^k_{e,I_q} + N^k_{v,t_{q-1}} \qquad (16)$$

$$\sum_{\{j:(i,j)\in E(T)\}} x^k_{(i,j)} = \sum_{e \in O^v} X^k_{e,I_q} + N^k_{v,t_q} \qquad (17)$$

O lado direito da equação (16) representa a soma dos fluxos de todas as arestas que chegam no nó $v \in V$ do grafo original no instante t_{q-1} mais o fluxo que permanece armazenado em v durante $[t_{q-2}, t_{q-1})$. Isso é igual à soma dos fluxos que chegam em $v(t_{q-1})$ no grafo expandido. Analogamente, o lado direito da equação (17) representa a soma dos fluxos de todas as arestas que saem do nó v no instante t_{q-1} mais o fluxo que permanece armazenado em v durante $[t_{q-1}, t_q)$, que é igual à soma dos fluxos no grafo expandido em arestas que saem de $v(t_{q-1})$. Daí, temos, para $i = v(t_{q-1})$:

$$\begin{aligned}
-a^k_i &\overset{(11)}{=} \sum_{\{j:(j,i)\in E(T)\}} x^k_{(j,i)} - \sum_{\{j:(i,j)\in E(T)\}} x^k_{(i,j)} \\
&\overset{(16),\,(17)}{=} \sum_{e \in I^v} R^k_{e,I_q} - \sum_{e \in O^v} X^k_{e,I_q} + N^k_{v,t_{q-1}} - N^k_{v,t_q} \\
&\overset{(12)}{=} \begin{cases} -m(k) & \text{se } v = s(k) \text{ e } t_{q-1} = \omega(k) \\ m(k) & \text{se } v = d(k) \text{ e } t_q = t_h \\ 0 & \text{caso contrário} \end{cases}
\end{aligned}$$

Isso mostra que as equações (11) são equivalentes às equações (9). O efeito das equações (8) se torna presente pela forma que arestas são construídas no grafo expandido ($e(p) = (v(p), w(t_p + d_{e,p})) \in E(T)$ para $p \in \{0, ..., i(t_T - d_{e,p})\}$ e $e = (v, w) \in E$). Isso implica que o fluxo transmitido no começo de uma aresta chega ao destino da aresta após $d_{e,p}$ unidades de tempo. As equações (10) e a construção das capacidades de arestas de *buffer* se encarregam das equações (4) e (5) com a observação de que, por construção, todos os intervalos $I_q \in \mathcal{I}$ têm o mesmo tamanho, o que significa que as capacidades podem ser redefinidas para tornar desnecessário o fator $|I_q|$ de (5).

Na formulação original, cada variável R^k_{e,I_q} tem coeficiente de custo $t_{q-1} - \omega(k)$. Cada X^k_{e,I_q} tem coeficiente $-(t_{q-1} - \omega(k))$. Para determinar os coeficientes de custo da nova formulação, observamos, das equações (13) e (14), que cada variável $x^k_{e(t_{q-1})}$ associada com uma aresta não-*buffer* $e \in E(T)$ na nova formulação deve aparecer na função objetivo com dois coeficientes:

$$(t_{q-1} - \omega(k))X^k_{e,I_q} \overset{(13)}{=} (t_{q-1} - \omega(k))x^k_{e(t_{q-1})}$$
$$(t_{p-1} - \omega(k))R^k_{e,I_p} \overset{(14)}{=} \sum_{t_{q-1} \in S_{e,p-1}} (t_{p-1} - \omega(k))x^k_{e(t_{q-1})}$$
$$\text{onde } S_{e,p} = \{t_q \in I_{\mathcal{I}} : t_q + d_{e,t_q} = t_p\}$$

Como $t_{p-1} = t_{q-1} + d_{e,t_{q-1}}$ acima, o coeficiente de custo de $x^k_{e(t_{q-1})}$ para $e \in E$ deve ser a diferença

$$(t_{p-1} - \omega(k)) - (t_{q-1} - \omega(k)) =$$
$$(t_{q-1} + d_{e,t_{q-1}} - \omega(k)) - (t_{q-1} - \omega(k)) =$$
$$t_{q-1} - t_{q-1} + d_{e,t_{q-1}} - \omega(k) + \omega(k) =$$
$$d_{e,t_{q-1}}$$

Sabemos que $d_{e,t_{q-1}} = d(e(t_{q-1}))$ da definição de atrasos de arestas no grafo expandido. Além disso, tanto as variáveis N^k_{v,t_q} na formulação original quanto os atrasos de

arestas de *buffer* no grafo expandido são iguais a 0. Consequentemente, os atrasos das arestas no grafo expandido são coeficientes de custo adequados na nova formulação.

3.4 Tamanhos das Formulações

Podemos comparar os tamanhos das duas formulações em termos do número de variáveis e do número de restrições lineares. Como a formulação da Seção 3.2 tem uma variável para cada mensagem e para cada aresta no grafo expandido, precisamos determinar como esse número se compara com o número de variáveis da formulação original.

Há dois tipos de arestas $e \in E(T)$. Aquelas criadas a partir de arestas do grafo original (não expandido) e arestas de *buffer*. Da construção daquele tipo de arestas, há no máximo $|E| \cdot (|\mathcal{I}| + 1)$ delas. Especificamente, elas são $(v(t_p), w(t_p + d_{e,p}))$, $\forall p \in \{0, ..., T\}$, $\forall e = (v, w) \in E$. Da construção de arestas de *buffer*, há $|V| \cdot |\mathcal{I}|$ delas, que são $(v(p), v(p+1))$, $\forall p \in \{0, ..., T-1\}$, $\forall v \in V$. No total, $|E(T)| \leq |E|+|\mathcal{I}|\cdot(|E|+|V|)$. Consequentemente, a formulação alternativa tem $|K| \cdot |E(T)| \leq |K| \cdot |E| + |K| \cdot |\mathcal{I}| \cdot (|E| + |V|)$ variáveis. A formulação original tem três tipos de variáveis: $|K| \cdot |E| \cdot |\mathcal{I}|$ variáveis R^k_{e,I_q}, $|K| \cdot |E| \cdot |\mathcal{I}|$ variáveis do tipo X^k_{e,I_q} e $|K| \cdot |V| \cdot (|\mathcal{I}| + 1)$ variáveis do tipo N^k_{v,t_q}. A soma disso é $2 \cdot |K| \cdot |E| \cdot |\mathcal{I}| + |K| \cdot |V| \cdot (|\mathcal{I}| + 1)$. A diferença entre o número de variáveis nas formulações alternativa e original é:

$$
\begin{aligned}
&|K||E| + |K||\mathcal{I}||E| + |K||\mathcal{I}||V| \\
&- (|K||V| + 2|K||\mathcal{I}||E| + |K||\mathcal{I}||V|) \\
&= |K||E| - |K||V| - |K||\mathcal{I}||E| \leq 0
\end{aligned}
$$

O número de restrições na formulação original é:

$$
\begin{aligned}
&|K| \cdot |V| \cdot |\mathcal{I}| && \text{para as restrições (2)} \\
&\quad + |K| \cdot |E| \cdot |\mathcal{I}| && \text{para as restrições (3)} \\
&\quad + |V| \cdot |\mathcal{I}| && \text{para as restrições (4)} \\
&\quad + |E| \cdot |\mathcal{I}| && \text{para as restrições (5)} \\
&\quad + |K| \cdot |V| && \text{para as restrições (6)} \\
&\quad + |K| \cdot |V| && \text{para as restrições (7)} \\
&\quad = |\mathcal{I}| \cdot (|K| + 1) \cdot (|V| + |E|) + 2 \cdot |V| \cdot |K|
\end{aligned}
$$

A diferença entre $|K| \cdot |V(T)| + |E(T)| = |K| \cdot |V| \cdot (|\mathcal{I}| + 1) + |E| + |\mathcal{I}| \cdot (|E| + |V|)$, que é o número de restrições na formulação alternativa, e o número de restrições na formulação original é:

$$
\begin{aligned}
&|K||V|(|\mathcal{I}| + 1) + |E| + |\mathcal{I}|(|E| + |V|) \\
&- |\mathcal{I}|(|K| + 1)(|V| + |E|) + 2|V||K| \\
&= |E| - |K||\mathcal{I}||E| - |K||V| \leq 0
\end{aligned}
$$

A conclusão é que a formulação alternativa não é maior que a original. Além disso, como $\mathcal{I}$ pode ser exponencialmente grande, a formulação alternativa pode ser significativamente menor que a original, embora ambas tenham tamanhos da mesma ordem de magnitude.

A formulação alternativa possui somente um tipo de variável, que representa fluxo que passa por cada arco do grafo expandido (daí o termo fluxo por arcos). Descrita dessa maneira, pode ser convertida para uma representação de fluxo por caminhos, a qual viabiliza a aplicação da técnica Geração de Colunas, que visa à diminuição do tempo necessário para a produção de um roteamento ótimo. Essa técnica é discutida na Seção 4.

4 GERAÇÃO DE COLUNAS

Nesta Seção, a representação de fluxo por caminhos equivalente à formulação de fluxo por arcos da Seção 3.3 é apresentada. Sua adequação à aplicação da técnica Geração de Colunas é discutida e a técnica em si é brevemente mostrada para facilidade de referência.

4.1 Formulação de Fluxo por Caminhos

A formulação de fluxo por arestas descrita na Seção 3.3 pode ser representada como uma formulação de fluxo por caminhos (um fato bem conhecido da literatura).

$$\text{Min} \sum_{k \in K} \sum_{P \in \mathbb{P}^k} d(P) f(P)$$

$$\sum_{k \in K} \sum_{P \in \mathbb{P}^k} \delta_e(P) f(P) \leq c(e) \qquad \forall e \in E(T) \tag{18}$$

$$\sum_{P \in \mathbb{P}^k} f(P) = m(k) \qquad \forall k \in K \tag{19}$$

$$f(P) \geq 0 \qquad \forall k \in K, P \in \mathbb{P}^k$$

Nessa formulação, $\mathbb{P}^k$ é o conjunto de todos os caminhos entre o nó origem e o destino da mensagem $k \in K$. Se $v \in V$ é a origem da mensagem k no grafo original G, $w \in V$ é seu destino e $t \in I_{\mathcal{I}}$ é seu tempo de injeção, dizemos que $v(t) \in V(T)$ é sua origem no grafo expandido e $w(t_h) \in V(T)$ é seu destino. $f(P)$ é a variável nessa formulação e representa a quantidade de fluxo que passa pelo caminho $P \in \mathbb{P}^k$ para cada $k \in K$. $d(P)$ representa o atraso do caminho P e é igual a $\sum_{e \in P} de$. $\delta_e(P)$ é um indicador cujo valor é 1 se $e \in P$ e 0 caso contrário.

Essa formulação de fluxo por caminhos tem um número menor de restrições lineares que a de fluxo por arestas. Especificamente, a formulação de fluxo por arestas tem $|E(T)| + |V(T)| \cdot |K|$ restrições enquanto a de fluxo por caminhos tem somente $|E(T)| + |K|$. Por outro lado, o número de variáveis é muito maior na formulação de

fluxos por caminhos, pois há uma para cada caminho possível entre o par de nós associado com cada mensagem. Porém, somente um pequeno número dessas variáveis tem fluxo positivo atribuído na solução ótima (Ahuja et al., 1993). Esse fato pode ser explorado pela técnica **geração de colunas**, a qual nos permite nunca listar explicitamente todas as variáveis.

4.2 Geração de Colunas

O método simplex de programação linear mantém, em cada iteração, uma **base**, que se trata de um subconjunto das variáveis do problema original. A cada restrição linear da formulação sendo processada, é calculado um valor chamado **multiplicador simplex** (relacionado a uma variável do problema linear dual). Como a formulação de fluxo por caminhos apresentada na Seção 4.1 possui uma restrição de capacidade para cada aresta e uma restrição de demanda para cada mensagem, existe um multiplicador simplex w_e para cada $e \in E(T)$ e um multiplicador σ^k para cada $k \in K$.

Os multiplicadores simplex são utilizados para o cálculo do **custo reduzido** de cada variável do problema. No nosso caso, o custo reduzido da variável $f(P)$ é

$$\overline{d}(P) = d(P) - \sum_{e \in P} w_e - \sigma^k \tag{20}$$

O propósito dos multiplicadores simplex é rearranjar o problema de forma que ele fique na forma canônica. Isso significa que os custos reduzidos das variáveis presentes na base devem ser iguais a zero. Em outras palavras, os multiplicadores simplex são w_e e σ^k tais que satisfaçam as equações $\overline{d}(P) = d(P) - \sum_{e \in P} w_e - \sigma^k = 0$ para cada variável $f(P)$ presente na base em uma dada iteração. Como o número de variáveis presentes na base é igual ao número de restrições lineares, o qual é igual ao número de multiplicadores simplex, essas equações produzem uma resposta única. Além disso, não é necessário consultar informações associadas a variáveis fora da base para o cálculo dos multiplicadores.

Na forma canônica, a função objetivo do problema em uma iteração associada com

a base B é

$$\text{Min} \sum_{P \notin B} \overline{d}(P) f(P) + z_0$$

Onde $\overline{d}(P)$ é o custo reduzido da variável não-básica $f(P)$ e z_0 é o termo constante resultante das operações elementares entre as restrições lineares e a função objetivo originais para produzir uma função objetivo cujos coeficientes das variáveis básicas são zero (e os das não-básicas são seus custos reduzidos).

No método simplex, o valor de toda variável não-básica é sempre zero. Por isso, se o custo reduzido $\overline{d}(P)$ de alguma variável não-básica $f(P) \notin B$ for negativo, a solução é sub-ótima, pois, se aumentássemos o valor dessa variável, o valor da função objetivo diminuiria. Para encontrar uma solução melhor, uma variável não-básica com custo reduzido negativo é selecionada para entrar na base no lugar de alguma variável básica. Dessa forma a solução obtida com a nova base tem um valor da função objetivo menor do que o da solução associada com a base anterior. Por esse motivo, uma das condições de otimalidade do simplex é, para cada variável $f(P)$ fora da base:

$$\overline{d}(P) = d(P) - \sum_{e \in P} w_e - \sigma^k \geq 0$$

Sabemos que o atraso $d(P)$ do caminho $P \in \mathbb{P}^k$ é igual à soma dos atrasos das arestas que o compõem. Em outras palavras, $d(P) = \sum_{e \in P} d_e$. Por isso, podemos reescrever a inequação acima como:

$$\overline{d}(P) = \sum_{e \in P} (d_e - w_e) - \sigma^k \geq 0 \tag{21}$$

Interpretamos $\sum_{e \in P}(d_e - w_e)$ como o custo do caminho P em termos dos custos modificados $d_e - w_e$ de cada uma de suas arestas $e \in P$. Podemos então verificar se a condição (21) está satisfeita para todos os caminhos fora da base em duas etapas. A primeira se resume à identificação, entre todos os caminhos $P \in \mathbb{P}^k$, daquele com custo modificado mínimo. A segunda etapa consiste em determinar se o custo modificado do caminho identificado é maior ou igual a σ_k. Se for, o algoritmo pode parar

pois encontrou uma solução ótima. Se não, a variável associada a esse caminho pode entrar na base, o que diminui o valor da função objetivo. Nesse caso, uma nova iteração acontece com a nova base, que permite o cálculo dos novos multiplicadores para encontrar os novos custos reduzidos e assim por diante.

Para realizar a primeira etapa, sem a técnica Geração de Colunas, todas as variáveis não-básicas precisam ser consultadas individualmente para determinar se alguma delas tem custo reduzido negativo. Isso seria inviável neste caso, em que o número de variáveis é exponencial. Em vez disso, ao explorar a estrutura da formulação de fluxo por caminhos, podemos identificar um caminho com custo modificado mínimo com o algoritmo de Dijkstra no grafo expandido uma vez para cada mensagem $k \in K$ considerando $d_e - w_e$ como custos das arestas. Esse caminho pode ser usado então na segunda etapa.

O tempo de execução da técnica de geração de colunas depende do número de colunas geradas, o qual é diferente para cada conjunto de valores de entrada. Como nem todas as colunas são listadas, porém, a redução no número de restrições em comparação com a formulação original tem um impacto dramático no tempo de execução.

Alguns exemplos de rede com mensagens roteadas são apresentados na Seção 5 e os desempenhos da implementação de Geração de Colunas e da formulação original de Jain et al. (2004) são comparados, tanto em termos de tempo de execução quanto de número de variáveis e de restrições lineares.

5 EXPERIMENTOS

Uma implementação do programa linear original apresentado por Jain et al. (2004) com as modificações da Seção 3.1 foi criada e usada pra resolver três instâncias de problemas simples de roteamento. As mesmas três instâncias foram também usadas como entrada em uma implementação da técnica Geração de Colunas com o objetivo de comparar tempos de execução e tamanhos das formulações em termos dos números de variáveis e restrições lineares. Ambas as implementações são baseadas na biblioteca programática CPLEX da IBM.

A primeira instância de rede é inspirada pelo exemplo de vila remota descrito por Jain et al. (2004). Nesse exemplo, uma cidade com acesso constante à Internet se comunica com uma vila rural remota por meio de três canais diferentes. Um satélite de baixa órbita fica dentro do alcance de transmissão da cidade e da vila algumas vezes por dia e oferece conexão entre elas. Um entregador de motocicleta visita a vila remota com um dispositivo de armazenamento e coleta e entrega dados de rede na vila durante sua visita. Então ele viaja para a cidade e encaminha os dados e coleta respostas. Essas viagens também são feitas algumas vezes por dia. Outra opção de roteamento é uma conexão discada, a qual fica disponível durante um período por dia.

Dado que alguns atrasos nessas opções de conexão são da ordem de segundos, o número de intervalos de tempo necessários para um período de 48 horas de observação é grande. Nesta instância, o passo de discretização do tempo descrito por Alonso e Fall (2003) resulta em 187.802 intervalos de tempo. Considerando somente duas mensagens sendo roteadas, o tamanho da formulação PL e tempo de execução em um computador com 24 processadores Intel®Xeon®com 2.4GHz e 128GB de RAM se encontram na coluna **Cenário 1** da Tabela 1.

A mesma instância de rede foi usada como entrada para a implementação de Geração de Colunas. Como esperado, o número de intervalos de tempo criado foi o mesmo, pois ambas as implementações compartilham o passo de discretização. O tamanho da formulação de fluxo por caminhos com duas mensagens e seu tempo de execução no mesmo computador são encontrados na coluna **Cenário 1** da Tabela 2.

Tabela 1: Tamanhos e tempos de execução de instâncias da formulação original que representam cada uma das três redes.

	Cenário 1	Cenário 2	Cenário 3
Restrições	6 M	7,8 M	11,5 M
Variáveis	5,63 M	8 M	11 M
Tempo de Execução	24m4s	30s	110m

Tabela 2: Tamanhos das formulações de fluxo por caminhos que representam cada uma das três redes e os tempos de execução na Geração de Colunas.

	Cenário 1	Cenário 2	Cenário 3
Restrições	2,2 M	1,4 M	5,7 M
Caminhos Gerados	2	5	1
Tempos de Execução	4,4s	3,6s	32s

Outro exemplo simples de rede usado como entrada nos experimentos é uma versão adaptada da instância emprestada de Ahuja et al. (1993). Trata-se de uma rede com 6 nós, 7 arestas e 2 mensagens. Uma rede tão simples quanto essa foi executada quase instantaneamente em ambas as implementações de Programação Linear quando um período pequeno de observação de 40 unidades de tempo foi considerado. Um período maior, porém, de 20.000 unidades de tempo, causou diferenças significativas entre as implementações. As colunas **Cenário 2** das Tabelas 1 e 2 mostram os tamanhos das formulações e os tempos de execução da aplicação dessa rede como entrada para o programa linear original e a Geração de Colunas, respectivamente, usando o período de 20.000 unidades de tempo.

Um exemplo artificial de rede foi criado com 70 nós. Entre cada par de nós, uma aresta direcionada foi criada com probabilidade de 50%, o que resultou em 1035 arestas. Cada aresta ficou ativa entre 1 e 5 segundos. O programa linear original precisou de 1,5 milhões de iterações para produzir um roteamento ótimo de somente uma mensagem. O tamanho dessa formulação pode ser encontrado na coluna **Cenário 3** da Tabela 1. A coluna de mesmo nome na Tabela 2 mostra o tamanho da formulação de fluxo por caminhos dessa rede e o tempo de execução da Geração de Colunas.

Como pode ser visto nas Tabelas 1 e 2, as formulações de Geração de Colunas têm um número bem menor de restrições lineares que a formulação original. Além disso, o tempo de execução da implementação de Geração de Colunas pode ser mais

rápido com uma diferença de até três ordens de magnitude em alguns casos.

6 CONCLUSÃO

Uma formulação de Programação Linear do tipo fluxo por arestas que minimiza o atraso médio das mensagens foi proposta para o roteamento em redes tolerantes a atrasos, a qual é baseada no programa linear apresentado por Jain et al. (2004). O uso de um grafo expandido permite a aplicação de teoria padrão de fluxos estáticos de custo mínimo de vários produtos. Os resultados de tal aplicação podem ser interpretada em termos das propriedades dinâmicas da rede usada para a construção do grafo. Os tamanhos da formulação original e da de fluxo por arestas são comparados e a conclusão é que a segunda não é maior do que a primeira em termos do número de restrições lineares e de variáveis.

A técnica Geração de Colunas foi aplicada na representação de fluxo por caminhos da formulação de fluxo por arestas. A vantagem da representação de fluxo por caminhos é um número muito menor de restrições lineares. Por outro lado, tem um número exponencial de variáveis, cada uma das quais representa um dos possíveis caminhos na rede entre o nó origem e o nó destino de cada mensagem. Contudo, Geração de Colunas permite a resolução do problema em um conjunto restrito dessas variáveis. Como todas as variáveis nunca são explicitamente listadas pelo algoritmo, uma solução ótima é tipicamente encontrada muito mais rapidamente do que com o uso da formulação original de fluxo por arestas.

Simulações em instâncias de redes mostram que a diferença entre os tempos de execução da implementação de geração de colunas e da implementação da formulação original apresentada por Jain et al. (2004) pode atingir até três ordens de magnitude usando a biblioteca programática CPLEX da IBM. Observa-se, também, que a formulação de fluxo por caminhos usada pela Geração de Colunas produz um número muito menor de restrições lineares que a formulação original. Soluções exatas cuja identificação era inviável em algumas instâncias usando formulações anteriores são agora viáveis.

Como pesquisa futura, a aplicação do algoritmo baseado em Grafos Evolutivos (Ferreira et al., 2010) pode ser comparado com o roteamento encontrado pelo método

baseado em Programação Linear apresentado neste trabalho. Apesar de ter execução bastante rápida, aquele algoritmo não produz roteamento ótimo na presença de fluxos. Essa comparação pode ocorrer com base em traços de mobilidade obtidos de situações reais, como movimentos de pessoas em parques municipais. Além disso, traços de mobilidade veicular relacionados aos trabalhos de Malandrino et al. (2013) e Malandrino et al. (2012a) também podem ser utilizados para validação do método baseado em Geração de Colunas.

Como produtos deste trabalho de mestrado, os artigos Amantea e Goldman (2011) e Amantea et al. (2013) foram aceitos para publicação em conferências.

REFERÊNCIAS

AHUJA, R. K.; MAGNANTI, T. L.; ORLIN, J. B. Network flows: theory, algorithms, and applications. 1993.

AKYILDIZ, I. F.; POMPILI, D.; MELODIA, T. State-of-the-art in protocol research for underwater acoustic sensor networks. In: **Proceedings of the 1st ACM international workshop on Underwater networks**, WUWNet '06, p. 7–16. ACM, New York, NY, USA, 2006.

ALONSO, J.; FALL, K. A linear programming formulation of flows over time with piecewise constant capacity and transit times. **IRB-TR-03**, volume 7, 2003.

AMANTEA, G.; GOLDMAN, A. Review and Analysis of Citations. In: **Proceedings of 8th Experimental Software Engineering Latin American Workshop**, p. 9. 2011.

AMANTEA, G.; RIVANO, H.; GOLDMAN, A. A Delay-Tolerant Network Algorithm Based on Column Generation. In: **2013 12th International Symposium on Network Computing and Applications IEEE NCA13**. 2013.

BALASUBRAMANIAN, A.; LEVINE, B.; VENKATARAMANI, A. DTN routing as a resource allocation problem. In: **Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications**, SIGCOMM '07, p. 373–384. ACM, New York, NY, USA, 2007.

BUI XUAN, B.; FERREIRA, A.; JARRY, A. Computing shortest, fastest, and foremost journeys in dynamic networks. Rel. Téc. RR-4589, INRIA, 2002.

BURGESS, J. et al. Maxprop: Routing for vehicle-based disruption-tolerant networks. In: **Proceedings of IEEE INFOCOM**, volume 6, p. 1–11. Barcelona, Spain, 2006.

BURLEIGH, S. et al. Delay-tolerant networking: an approach to interplanetary Internet. **Communications Magazine, IEEE**, volume 41, nº 6, p. 128 – 136, 2003.

BURNS, B.; BROCK, O.; LEVINE, B. N. MORA routing and capacity building in disruption-tolerant networks. **Ad Hoc Networks**, volume 6, nº 4, p. 600 – 620, 2008.

CHEN, J.; SUBRAMANIAN, L.; LI, J. RuralCafe: web search in the rural developing world. In: **Proceedings of the 18th international conference on World wide web**, WWW '09, p. 411–420. ACM, New York, NY, USA, 2009.

CHENG, P.-C. et al. GeoDTN+Nav: Geographic DTN Routing with Navigator Prediction for Urban Vehicular Environments. **Mobile Networks and Applications**, volume 15, p. 61–82, 2010.

DANG, H.; WU, H. Clustering and cluster-based routing protocol for delay-tolerant mobile networks. **IEEE Transactions on Wireless Communications**, volume 9, nº 6, p. 1874 –1881, 2010.

DU, B.; DEMMER, M.; BREWER, E. Analysis of WWW traffic in Cambodia and Ghana. In: **Proceedings of the 15th international conference on World Wide Web**, WWW '06, p. 771–780. ACM, New York, NY, USA, 2006.

FALL, K. A delay-tolerant network architecture for challenged internets. In: **Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications**, SIGCOMM '03, p. 27–34. ACM, New York, NY, USA, 2003.

FENG, Y. et al. Minimum Expected Delay-Based Routing Protocol (MEDR) for Delay Tolerant Mobile Sensor Networks. **Sensors**, volume 10, nº 9, p. 8348–8362, 2010.

FERREIRA, A.; GOLDMAN, A.; MONTEIRO, J. Performance evaluation of routing protocols for MANETs with known connectivity patterns using evolving graphs. **Wireless Networks**, volume 16, nº 3, p. 627–640, 2010.

FORD, D. R.; FULKERSON, D. R.: *Flows in Networks*. Princeton University Press, Princeton, NJ, USA, 2010.

GUO, Z.; WANG, B.; CUI, J.-H. Prediction Assisted Single-Copy Routing in Underwater Delay Tolerant Networks. In: **Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE**, p. 1 –6. 2010.

HAY, D.; GIACCONE, P. Optimal routing and scheduling for deterministic delay tolerant networks. In: **Sixth International Conference on Wireless On-Demand Network Systems and Services, 2009. WONS 2009**, p. 27–34. IEEE, 2009.

HEIDEMANN, J. et al. Research challenges and applications for underwater sensor networking. In: **Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE**, volume 1, p. 228 –235. 2006.

JAIN, S.; FALL, K.; PATRA, R. Routing in a delay tolerant network. In: **Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications**, SIGCOMM '04, p. 145–158. ACM, New York, NY, USA, 2004.

JONES, E.; WARD, P. Routing strategies for delay-tolerant networks. **Submitted to ACM Computer Communication Review (CCR)**, 2006.

JONES, E. et al. Practical Routing in Delay-Tolerant Networks. **IEEE Transactions on Mobile Computing**, volume 6, nº 8, p. 943 –959, 2007.

JUANG, P. et al. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. **SIGOPS Oper. Syst. Rev.**, volume 36, nº 5, p. 96–107, 2002.

KARALIOPOULOS, M.; ROHNER, C. Trace-based performance analysis of opportunistic forwarding under imperfect node cooperation. In: **2012 Proceedings IEEE INFOCOM**, p. 2651 –2655. 2012.

LINDGREN, A.; DORIA, A.; SCHELÉN, O. Probabilistic Routing in Intermittently Connected Networks. In: P. Dini; P. Lorenz; J. de Souza, eds., **Service Assurance with Partial and Intermittent Resources**, *Lecture Notes in Computer Science*, volume 3126, p. 239–254. Springer Berlin / Heidelberg, 2004.

MALANDRINO, F. et al. Offloading cellular networks through ITS content download. In: **2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)**, p. 263–271. 2012a.

MALANDRINO, F. et al. Optimal Content Downloading in Vehicular Networks. **IEEE Transactions on Mobile Computing**, volume PP, nº 99, p. 1–1, 2012b.

MALANDRINO, F. et al. Optimal Content Downloading in Vehicular Networks. **IEEEEE Transactions on Mobile Computing**, volume 12, nº 7, p. 1377–1391, 2013.

MISHRA, S. et al. Economic Analysis of Networking Technologies for Rural Developing Regions. In: X. Deng; Y. Ye, eds., **Internet and Network Economics**, *Lecture Notes in Computer Science*, volume 3828, p. 184–194. Springer Berlin / Heidelberg, 2005.

MONTEIRO, J.; GOLDMAN, A.; FERREIRA, A. Using Evolving Graphs Foremost Journey to Evaluate Ad-Hoc Routing Protocols. In: **Proceedings of 25th Brazilian Symposium on Computer Networks (SBRC'07)**, p. 17–30. 2007.

MUSOLESI, M.; HAILES, S.; MASCOLO, C. Adaptive routing for intermittently connected mobile ad hoc networks. In: **Sixth IEEE International Symposium on World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005**, p. 183 – 189. 2005.

MUSOLESI, M.; MASCOLO, C. CAR: Context-Aware Adaptive Routing for Delay-Tolerant Mobile Networks. **IEEE Transactions on Mobile Computing**, volume 8, nº 2, p. 246 –260, 2009.

NAYEBI, A.; SARBAZI-AZAD, H.; KARLSSON, G. Performance analysis of opportunistic broadcast for delay-tolerant wireless sensor networks. **Journal of Systems and Software**, volume 83, nº 8, p. 1310 – 1317, 2010.

PASZTOR, B.; MUSOLESI, M.; MASCOLO, C. Opportunistic Mobile Sensor Data Collection with SCAR. In: **IEEE International Conference on Mobile Adhoc and Sensor Systems, 2007. MASS 2007**, p. 1 –12. 2007.

PATHIRANA, P. et al. Node localization using mobile robots in delay-tolerant sensor networks. **IEEE Transactions on Mobile Computing**, volume 4, nº 3, p. 285 – 296, 2005.

PERKINS, C. E.; BHAGWAT, P. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. **SIGCOMM Comput. Commun. Rev.**, volume 24, nº 4, p. 234–244, 1994.

RISTANOVIC, N.; THEODORAKOPOULOS, G.; BOUDEC, J.-Y. L. Traps and pitfalls of using contact traces in performance studies of opportunistic networks. In: **INFOCOM'12**, p. 1377–1385. 2012.

SOARES, V.; FARAHMAND, F.; RODRIGUES, J. A layered architecture for Vehicular Delay-Tolerant Networks. In: **IEEE Symposium on Computers and Communications, 2009. ISCC 2009**, p. 122 –127. 2009.

SPYROPOULOS, T.; PSOUNIS, K.; RAGHAVENDRA, C. S. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In: **Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking**, WDTN '05, p. 252–259. ACM, New York, NY, USA, 2005.

SPYROPOULOS, T.; PSOUNIS, K.; RAGHAVENDRA, C. S. Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility. In: **Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07**, p. 79 –85. 2007.

TATCHIKOU, R.; BISWAS, S.; DION, F. Cooperative vehicle collision avoidance using inter-vehicle packet forwarding. In: **Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE**, volume 5, p. 5 pp. –2766. 2005.

VAHDAT, A.; BECKER, D. et al. Epidemic routing for partially connected ad hoc networks. Rel. téc., Technical Report CS-200006, Duke University, 2000.

WANG, R. et al. Interplanetary Overlay Network (ION) for Long-Delay Communications with Asymmetric Channel Rates. In: **2011 IEEE International Conference on Communications (ICC)**, p. 1 –5. 2011.

WANG, Y. et al. Erasure-coding based routing for opportunistic networks. In: **Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking**, WDTN '05, p. 229–236. ACM, New York, NY, USA, 2005.

XIAN, Y.; HUANG, C.-T.; COBB, J. Look-Ahead Routing and Message Scheduling in Delay-Tolerant Networks. **Computer Communications**, volume 34, nº 18, p. 2184 – 2194, 2011.

YUN, Y.; XIA, Y. Maximizing the Lifetime of Wireless Sensor Networks with Mobile Sink in Delay-Tolerant Applications. **IEEE Transactions on Mobile Computing**, volume 9, nº 9, p. 1308 –1318, 2010.

ZHAO, W.; AMMAR, M. Message ferrying: proactive routing in highly-partitioned wireless ad hoc networks. In: **The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems, 2003. FTDCS 2003. Proceedings**, p. 308 – 314. 2003.

A TAXONOMIA

Neste apêndice, os protocolos de roteamento em DTNs mencionados na Seção 2 são classificados de acordo com os problemas que pretendem resolver segundo seus autores. Para facilidade de referência, estes são os algoritmos considerados.

Context-Aware Routing (**CAR**); *Erasure Coding* (**ERASURE**); *First Contact* (**FC**); *minimum estimated expected delay* (**MEED**); **MaxProp**; os algoritmos baseados em caminhos mínimos de Dijkstra apresentados por Jain et al. (2004) (**Partial**); o algoritmo baseado em programação linear apresentado por Jain et al. (2004) (**Fluxos**); **PROPHET**; **RAPID**; e *spray and wait* (**Spray**).

A Tabela 3 mostra tal classificação. Cada linha da Tabela representa um dos algoritmos supramencionados. Cada coluna representa um problema apresentado por alguns algoritmos. O significado de cada coluna está explicado nas Seções de A.1 a A.4. A entrada (i, j) da Tabela está marcada se o principal artigo que divulga o algoritmo da linha i menciona o problema da coluna j como algo a ser resolvido pelo algoritmo.

Tabela 3: Comparação de algoritmos por problemas resolvidos

Algoritmo	I	II	III	IV
CAR	✓			
Fluxos				✓
ERASURE		✓		✓
FC				✓
MaxProp			✓	
MEED	✓	✓	✓	
Partial				✓
PROPHET		✓		
RAPID	✓		✓	
Spray		✓	✓	✓

A.1 Falta de Informações (I)

Alguns algoritmos de roteamento requerem que os nós tenham conhecimentos sobre a rede que podem não ser realistas em muitas situações. O algoritmo chamado MED, ou *Minimum Expected Delay* (Jain et al., 2004), por exemplo, requer que cada

aresta do grafo que representa a rede seja pré-configurada com o atraso esperado de mensagens passando por ela. A suposição de que essa informação está disponível para os nós não é realista para um grande número de redes tolerantes a atrasos, como, por exemplo, redes cujos nós são sensores anexados a animais (Juang et al., 2002).

O protocolo CAR (Musolesi e Mascolo, 2009) evita, em parte, esse problema pois não requer conhecimento sobre as rotas de nós na rede ou localizações geográficas de nós baseadas em GPS. Também não requer o conhecimento profundo sobre a topologia da rede que é necessário para os algoritmos apresentados por Jain et al. (2004).

MEED (Jones et al., 2007) diminui a quantidade de informações necessárias para execução do algoritmo de caminhos mínimos de Dijkstra aplicado por Jain et al. (2004). Ao contrário deste último, que requer a pré-configuração do atraso esperado mínimo de cada aresta do grafo que representa a rede, o MEED calcula uma estimativa para esse atraso durante sua execução.

RAPID (Balasubramanian et al., 2007) não requer o uso de nenhum dos oráculos de informações sobre a rede apresentados por Jain et al. (2004).

A.2 Desperdício de Energia (II)

Algoritmos baseados na criação de cópias de uma mesma mensagem com o objetivo de que alguma delas chegue rapidamente a seu destino causam desperdício de energia por causa de transmissões desnecessárias de mensagens. Um dos protocolos com maior desperdício de energia causado por transmissões desnecessárias é o epidêmico (Vahdat et al., 2000). Nesse protocolo, em cada contato entre dois nós, todas as mensagens transportadas por um deles são transferidas para o outro, se este ainda não tiver uma cópia da mesma mensagem. Dessa forma, as mensagens se espalham pela rede rapidamente e atingem seus destinos com atraso mínimo. Porém, a maior parte de suas cópias são desnecessárias, e consomem energia a cada transmissão.

O algoritmo baseado em *Erasure-Coding* (Wang et al., 2005) tem o objetivo de distribuir pedaços de cada mensagem a um grande número de nós portadores, mas não aumenta a quantidade de *bytes* transmitidos quando comparado com algoritmos que transmitem cópias completas de mensagens, a partir do nó de origem, a um número fixo (e menor) de portadores.

MEED (Jones et al., 2007), apesar de usar um método epidêmico para disseminar o estado das ligações entre os nós para que cada nó possa calcular sua estimativa do atraso esperado mínimo em cada aresta que representa o grafo, não usa esse algoritmo para o encaminhamento das mensagens de dados.

O protocolo PROPHET (Lindgren et al., 2004) limita o número de transmissões por meio da transferência de mensagens de um nó para outro somente se o último apresenta uma alta probabilidade de entrega da mensagem a seu destino.

Spray and Wait (Spyropoulos et al., 2005) é um algoritmo baseado em replicação (múltiplas cópias da mesma mensagem) que limita o número de transmissões desnecessárias ao estabelecer que cada mensagem terá um número fixo L de cópias distribuídas a L nós vizinhos do nó-origem. Depois disso, cada um dos L portadores tem permissão de transferir suas cópias somente para o destino da mensagem, caso o encontrem.

A.3 Sobrecarga de *Buffers* (III)

De forma similar à seção A.2, algoritmos que criam várias cópias da mesma mensagem para maximizar a taxa de entrega e minimizar o atraso tipicamente causam um uso desnecessariamente alto de espaço em *buffer* dos nós da rede. Isso limita a escalabilidade no número de mensagens na rede (Jones et al., 2007; Zhao e Ammar, 2003). Além disso, muitos protocolos não utilizam um esquema de confirmação de entrega de mensagens a seus destinos, o que faz com que mensagens já entregues continuem ocupando espaço em *buffer* de nós intermediários (Burgess et al., 2006).

MaxProp (Burgess et al., 2006) usa um mecanismo inteligente para gerenciamento de *buffer*. Além de confirmação de entrega, usado para remover mensagens já en-

tregues dos *buffers* intermediários, quando o *buffer* de um nó fica cheio, o protocolo usa um critério baseado na probabilidade estimada de entrega de mensagens para remover do *buffer* aquelas com menor probabilidade.

Ao utilizar somente uma cópia de cada mensagem durante o roteamento, o algoritmo MEED (Jones et al., 2007) usa uma quantidade bem menor de *buffer* nos nós do que protocolos baseados em replicação.

O algoritmo RAPID (Balasubramanian et al., 2007) também utiliza confirmação de entrega de mensagens para remover dos *buffers* dos nós as mensagens cujas entregas já foram confirmadas.

O limite de L transmissões de cada mensagem pelo protocolo *Spray and Wait* (Spyropoulos et al., 2005) faz com que o uso de *buffers* nos nós da rede não seja tão alto quanto de protocolos epidêmicos.

A.4 Grandes Atrasos (IV)

Aplicações de DTNs não supõem pequenos atrasos, já que as características fundamentais da rede não permitem garantia de entrega rápida de mensagens. Porém, isso não significa que essas aplicações não seriam beneficiadas por atrasos menores. Protocolos fundamentalmente **reativos**, isto é, cujos nós esperam passivamente por oportunidades de transmissão de dados, podem apresentar atrasos significativos (Zhao e Ammar, 2003). Wang et al. (2005) mostram que, em casos extremos, o atraso médio de estratégias simples de replicação é infinito. Além disso, os protocolos mais comuns que atingem atrasos menores não são eficientes em termos de energia.

O protocolo baseado em *Erasure-Coding* (Wang et al., 2005) divide cada mensagem em blocos de código, os quais são espalhados por diversos nós da rede, chamados portadores, sem transmitir um maior número de *bytes* do que protocolos similares que transferem mensagens completas. Com um maior número de nós transportando frações de mensagens, há uma maior chance de que alguns desses nós entregarão partes suficientes da mensagem a seu destino com atraso pequeno de forma que a mensagem original possa ser reconstruída.

Ao empregar replicação limitada de mensagens, o *Spray and Wait* (Spyropoulos et al., 2005) consegue, como observado em simulações, limitar o desperdício de energia decorrente de transmissões desnecessárias e o desperdício de espaço em *buffers* intermediários enquanto mantém o atraso médio das mensagens baixo.

O protocolo FC (Jain et al., 2004) não evita o problema de atraso elevado por ser baseado em decisões aleatórias sem levar em consideração informações sobre a rede ou qualquer heurística para diminuir o atraso. Está marcado na coluna IV porque é divulgado no artigo de Jain et al. (2004), em que vários algoritmos são apresentados, e a maioria tem o objetivo de minimizar o atraso em vez de maximizar a taxa de entregas. O motivo é que os autores consideram que, além do fato de que atraso mínimo é uma boa aproximação para alta taxa de entrega, apesar de as aplicações serem tolerantes a atrasos, essas aplicações se beneficiariam de atrasos diminuídos. Os algoritmos *Partial* e *Fluxos* têm o objetivo de minimizar o atraso médio das mensagens usando as informações disponíveis.

A.5 Limitações

Esta Seção mostra uma classificação dos algoritmos selecionados de acordo com problemas que cada um tem a intenção de resolver segundo seus autores. Como consequência do fato de que eles foram publicados em datas potencialmente bastante separadas entre si, podemos assumir que os protocolos mais antigos podem não resolver os problemas mencionados satisfatoriamente segundo os critérios atuais.

Outra consequência do método de classificação é que os problemas relatados resolvidos por cada algoritmo podem não incluir necessidades mais relevantes satisfeitas pelo protocolo. A falta de um cenário de simulação com base na qual os algoritmos seriam comparados em diversos aspectos pode omitir o reconhecimento de alguma grande vantagem de um dos algoritmos sobre outros, a qual pode ter sido negligenciada pelo autor do protocolo.

B REVIEW AND ANALYSIS OF CITATIONS

Review and Analysis of Citations

Guilherme Amantea[1], **Alfredo Goldman**[2]

[1] Instituto de Pesquisas Tecnológicas (IPT)
São Paulo, SP, Brazil

[2]Departamento de Ciência da Computação
Instituto de Matemática e Estatística (IME)
Universidade de São Paulo (USP) – São Paulo, SP – Brazil

gcamantea@gmail.com, gold@ime.usp.br

Abstract. *This paper aims at classifying the citations of [Jain et al. 2004] according to their objectives. Also, we compare four different search tools according to the quantity of results in each class of citations per year after 2004, when the original work was published. Then, we use our results to criticize our method of citation review. We conclude that citations are not reported homogeneously by different tracking tools, the majority of citers do not take advantage of the cited work's specific content and the most appropriate classification of citations depends on the knowledge of the cited work's topic.*

1. Motivation

During the literature review of another work based on [Jain et al. 2004], we noticed that most of the texts that cite their article were weakly related to its specific contribution. Since the number of citations was high, we needed to identify the more relevant ones and concentrate our review on them. This motivated us to classify the references to Jain, Fall and Patra's work according to their citation purpose.

In order to do that, we first identified recurring reference objectives. Some of these were to extend the referred results. Others borrowed the model of a problem from the cited work. There were cases in which the reference's objective was to use a demonstrated result or a simulation environment introduced by Jain et al. The most frequent citation intention, however, was just to exemplify alternatives to the referring research about the same topic, with no particular interest in its content specifically.

We used four tools to find citers of [Jain et al. 2004]: Google Scholar ([Google]), ACM ([ACM]), CiteSeerx ([CiteSeerx]) and ISI Web of Knowledge ([ISI]). Since the different tools may yield varied citation counts and results ([Meho and Yang 2007], [Bakkalbasi et al. 2006]), we compared their output and whether some database was more relevant than the other. Then we gathered the texts for all the results in English to which we had access and classified them according to their relationship with [Jain et al. 2004].

The rest of this paper is organized as follows. Section 2 describes papers whose topics are relevant to this study. Section 3 contains a brief description of the content of [Jain et al. 2004]. Section 4 lists and explains the categories in which the referring works were classified. In section 5, we perform a systematic analysis of the collected data. Section 6 shows our results interpretations and conclusions.

2. Related Work

[Meho and Yang 2007] discuss if ISI databases (or Web of Science - WoS) are sufficient for locating citations of the work of library and information science (LIS) faculty members, which was used as a case study. They found that Scopus significantly alters the ranking of scholars as measured by ISI Web of Science by affecting the relative rating of the scholars that appear in the middle of the rankings. Also, Google Scholar stands out in its coverage of conference proceedings as well as international, non-English language journals.

[Bakkalbasi et al. 2006] analyze the performance of three different citation tracking engines - Google Scholar, Web of Science and Scopus - comparing their results of articles regarding two disciplines (oncology and condensed matter physics) that were written in 1993 and 2003 to test the hypothesis that the different scholarly publication coverage provided by the three search tools would lead to different citation counts from each. They found that the question "which tool provides the most complete set of citing literature?" may depend on the subject and publication year of a given article.

According to [Mercer and Di Marco 2003], the great amount of available scientific literature and the increasing number of digital libraries cause standard citation indexes to no longer be suitable for providing precise and accurate information and the solution is a better way to judge the relevancy of papers related to researchers specific needs. Thus, categorizing the citations evolved from citation analysis studies. In their work, they propose that fine-grained cue phrases within citation sentences may provide a basis for categorization.

[Garzone and Mercer 2000] propose an attempt to classify citations according to function in a fully automatic manner such that articles can be associated with 35 predefined categories listed in the paper's appendix.

3. Context

We considered [Jain et al. 2004] an appropriate subject because it might be of interest for several study fields and sub-fields. It includes elements of mathematics, represented by the time and cost models used by routing algorithms and the Linear Programming formulation, technology, represented by the work's main application and simulation environment, and proposes a change in the networking paradigm by introducing a new approach to Delay-Tolerant Networks (DTNs). Therefore, articles of different areas might make a reference to [Jain et al. 2004], which lowers the chance of limiting our results to particularities of authors regarding one topic.

[Jain et al. 2004] defines and motivates the routing problem in DTNs when the connectivity patterns are known. Then, it describes a model that represents the possible amounts of knowledge that a routing algorithm may have available about the network, and uses this model as an evaluation framework. Then it proposes six algorithms and classifies them according to the model. It compares the algorithms in a simulation environment which is also described.

The paper describes a model of a delay tolerant network based on a multi-graph where the edge capacities are time-dependent. It also defines a *contact* as an opportunity to send data over an edge (i.e. a time interval in which the edge capacity is strictly pos-

itive). Then it proposes an evaluation framework based on the definition of *knowledge oracles*, which are abstraction elements that encapsulate particular knowledge about the network required by different algorithms, and an algorithm classification according to the set of oracles they use. The oracles are *Contacts Summary*, *Contacts*, *Queuing* and *Traffic Demand*.

The proposed algorithms are separated into: zero knowledge, complete knowledge and partial knowledge. Those classified as partial knowledge are modified versions of Dijkstra's shortest path algorithm whose *cost* functions take into account the time at which a message arrives at a node and any other information the corresponding oracles may provide. They also provide an algorithm classified as complete knowledge based on a Linear Programming formulation that uses all the oracles to determine the optimal routing for minimizing average delay in the network.

The paper describes a DTN simulator and uses it to evaluate the proposed algorithms in two different scenarios. The conclusion is that the algorithms that use more information about the network (i.e. more oracles or more informed oracles) outperform the simpler algorithms, both in terms of delay and delivery ratio.

4. Categories

The citers of [Jain et al. 2004] were assigned to categories according to the purpose of each of the references to that work. The categories are as follows.

- *Illustration*: The reference's objective is to exemplify alternatives to the citer's work or portion of it. In this category, there are references in the *Related Work* sections.
- *Competition*: The reference's objective is to compare the performance of an algorithm of [Jain et al. 2004] with another one, typically proposed by the citer.
- *Motivation*: The reference is about the motivation of [Jain et al. 2004] or uses it as a motivation for the citer.
- *Comparison*: The reference's objective is to clarify that the problem being introduced by the citer is different from the one presented by Jain et al.
- *Extension*: The reference's objective is to modify the network model or the evaluation framework or some or all the algorithms presented by Jain et al. in order to extend or improve them.
- *Vocabulary*: The reference's objective is to use the definition of some term presented by Jain et al.
- *Model*: The reference's objective is to use a model introduced by Jain et al. or a portion of it (e.g. network model and evaluation framework).
- *Usage*: The reference's objective is to use the definition of some concept presented by Jain et al. or some demonstrated result.
- *Simulation*: The reference is about the simulation environment developed by Jain et al. or one that could be used in their work.
- *Formulation*: The reference's objective is to use the LP formulation introduced by Jain et al.

Although some of these categories are general enough to be appropriate to categorize citations to any work, others are particularly relevant to [Jain et al. 2004] (e.g. Simulation and Formulation). This set of categories is, therefore, not a subset of those proposed by related work ([Garzone and Mercer 2000]).

5. Analysis

During one week (from July 18th to 25th, 2010) four tools were used to find citers of [Jain et al. 2004], in this order: Google Scholar, CiteSeerx, ACM and ISI. First, we downloaded the search results texts to use them for the analysis. This was important because the same search in the same tool yielded different results at different times, so we used the snapshot of each search at a specific date.

5.1. Methodology

Our first step, for each citer, was to determine whether it would be considered. A given citer was not considered if, and only if, it satisfied at least one of the following criteria.

- The work is not in English;
- The work is marked as *Citation* by Google Scholar;
- The link to the work was broken;
- We did not have free access to the work;
- The work is a book.

If a work did not satisfy any of the above, we searched for references to [Jain et al. 2004] on it. Each reference was assigned to one of the categories described in Section 4, then we stored the citation's year of publication, title, list of authors and list of categories in custom format files that facilitated the subsequent data retrieval: one file for each tool that hold one line for each citer. The line format is the following.

```
<Year>;<Title>;<List of authors>;<List of categories>
```

We could not find all the above information for all citations. Many search results did not offer the year of publication. We discovered some of them by searching for their title on Google.

In some cases, the list of categories is empty. These are the citers that contain no explicit reference to the work of Jain et al. in their texts, but only in their list of references at the end.

5.2. Results

During the data collection process, we noticed that three categories' definitions did not have a very clear distinction between each other, which led to confusion about which should be used to classify the references. They are *Vocabulary*, *Model* and *Usage*. In order to improve the reliability of the results, we decided to merge them into the *Usage* category.

Table 1 shows the total number of citations for each database per year and also those with no associated year of publication. Figure 1 shows a graphical representation of Table 1.

Table 2 shows the representativity of each database search result per year. It is interesting to observe that, up to 2007, CiteSeerx seemed to have been more "active" than ACM, yielding more results. From that year on, this characteristic inverted.

The next images display the summary of the collected data for each database by year by category. Each bar represents the total number of citations of a fixed category in a specific database result set in a given year.

Table 1. Total number of works per year of publication

General	2004	2005	2006	2007	2008	2009	2010	No Year
Google	6	28	62	67	92	131	37	43
CiteSeerx	4	22	36	31	25	22	5	8
ACM	1	11	17	35	28	53	12	2
ISI	0	1	0	4	5	12	1	0

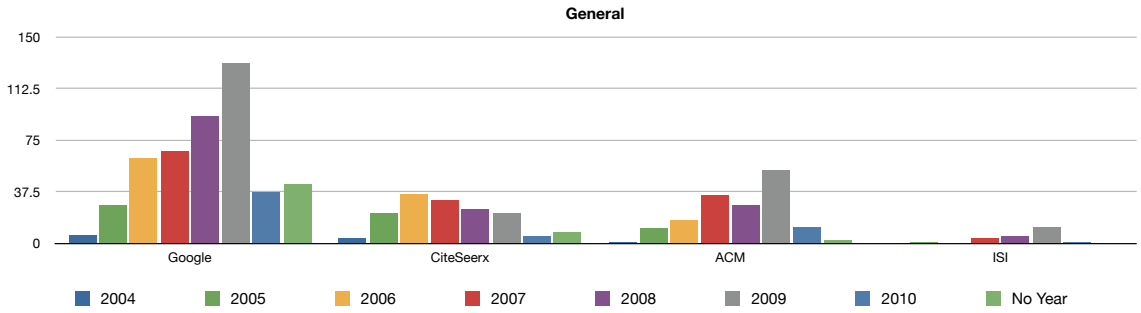

Figure 1. Graphical representation of Table 1.

Table 2. Representativity of each base per year

Representativity	2004	2005	2006	2007	2008	2009	2010	No Year
Google	54.5%	45.2%	53.9%	48.9%	61.3%	60.1%	67.3%	81.1%
CiteSeerx	36.4%	35.5%	31.3%	22.6%	16.7%	14.7%	9.1%	15.1%
ACM	9.1%	17.7%	14.8%	25.5%	18.7%	35.3%	21.8%	3.8%
ISI	0.0%	1.6%	0.0%	2.9%	3.3%	8.0%	1.8%	0.0%

Figure 2 displays the evolution of citation categories over the years in Google Scholar. We can observe a tendency of increasing number of citations in each category in the period between the years 2004 and 2009.

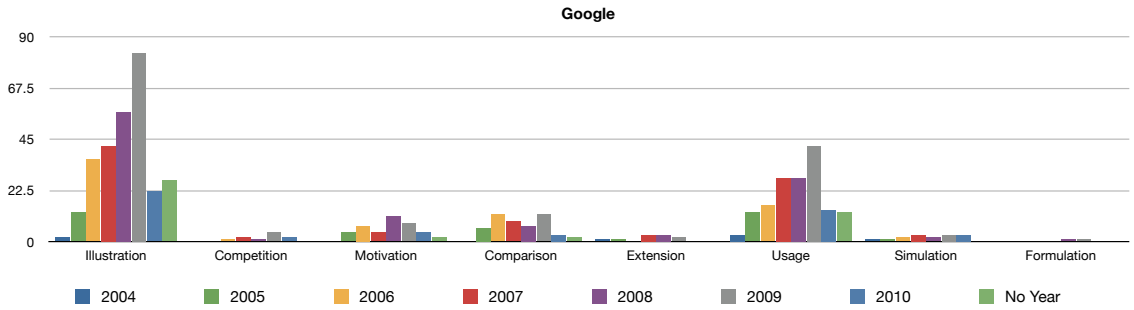

Figure 2. Number of citations in each category by year from Google Scholar.

In spite of what we can clearly see in Figures 2 and 4, Figure 3, which shows the results for CiteSeerx, displays no tendency that the number of citations per category will increase every year. In fact, after 2006 and 2007, it seems that the number of citations per category will decrease with time.

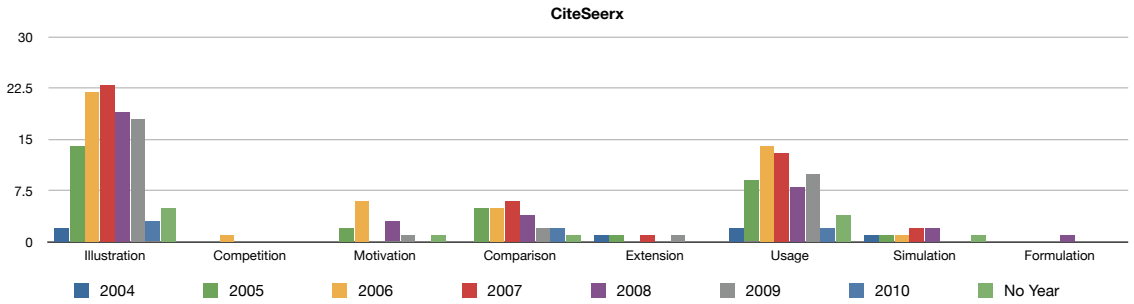

Figure 3. Number of citations in each category by year from CiteSeerx.

Figure 4 reports the results in ACM. Again, as in Figure 2, we can observe that the number of citations per category increases with each year.

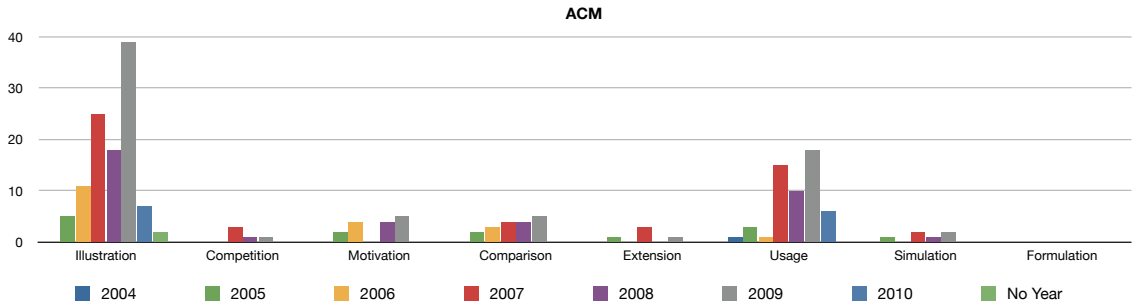

Figure 4. Number of citations in each category by year from ACM.

Figure 5 represents the results from ISI. Since the number of works analyzed in this database is very small, any observation based on this graph does not have a high relevance. However, there are hints that the number of citations per category might also be increasing with each year.

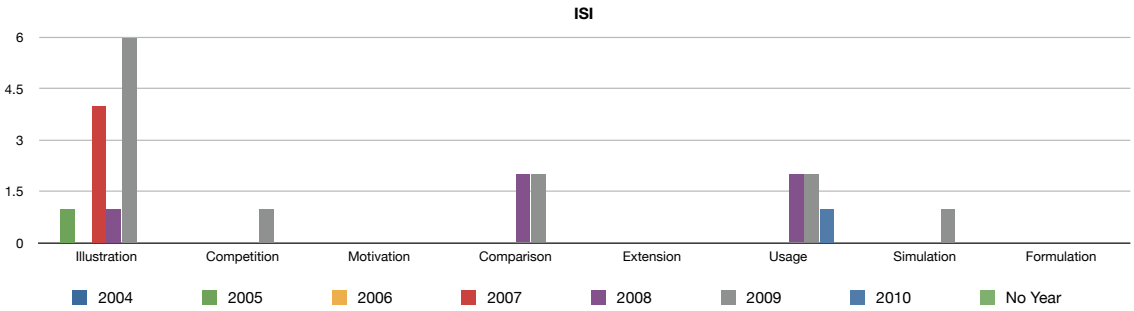

Figure 5. Number of citations in each category by year from ISI.

Another interesting observation about the different search results is that, although they have similar objectives (i.e., to report all the citations of a given work), there are papers present in all of them, but also some that are listed by one of the tools and not by the others. None of the four databases yields a result set that contains those from all the other three. Even Google Scholar, which clearly offers the largest number of search results, does not find all the citations reported by the other tools. Table 3 shows how many works were found by each tool on the rows that were not detected by the tools on the columns, considering only those that were assigned to some category.

Table 3. Works not found between databases

Cross	Google	CiteSeerx	ACM	ISI
Google	-	349	320	452
CiteSeerx	30	-	94	144
ACM	8	101	-	146
ISI	9	20	15	-

In the analysis of the results, it also stands out that the category *Illustration* contains the greatest number of citations, which means that many of the references to the paper of Jain et al. did not use its specific content to build their own work upon it, but instead, they only mentioned its existence, typically in the *Related Work* sections.

We can also observe that the number of citations is likely to increase with each year in three databases, but, for some reason, CiteSeerx does not follow this rule.

6. Conclusion

The results reported by this paper were collected over a period of five months of non-exclusive work. The authors had to analyze over one thousand of search results and over five hundred works to achieve a more appropriate literature review about [Jain et al. 2004]. We believe that this exhaustive approach is not practical and alternative methods should be considered.

It is evidenced by this study that none of the analyzed tools is enough to find a complete list of citations of a given work. They produce far from similar results. More tracking engines must used for a citation analysis to have a more comprehensive outcome.

Another conclusion is that most citations of [Jain et al. 2004] are in the *Illustration* category, which means that [Jain et al. 2004]'s content was not relevant to most of its

citers. This leads us to believe that pure number of citations may be a misleading measure of quality for a research.

Also, as supported by [Mercer and Di Marco 2003], the huge amount of scientific literature available makes it difficult for researchers to draw precise and accurate information without categorizing it according to their needs. However, in spite of the very useful method of automatic categorization proposed in [Garzone and Mercer 2000], the most appropriate categories for some specific researcher needs (e.g. detecting works that use [Jain et al. 2004]'s simulation tool) must be created using some knowledge about the referred paper.

Future work includes collecting similar data about citations of other works to investigate whether our conclusions can be generalized. Another topic for future research is to replicate this study to reduce the possibility of bias in the results inherent to data analysis performed by few observers. Since, for this paper, citations were subjectively associated with categories by one of the authors, replication would be beneficial.

References

ACM. The acm portal. `http://portal.acm.org/`.

Bakkalbasi, N., Bauer, K., Glover, J., and Wang, L. (2006). Three options for citation tracking: Google Scholar, Scopus and Web of Science. *Biomedical Digital Libraries*, 3(1):7.

CiteSeerx. Citeseerx. `http://citeseerx.ist.psu.edu`.

Garzone, M. and Mercer, R. (2000). Towards an automated citation classifier. *Advances in Artificial Intelligence*, pages 337–346.

Google. Google scholar. `http://scholar.google.com`.

ISI. Isi web of knowledge. `http://isiknowledge.com/`.

Jain, S., Fall, K., and Patra, R. (2004). Routing in a delay tolerant network. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 145–158. ACM.

Meho, L. and Yang, K. (2007). Impact of data sources on citation counts and rankings of LIS faculty: Web of Science versus Scopus and Google Scholar. *Journal of the American Society for Information Science and Technology*, 58(13):2105–2125.

Mercer, R. and Di Marco, C. (2003). The importance of fine-grained cue phrases in scientific citations. *Advances in Artificial Intelligence*, pages 995–995.