Uma Ferramenta para Mineração Multi-relacional de Redes Sociais Baseada em Programação Lógica e *Workflows**

Manuel Almeida, Aline Paes e Daniel de Oliveira

Instituto de Computação – Universidade Federal Fluminense (IC/UFF)
Niterói – Rio de Janeiro – Brasil

manuelmba@id.uff.br,{alinepaes, danielcmo}@ic.uff.br

Resumo. *Técnicas de mineração de dados multi-relacional (MDMR) constituem as estratégias mais apropriadas para lidar com bases de dados contendo múltiplos relacionamentos entre entidades não homogêneas, o que é justamente o caso de bases obtidas a partir de redes sociais. Entretanto, o espaço de busca de hipóteses candidatas de tais estratégias é mais complexo do que aqueles obtidos a partir de técnicas tradicionais de mineração de dados. Para permitir uma busca factível no espaço de hipóteses, as técnicas de MDMR adotam vieses de linguagem e de busca ao processo de mineração. Porém, uma análise experimental detalhada requer a combinação de vários parâmetros distintos, o que torna o controle manual de tal processo complexo. Nesse artigo, é apresentada uma ferramenta que instancia um workflow científico para a análise de um processo de MRDM, modelado a partir do Sistema de Gerência de Workflows SciCumulus, denominada LPFlow4SN. Por controlar o processo experimental de forma automática, o LPFlow4SN tem o potencial de tornar a mineração de redes sociais eficiente.*

1. Introdução

As tecnologias de redes sociais tem possibilitado a inserção de novos costumes e práticas em nossa sociedade (Baksh*y et al.* 2012). Por meio de redes sociais, membros podem trocar informações, promover eventos e debates, *etc*. Entretanto, existem informações sobre as redes sociais que não são facilmente detectadas. Por exemplo, considere uma rede social de alunos, professores, colaboradores e disciplinas de uma universidade. Suponha que um relacionamento a ser representado dentro desse contexto poderia ser o de orientação. A única informação explícita que temos nessa representação é de que um professor *X* orienta o aluno *Y*. Entretanto, se considerarmos outras características como disciplinas ministradas ou o cargo na universidade podemos obter regras que não são triviais de serem extraídas, como por exemplo: "se o professor *X* ministrou a disciplina *C* no período *Q*, a disciplina *C* é de nível de pós-graduação e *Y* foi assistente de pesquisa na disciplina, então *Y* é orientado por *X"*. Para extrair tal tipo de informação necessitamos de algoritmos de mineração de dados.

Entretanto, um problema em utilizar os algoritmos de mineração de dados tradicionais é que eles assumem que os dados são homogêneos, independentes uns dos outros e descritos no formato de atributo-valor (Ha*n et al.* 2011). Como os dados obtidos a partir de redes sociais são comumente multi-relacionados, estruturados e amostrados a partir de relacionamentos complexos (como por exemplo, o caso anteriormente apresentado) essa análise se torna inviável. Assim, devido a essa suposição básica, os algoritmos tradicionais

* Os autores gostariam de agradecer ao CNPq e a FAPERJ pelo apoio financeiro

como o C4.5 e o Apriori (Ha*n et al.* 2011) que respectivamente induzem árvores de decisão e regras de associação, falham em capturar e descrever corretamente e de forma expressiva como os indivíduos de redes sociais se relacionam, bem como em expressar características desses relacionamentos.

Por outro lado, a área de pesquisa de mineração de dados multi-relacional (Džeroski e Lavrač 2001) (*i.e.* MDMR) tem como objetivo principal induzir automaticamente hipóteses a partir de dados dispostos em múltiplas tabelas relacionadas. Assim, algoritmos de MDMR se tornam bastante adequados para descobrir padrões e descrever características dos relacionamentos inseridos em uma rede social. Nesse contexto, a área de Programação em Lógica Indutiva (*i.e.* ILP do inglês *Inductive Logic Programming*) (Raedt 2008) tem obtido bastante destaque em sua aplicação para MDMR, sendo a interseção de aprendizado de máquina e representação do conhecimento. Em ILP, as hipóteses são induzidas no formato de programas lógicos e com isso se beneficiam do poder de expressividade da lógica de primeira-ordem (Nilsson e Małuszyński 1995). Além dos exemplos, um conhecimento preliminar a respeito do problema também pode ser fornecido como entrada para os algoritmos de ILP, ambos descritos na linguagem de lógica de primeira-ordem.

Porém, a descoberta de hipóteses mais expressivas vem acompanhada de estratégias de indução mais complexas, e, portanto menos eficientes, do que as utilizadas nas abordagens tradicionais. Em ILP, por exemplo, o refinamento de uma hipótese requer a verificação de quais exemplos são inferidos a partir da hipótese corrente e do conhecimento preliminar. Adicionalmente, para diminuir o espaço de possibilidades de hipóteses a serem refinadas, é necessário fornecer vieses para a linguagem e para o algoritmo de refinamento. Essas entradas não são definidas trivialmente, o que faz com que vários valores diferentes para os parâmetros envolvidos na análise precisem ser experimentados durante o processo. Uma vez que todos os valores possíveis para os parâmetros tenham sido avaliados temos maior chance de que boas hipóteses sejam obtidas. Entretanto, como essa avaliação das combinações de valores tende a ser uma tarefa computacionalmente custosa e propensa a erros, nem todos os analistas optam por avaliar todas as combinações. Essa simplificação insere um risco ao processo de análise como um todo, pois resultados válidos podem ser descartados.

Ao mesmo tempo, a comunidade de *e-Science* vem trabalhando em técnicas para encadeamento de programas científicos, os chamados *workflows* científicos (Mattos*o et al.* 2010). Como um processo de MDMR é composto pelo encadeamento de uma série de aplicações, é intuitivo que modelemos processos de MDMR como um *workflow* científico e nos aproveitemos dos Sistemas de Gerência de *Workflows* Científicos (SGWfC). Essas soluções incluem apoio a execuções em ambientes de computação de alto desempenho (*e.g. clusters* e *grids*), que podem proporcionar ao analista a oportunidade de explorar todas as combinações de parâmetros possíveis em um tempo viável. Adicionalmente, os SGWfCs proveem os chamados dados de proveniência (Freir*e et al.* 2008) que registram um histórico completo da execução. Esse histórico é útil para o analista realizar uma análise *post-mortem* do seu experimento. Desta forma, podemos utilizar ferramentas de apoio a *workflows* científicos como forma de melhorar a gerência do processo de descoberta de conhecimento em MDMR. Entretanto, modelar um *workflow* científico requer que o analista conheça os sistemas disponíveis e também tenha domínio sobre as configurações dos ambientes, o que pode não ser trivial.

No sentido de auxiliar os analistas a se beneficiarem da tecnologia de *workflows* científicos em ambientes de computação de alto desempenho para realizar sua análise de redes sociais, foi desenvolvida uma ferramenta chamada LPFlow4SN (*i.e. Logic*

Programming Workflow for Social Networks). A LPFlow4SN é uma aplicação que instancia um *workflow* de MDMR baseado em ILP e modelado no SGWfC SciCumulus (Oliveir*a et al.* 2010) e que executa o processo de análise de redes sociais em paralelo em nuvens de computadores. Na LPFlow4SN o analista não precisa se preocupar em configurar o processo de análise, devendo somente informar qual coleção de dados será analisada. A LPFlow4SN verifica quais combinações de parâmetros devem ser processadas e executa o processo todo em diversas máquinas virtuais em paralelo na nuvem. Para o melhor do nosso conhecimento, esta é a primeira ferramenta capaz de executar experimentos com dados multi-relacionais representados em lógica a partir de *workflows* científicos.

Este artigo está organizado em 4 seções além desta introdução. A Seção 2 apresenta um levantamento conceitual sobre *workflows* científicos e programação em lógica indutiva. A Seção 3 apresenta a abordagem LPFlow4SN. A Seção 4 apresenta o estudo de caso executado com a LPFlow4SN. Finalmente, a Seção 5 conclui este artigo e aponta os trabalhos futuros.

2. Referencial Teórico

2.1 Programação em Lógica Indutiva

Algoritmos de mineração de dados tradicionais usualmente utilizam bases de dados expressas em um formato de representação proposicional, no sentido de que buscam por padrões a partir de dados homogêneos, amostrados a partir de relações simples. Entretanto, dados de redes sociais usualmente contém múltiplas entidades, dispostas tipicamente em várias tabelas relacionadas. A ILP é uma subárea de aprendizado de máquina que possibilita a mineração de dados multi-relacional ao combinar aprendizado de máquina e representação de conhecimento para induzir hipóteses expressivas, escritas no formato de programas lógicos.

A entrada de um sistema de ILP é composta de um conjunto de exemplos, divididos em exemplos positivos e exemplos negativos, um conhecimento preliminar (*i.e.* BK) e um conjunto de parâmetros que especificam o viés da linguagem e do espaço de busca de uma hipótese. Para melhor exemplificar, consideremos o mesmo cenário apresentado na introdução: uma rede social de alunos, professores e colaboradores de uma universidade (Richardson e Domingos 2006). Outras entidades nesse domínio poderiam ser os projetos em desenvolvimento na universidade, as disciplinas ministradas, publicações, *etc*. Um relacionamento típico a ser explicado dentro desse contexto poderia ser o de orientação. Assim, os exemplos positivos seriam amostras conhecidas de tal relacionamento, enquanto que exemplos negativos seriam amostras de indivíduos que não são coparticipantes de tal relacionamento. Supondo, por exemplo, que *person1* é um aluno e *person10* é um professor, um exemplo positivo para um sistema de ILP seria *advisedBy(person1,person10),* caso *person1* seja orientado por *person10*. Por outro lado, um exemplo negativo poderia ser *advisedBy(person11,person10),* considerando que *person11* não é orientado por *person10*. O conhecimento preliminar é composto de *fatos* e *cláusulas* lógicas (Nilsson e Małuszyński 1995), que permitem propriedades e características dos indivíduos do domínio, bem como regras de regularidade previamente conhecidas. No mesmo domínio anterior, o conhecimento preliminar poderia conter um fato lógico *projectMember(project1, person10)*, especificando que *person10* participa de *project1*. Um exemplo de cláusula lógica nesse contexto seria:

courseLevel(C,basiclevel) :- ta(C,S,Q), inPhase(S,pre_Quals), student(S), course(C). (1)

que pode ser lida em linguagem natural da seguinte forma: *"se S foi um assistente de ensino no curso C e no período Q, S está na fase de pré-qualificação, S é um estudante e C é uma disciplina, então C é uma disciplina do nível básico*". Observe que C e S são *variáveis* que

podem ser substituídas por entidades quaisquer no domínio, mas no momento em que elas são substituídas por um indivíduo em algum literal da regra, ela não pode ser substituída por outro indivíduo em outro literal da mesma regra.

A saída de um sistema de ILP será um programa lógico, composto por cláusulas em lógica de primeira-ordem. No domínio exemplificado anteriormente, o sistema poderia induzir a seguinte regra, a partir do conjunto de exemplos e do BK:

advisedBy(S, P) :- taughtBy(C, P, Q), courseLevel(C, graduateLevel), ta(C, S, Q). (2)

que indica que uma disciplina de pós-graduação ministrada por um professor teria como assistente de ensino um orientando daquele professor. Ou seja, "*se o professor P ministrou a disciplina C no período Q, a disciplina C é de nível de pós-graduação e S foi assistente de pesquisa na disciplina, então S é orientado por P*".

2.2 *Workflows* Científicos

Os *workflows* científicos são uma alternativa atraente para representar os encadeamentos de programas ao invés de usarmos uma abordagem *ad hoc* manual ou baseada em *scripts*. Os *workflows* científicos podem ser definidos como uma abstração para modelar o fluxo de atividades e de dados. Em *workflows* científicos, essas atividades são geralmente programas ou serviços que representam algoritmos e métodos computacionais sólidos. Tais *workflows* são gerenciados por complexos sistemas chamados de Sistemas de Gerência de *Workflows* científicos (*i.e.* SGWfC) (Mattos*o et al.* 2010). Existem diversos SGWfC, cada qual com uma característica principal como apoio a visualização, uso de ontologias ou mecanismos de computação de alto desempenho.

Uma das principais vantagens dos SGWfCs está nos recursos de armazenamento de metadados do experimento, chamados de dados de proveniência (Freire *et al.* 2008). Registrar a origem e o histórico de uma informação em um experimento científico é fundamental para que os procedimentos executados (no caso o *workflow*) e os dados consumidos e produzidos sejam validados e passíveis de reprodução por parte de terceiros. Desta forma, Freire *et al.* definem o termo como a descrição da origem de um dado e o processo pelo qual este chegou a um banco de dados. Estes metadados são fundamentais em processos de MDMR, pois em uma mesma análise, o *workflow* pode ser executado *n* vezes apenas variando seus parâmetros.

Por exemplo, um componente essencial do processo de refinamento de uma hipótese é a função de otimização utilizada para pontuar as hipóteses candidatas. Comumente, a função de otimização empregada é a cobertura de exemplos, ou seja, a diferença entre a quantidade de exemplos positivos inferidos corretamente pela hipótese candidata e a quantidade de negativos inferidos incorretamente. Entretanto, caso a base de treinamento seja desbalanceada, ou seja, existem muito mais exemplos negativos do que positivos (ou vice-versa), essa função guiará o refinamento para uma hipótese muito específica, que erroneamente pode classificar uma parcela muito pequena dos exemplos positivos. Usualmente, redes sociais geram bases de dados desbalanceadas, uma vez que, considerando todos os indivíduos envolvidos, é mais fácil que não exista um relacionamento entre um par de indivíduos quaisquer. Nesses casos, as funções de otimização que dão mais importância a exemplos positivos classificados corretamente do que a exemplos negativos cobertos, como a medida F (Powers 2007) são mais apropriadas. É fundamental que se saiba exatamente qual resultado está associado a qual medida de avaliação no momento da análise, até mesmo para desconsiderar em experimentos futuros que tenham as mesmas características do experimento atual as funções que não guiaram para boas hipóteses. Da maneira como o analista realiza a

análise hoje, todo esse controle é realizado de forma manual, *i.e.* susceptível a erros. Uma sistematização do processo com o uso de *workflows* científicos é uma solução com muitas vantagens, como, por exemplo, o registro do conhecimento tácito do processo de MDMR. Quando lidamos com processos de MDMR, é também comum a necessidade de ambientes de computação de alto desempenho para que o processo possa ser paralelizado. Para tal, devemos utilizar um SGWfC que proveja capacidades de paralelismo em ambientes de computação de alto desempenho aliadas a captura dos dados de proveniência das análises. Entretanto, modelar um *workflow* científico nestes SGWfC e nesses ambientes não é uma tarefa trivial de ser desempenhada.

3. LPFlow4SN: uma ferramenta de para mineração multi-relacional de redes sociais

Nesta seção apresentamos a ferramenta LPFlow4SN que é implementada sobre o SGWfC SciCumulus e um *workflow* modelado para análise de redes sociais.

3.1 O SciCumulus

O SciCumulus é um SGWfC projetado para distribuir e controlar a execução paralela de *workflows* científicos em um ambiente de nuvem, como o Amazon EC2. O SciCumulus orquestra a execução das atividades do *workflow* científico em um conjunto distribuído de máquinas virtuais (*i.e. cluster* virtual), oferecendo dimensionamento dos recursos durante o curso de execução do *workflow* e mecanismos de tolerância a falhas por meio de seus serviços. Para isolar os cientistas da complexidade de distribuir as atividades dos *workflows* científicos em ambientes de nuvem, o SciCumulus fornece mecanismos para oferecer paralelismo por varredura de parâmetros. Este é exatamente o caso das tarefas de mineração de dados multi-relacionais onde um mesmo *workflow* (que representa o processo de mineração) é executado variando-se os *datasets* de entrada e os valores dos parâmetros. Controlar esse tipo de paralelismo em ambientes de nuvem é difícil quando a gerência é feita de forma *ad hoc* devido à grande quantidade de execuções de atividades a serem gerenciadas. Desta forma, o SciCumulus fornece a infraestrutura necessária para dar o apoio computacional para o paralelismo de *workflows* com coleta de proveniência distribuída. O SciCumulus foi a escolha de SGWfC para implementarmos o LPFlow4SN uma vez que ele provê capacidades de paralelismo (necessária para análises multi-relacionais) aliado a utilização da nuvem, que pode ser usada independentemente do usuário possuir uma infraestrutura própria ou não. A seguir apresentamos como o processo de mineração multi-relacional foi modelado como um *workflow* científico.

3.2 O Processo de MDMR em Redes Sociais como um *Workflow* Científico

Execuções distintas do processo de descoberta de hipóteses em redes sociais utilizando algoritmos de MRDM empregam parâmetros específicos para enviesar a linguagem e o espaço de busca, em busca da melhor hipótese. Tal como no ciclo de vida de um experimento científico tradicional, o analista de MDMR deve executar, de forma controlada, todas as etapas do processo de MDMR.

O poder de expressividade obtido com abordagens de MDMR vem acompanhado de espaços de busca inerentemente mais complexos. Para que o processo seja viável torna-se então necessário definir parâmetros que norteiem o espaço de busca de hipóteses. Uma combinação específica de parâmetros pode fazer com que o processo de mineração retorne hipóteses mais acuradas. Entretanto, não existe um conjunto de parâmetros que possa ser

usado de forma eficaz para qualquer domínio e encontrar a melhor combinação (ou ao menos uma combinação boa o suficiente), não é uma tarefa trivial. Assim, apesar de a ideia por trás do *workflow* apresentado nessa seção ser simples, o mesmo demanda um alto poder computacional, pois existem diversas combinações de parâmetros para serem testadas antes de retornar a hipótese final a ser utilizada pelo especialista. Além das várias combinações de parâmetros a serem exploradas, para minimizar riscos de superajustar a hipótese, o processo experimental em mineração de dados faz uso do mecanismo de validação cruzada (Ha*n et al.* 2011). Na validação cruzada, os exemplos da base de dados são divididos em *k* partições e o processo experimental é executado *k* vezes. Na primeira iteração, a primeira partição é usada para validar a hipótese, aprendida com os exemplos de treinamento obtidos a partir das *k-1* demais partições. Na segunda iteração, a segunda partição é escolhida para validar o modelo e as demais são usadas para treiná-lo, e assim sucessivamente. Ao final, cada partição terá participado de alguma iteração como conjunto de validação. Embora tal mecanismo seja essencial para produzir hipóteses que generalizem o conjunto de exemplos, sua utilização introduz um elemento a mais de complexidade no processo experimental, uma vez que cada iteração do processo de mineração será executada *k* vezes. Ao utilizar o SciCumulus, a indução de hipóteses com cada configuração de parâmetros pode ser executada em várias máquinas virtuais na nuvem, reduzindo o tempo total de execução do *workflow* (e muitas vezes tornando essa análise viável).

No *workflow* desenvolvido, o arcabouço de MDMR utilizado é o Aleph (***A L**earning **E**ngine for **P**roposing **H**ypothesis*) (Muggleton 1995). O sistema Aleph foi desenvolvido originalmente para induzir hipóteses usando o paradigma de resolução inversa, mas com o tempo ele evoluiu para emular funcionalidades de diversas outras abordagens de ILP. Devido a esse fato, atualmente Aleph é o sistema mais usado para fins de comparação com novos algoritmos desenvolvidos na área de MDMR. Além disso, como seu código é livre e aberto, o Aleph se mostra como um sistema para desenvolvimento e exploração de novas abordagens de MDMR. Todas essas razões fazem com que o sistema Aleph seja ideal para compor o *workflow* desenvolvido neste artigo. Para executar o Aleph, é necessário um compilador Prolog como o YAP. A entrada fornecida ao sistema Aleph é disposta em três arquivos Prolog: (i) um arquivo com a extensão "*.f"* que contém os exemplos positivos; (ii) um arquivo com a extensão "*.n"* que contém os exemplos negativos e (iii) um arquivo com a extensão "*.b"* que contém o conhecimento preliminar, no formato de fatos e/ou regras, as restrições da linguagem e da busca e quaisquer outras diretivas entendidas por um interpretador Prolog. Todas as informações contidas neste último arquivo também podem ser repassadas diretamente ao interpretador Prolog.

O Aleph possui diversos parâmetros que devem ser explorados a fim de encontrarmos resultados com maior acurácia. Por exemplo, o Aleph assume por *default* que o tamanho máximo de uma regra é de 5 literais. Entretanto, em ILP, cláusulas maiores poderão constituir hipóteses mais específicas e que, portanto cobrirão menos exemplos negativos. Assim, em alguns domínios pode ser útil gerar cláusulas com mais que 5 literais, mas que ainda tenham um tamanho máximo de forma que a inferência não seja muito custosa. Com o parâmetro *clauselength* é possível estabelecer esse tamanho máximo. Novamente, problemas diferentes podem requerer cláusulas com tamanhos máximos diferentes. Outra possibilidade de parâmetro que pode impactar na indução de regras é o que define a função de otimização a ser empregada durante a busca, chamado de *evalfn*. Algumas possibilidades de valores para esse parâmetro são "*coverage*", que tenta aperfeiçoar a cobertura de exemplos positivos sobre os negativos; "*accuracy*", que faz com que o objetivo da busca seja maximizar o número de exemplos classificados de forma correta. Outro exemplo que costuma gerar hipóteses bastante

diversificadas é o parâmetro *noise*, cuja função é estabelecer uma quantidade máxima de exemplos negativos a serem cobertos por uma cláusula, a fim de criar cláusulas menos superajustadas aos exemplos de treinamento. O sistema Aleph acomoda a variação de muitos outros parâmetros. Para uma descrição completa de todos eles, por favor, faça referência ao manual do Aleph (Muggleton 1995).

Assim, utilizando o arcabouço Aleph, desenvolvemos o *workflow* utilizado como base para o LPFlow4SN. O *workflow* que apresentamos a seguir permite que qualquer combinação de valores de quaisquer parâmetros do sistema sejam fornecidos como viés da linguagem e/ou do espaço de busca em uma rede social. A fim de permitir um processo experimental extensivo, explorando todas as possibilidades de configuração do Aleph, o *workflow* proposto associa uma execução de atividade a cada fase da análise a ser executada. O *workflow* é composto por três atividades conforme apresentado na Figura 1.

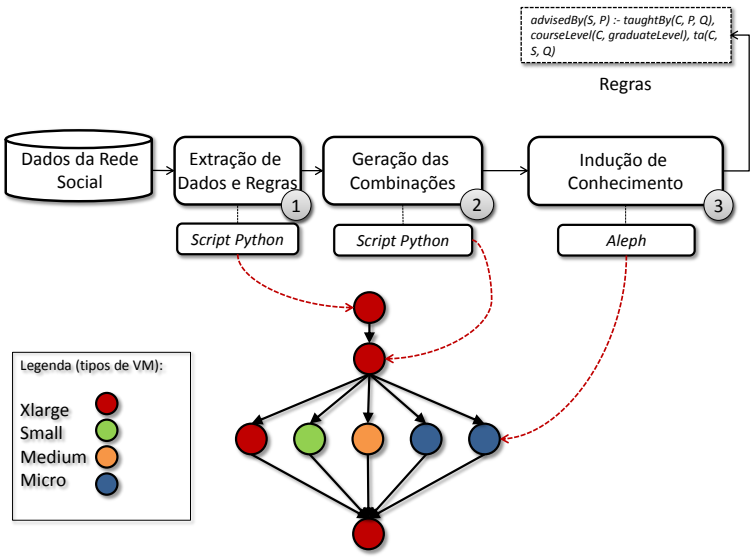

Figura 1 Definição conceitual do *workflow* de MDMR

A primeira atividade do *workflow* (Extração de Dados e Regras) tem como objetivo gerar os arquivos de configuração a serem utilizados na abordagem de MDMR (no caso o Aleph). Essa atividade é executada por meio de um *script* Python que "varre" a estrutura da rede social, identifica e extrai os exemplos positivos, os exemplos negativos e também o conhecimento preliminar. É importante ressaltar que esse *script* deve ser customizado para cada tipo de rede social a ser analisada. Por exemplo, dados do Facebook™ diferem estruturalmente dos dados do Twitter™. Após essa extração, o *script* ajusta essas informações no formato de fatos e/ou regras de forma que possam ser processadas por um interpretador Prolog. Essa atividade não é executada em paralelo pelo SciCumulus, pois não é uma atividade computacionalmente custosa. A segunda atividade do *workflow* é a atividade de geração de combinações de parâmetros. De posse das informações extraídas na primeira atividade, a segunda atividade (Geração das Combinações) analisa quais parâmetros podem ser explorados e qual o domínio de valores associado a cada um deles. Para cada combinação de parâmetro-valor, o SciCumulus gera uma tarefa que será executada em paralelo na terceira atividade (Indução do conhecimento). Na terceira atividade o Aleph será invocado em diversas máquinas virtuais em paralelo, onde cada máquina executará uma combinação de valores de parâmetros diferentes. No exemplo da Figura 1, cada tipo de máquina virtual (na Amazon AWS) é apresentado em uma cor diferente. Assim, as duas primeiras atividades não são paralelizadas e são executadas na máquina do tipo *xlarge*. A terceira atividade é passível

de paralelização e é executada nas máquinas *xlarge*, *medium*, *small* e *micro* (em paralelo), onde cada uma delas consome uma combinação de valores de parâmetros.

Cada atividade no contexto do SciCumulus deve receber como entrada um conjunto de valores de parâmetros (*i.e.* relações) que são processados em paralelo. Na primeira atividade, somente o nome do *dataset* e o caminho devem ser informados. Uma vez que a primeira atividade foi executada e o *dataset* foi convertido para regras Prolog, a segunda atividade é executada, já consumindo os valores de *folds* da validação cruzada, por exemplo. Após a geração das combinações na segunda atividade são geradas todas as combinações a serem processadas na terceira atividade. Uma extração parcial das relações envolvidas em uma execução do *workflow* com o SciCumulus pode ser visualizada na Figura 2. Em adição, na Figura 2 os círculos coloridos ao lado de cada linha da tabela indicam em qual máquina virtual cada uma das combinações foi processada.

Extração de Dados e Regras

NAMEDATASET	PATH
uwcse	/root/exp_Aleph/dataset

Geração das Combinações

NFOLDSE	NFOLDSI	DATASET	SETFOLDS	SETSETS
1	2	uwcse	0	1

Indução de Conhecimento

DATA	TST	TUNE	PARAM	VALPARAM	COM
uwcse	0	1	clauselength	coverage	induce
uwcse	0	2	clauselength	coverage	induce
uwcse	1	1	clauselength	coverage	induce
uwcse	1	2	evalfn	compression	induce
uwcse	1	1	evalfn	compression	induce

Figura 2 Extração das relações em uma execução do *workflow*

Assim, para que o analista defina o *workflow*, ele deve criar um arquivo XML com o formato que possa ser compreendido pelo SciCumulus. Neste arquivo XML necessitamos informar parâmetros de configuração do *workflow* e do ambiente. Por exemplo, na *tag database* o analista informa em qual servidor de banco de dados os dados de execução serão armazenados. Desta forma, podemos recuperar as informações após o término da execução. Na *tag Workflow* o analista informa nome do *workflow* a ser executado e define suas atividades. Cada atividade do *workflow* possui uma *tag Activity* associada com suas relações de entrada (*Relation*) que são compostas de parâmetros (*Field*). De posse do XML modelado, o analista já pode executar o *workflow* e analisar seus resultados. Porém, desta forma o analista deve instanciar as máquinas virtuais e executar o *workflow* "manualmente" de forma *ad hoc*. Para facilitar esse processo, propomos a ferramenta LPFlow4SN apresentada a seguir.

3.3 Arquitetura

A fim de facilitar a execução do processo de MDMR em redes sociais propomos a ferramenta LPFlow4SN. A LPFlow4SN foi desenvolvida baseada no *workflow* apresentado na subseção anterior. Ela possui 4 componentes principais conforme apresentado na Figura 3: (i) o Portal LPFlow4SN, (ii) o Repositório de Proveniência, (iii) o Monitor LPFlow4SN e (iv) o *workflow* modelado no SciCumulus.

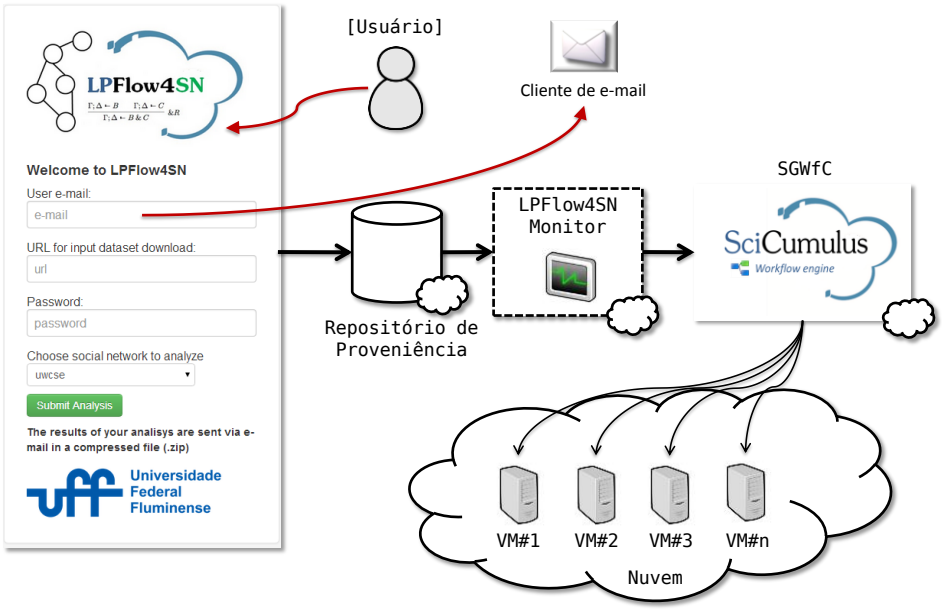

Figura 3 Arquitetura da LPFlow4SN

O portal LPFlow4SN provê a interface entre o usuário analista e o *workflow* a ser executado. Sua interface é bem simples, contendo apenas os campos de e-mail do analista, URL do *dataset* a ser processado, uma senha e a escolha da rede social a ser analisada. Essa senha será usada para descompactar os resultados da execução que serão enviados em um arquivo "*.zip*" por e-mail ao seu término. A rede social escolhida é importante para que o *script* de conversão correto seja invocado na primeira atividade do *workflow*. Uma vez que o analista cadastre uma execução no portal, ela é inserida no Repositório de Proveniência. O repositório de proveniência é composto por tabelas em um banco de dados relacional PostgreSQL que armazenam desde os dados provenientes das execuções passadas do *workflow* até a fila de atendimento de requisições futuras (o atendimento segue a política FIFO – *First In First Out*). O Monitor LPFlow4SN é um serviço residente que é responsável por verificar no repositório de proveniência se existem análises cadastradas que não foram executadas. Caso existam, ele realiza o *download* do *dataset* de entrada (através da URL informada) e inicia uma nova execução do *workflow* utilizando o SciCumulus na nuvem de computadores. Ao término da execução do *workflow* por parte do SciCumulus, o monitor compacta os resultados e envia por *e-mail* para o analista. Após a execução do *workflow*, o analista ainda pode executar as análises *post-mortem* do processo. Essa análise é facilitada com o banco de dados de proveniência que pode ser consultado via consultas SQL. Por exemplo, consideremos que o analista queira saber quais foram os parâmetros avaliados em cada execução de cada atividade do *workflow* e onde se encontram os arquivos binários gerados. Com a seguinte consulta SQL este tipo de análise pode ser realizada rapidamente, o que demandaria muito mais esforço caso não se usasse uma abordagem baseada em *workflows* como base.

```
SELECT a.tag, f.fname, v.textvalue, fil.fname, fil.fdir
FROM hworkflow w, hactivity a, hrelation rin, hfield f, hvalue v, hactivation ac,
hfile fil
WHERE w.wkfid = a.wkfid
AND a.actid = rin.actid
AND f.relidinput = rin.relid
AND v.fieldid = f.fieldid
AND v.actid = a.actid
```

```
AND v.taskid = ac.taskid
AND fil.taskid = ac.taskid
```

4 Avaliação Experimental

Esta seção apresenta os resultados obtidos com a análise de uma rede social acadêmica com a ferramenta e a avaliação do desempenho da ferramenta quanto ao paralelismo oferecido.

4.1 Avaliação dos Resultados da Análise

Nesta seção são apresentadas algumas hipóteses induzidas para o domínio previamente mencionado: uma rede social definida pelo relacionamento de orientação entre alunos e professores do departamento de Ciência da Computação da Universidade de Washington (Richardson e Domingos 2006). O relacionamento de orientação (ou a falta dele) constitui os exemplos positivos (ou negativos), representados pelo predicado *advisedby*. Quando existe mais de uma entidade envolvida, temos um relacionamento. Quando apenas uma entidade estiver listada, temos uma propriedade associada àquela entidade. A base de dados utilizada contém 114 exemplos positivos, 2.438 exemplos negativos e 5.058 fatos lógicos sem variáveis, associados a 10 predicados. Esses predicados constituem os demais relacionamentos entre as entidades professores, cursos, alunos e períodos letivos, bem como propriedades de entidades, como a posição de um professor na universidade.

Para avaliar a qualidade de diferentes hipóteses induzidas a partir de diferentes restrições no processo de refinamento, foram testados parâmetros como *clauselength*, *evalfn* e *noise*. No primeiro caso, os valores do parâmetro variaram de 5 a 10 literais como tamanho máximo de regras induzidas. No segundo caso, foram consideradas as seguintes funções de otimização em cada atividade do *workflow*: "*cobertura*", que, a saber: a função de otimização c, variando o tamanho máximo das regras induzidas de 3 a 6 literais e "*evalfn*", que define a função de otimização a ser utilizada durante o processo de refinamento, onde foram consideradas as seguintes opções: cobertura (diferença entre exemplos positivos e negativos inferidos pela hipótese); compressão (diferença entre a cobertura e a quantidade total de literais em uma hipótese, para obter hipóteses mais compactas); medida F que é a média harmônica da precisão e da sensibilidade e a estimativa M, que computa a utilidade de uma cláusula de acordo com os exemplos positivos cobertos. No terceiro caso, o valor do parâmetro *noise* variou de 0 a 100, de 10 em 10. Todos os demais parâmetros continuaram com seus valores padrão em todas as atividades. Observe que muitos outros parâmetros com diferentes combinações poderiam ser usados para instanciar as atividades do *workflow*. Aqui, discutiremos apenas exemplos de hipóteses obtidas com alguns desses parâmetros.

Selecionamos a atividade com a função de otimização definida como a estimativa F, o tamanho máximo de cláusulas definido como 10 e o parâmetro *noise* definido como 100. Nesse caso, foram obtidas 15 regras, totalizando uma precisão preditiva (medida sobre um conjunto de teste independente) de 85% e sensibilidade de 18%. Os resultados da segunda atividade selecionada foram obtidos com o parâmetro *noise* alterado para o valor 0. Neste último caso, a precisão preditiva foi de 25% e a sensibilidade foi apenas 1%, com uma hipótese de apenas 5 regras. Uma das regras obtidas após a execução da primeira atividade declara que um estudante em fase de pós-qualificação tem uma publicação em conjunto com seu orientador, ou seja:

advisedby(C,A) :- publication(B,A), publication(B,C), phase(C,post_quals). (3)

Por outro lado, na segunda atividade, que requer hipóteses mais específicas e que se mostraram um menor poder de generalização, acrescenta nesta mesma regra mais uma

condição, em que o professor orientador deve estar em certo nível da carreira chamado *faculty_8*, conforme abaixo:

advisedby(C,A) :- publication(B,A), publication(B,C), phase(C,post_quals), position(C,faculty_8). (4)

Esse tipo de regra não é trivial de ser inferida e a análise mostrou que podemos extrair conhecimento tácito das redes sociais com a abordagem proposta. A utilização de várias combinações de parâmetros para compor o processo experimental e conseguinte análise das hipóteses induzidas apenas se mostra uma tarefa possível com o uso de SGWfC. Caso contrário, o processo e a análise dos dados seriam feitos de forma manual, o que acarreta um gasto extraordinário de tempo do especialista e uma maior propensão a erros.

4.2 Avaliação do Desempenho

Para realizar a avaliação do desempenho da LPFlow4SN realizamos o *deploy* da ferramenta na nuvem da Amazon AWS em 4 máquinas virtuais do tipo *large* (EC2 ID: m1.large – 7.5 GB RAM, 850 GB, 2 núcleos) totalizando 16 núcleos de processamento. Assim, utilizamos o mesmo *dataset* apresentado na subseção anterior como dado de entrada. Esse *dataset* gerou 89 combinações de parâmetros que foram executadas em paralelo nessas máquinas virtuais. A avaliação foi realizada medindo-se o tempo necessário para a execução da análise para 1, 2, 4, 8 e 16 núcleos de processamento conforme apresentado na Figura 4.

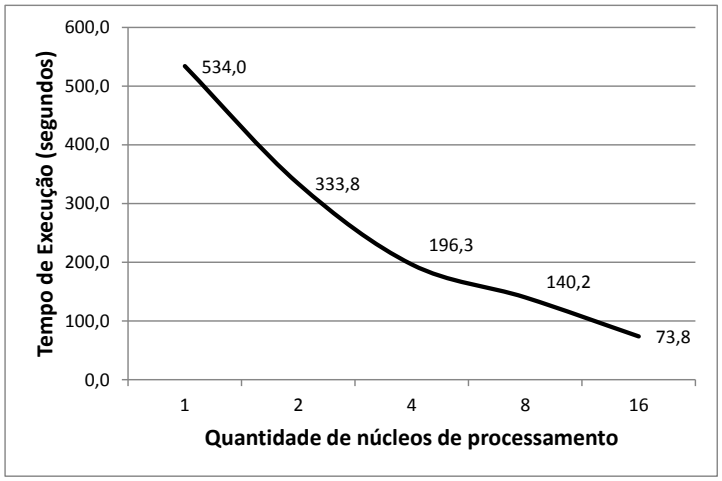

Figura 4 Tempo de execução do LPFlow4SN na Amazon AWS

Não houve necessidade de alterar o código do Aleph, uma vez que o SciCumulus replica o programa nas diversas máquinas virtuais. Podemos perceber que houve uma redução de 87% no tempo de execução com 16 núcleos se comparado ao tempo de execução em um único núcleo, o que é um ganho significativo. Este teste ainda pode ser considerado de pequena escala, uma vez que análises de redes sociais mais complexas podem gerar mais combinações a serem avaliadas o que vai demandar mais poder computacional para que o resultado seja produzido em um tempo viável.

5. Considerações Finais

Este artigo apresentou a ferramenta LPFlow4SN que tem por objetivo inferir conhecimentos em redes sociais através de processos de MDMR. A modelagem do processo de MDMR é realizada utilizando técnicas de ILP associadas ao conceito de *workflow* científico para a modelagem e execução estruturada do processo. A experiência apresentada neste artigo nos mostrou a adequação de um processo de MDMR em redes sociais como um *workflow*

científico, usufruindo das vantagens já oferecidas pelos SGWfC existentes. Como esse processo pode ser computacionalmente custoso devido à exploração de valores de parâmetros em larga escala, optou-se por utilizar o SGWfC SciCumulus como base para o LPFlow4SN já que o mesmo oferece capacidades de execução paralela além da análise dos resultados *a posteriori*. Os resultados mostram que para atingirmos uma solução genérica para a análise de redes sociais com a ferramenta LPFlow4SN, ainda são necessários diversos ajustes e melhorias na implementação. Contudo, o objetivo de oferecer uma maneira simples e estruturada de executar um processo complexo de MDMR e ainda melhorar o tempo de execução de processos de MDMR muito demorados, por especialistas que não são especialistas em processamento paralelo foi atendido, particularmente apresentando o cenário de exploração de parâmetros. Os resultados experimentais mostram que conseguimos inferir regras não triviais na rede utilizada como estudo de caso e ainda obtivemos ganho de desempenho à medida que processadores foram adicionados. Esse ganho recebe o valor agregado ao registro e análise dos dados de proveniência que podem ser realizados via consultas SQL simples.

Referências

Bakshy, E., Rosenn, I., Marlow, C., Adamic, L., (2012), "The Role of Social Networks in Information Diffusion". In: *Proceedings of the 21st International Conference on World Wide Web*, p. 519–528, New York, NY, USA.

Džeroski, S., Lavrač, N., (2001), *Relational data mining*. Berlin; New York, Springer.

Freire, J., Koop, D., Santos, E., Silva, C. T., (2008), "Provenance for Computational Tasks: A Survey", *Computing in Science and Engineering, v.10*, n. 3, p. 11–21.

Han, J., Kamber, M., Pei, J., (2011), *Data Mining: Concepts and Techniques, Third Edition*. 3 ed. Morgan Kaufmann.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Murta, L., Ogasawara, E., Oliveira, D., Cruz, S. M. S. da, Martinho, W., (2010), "Towards Supporting the Life Cycle of Large-scale Scientific Experiments", *International Journal of Business Process Integration and Management*, v. 5, n. 1, p. 79–92.

Muggleton, S., (1995), "Inverse entailment and progol", *New Generation Computing*, v. 13, n. 3-4 (Dec.), p. 245–286.

Nilsson, U., Małuszyński, J., (1995), *Logic, programming, and Prolog*. Chichester; New York, John Wiley.

Oliveira, D., Ogasawara, E., Baião, F., Mattoso, M., (2010), "SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows". In: *3rd International Conference on Cloud Computing*, p. 378–385, Washington, DC, USA.

Powers, D., (2007), *{Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation}*

Raedt, L., (2008), "Logical and Relational Learning". In: *Proceedings of the 19th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, p. 1–1, Berlin, Heidelberg.

Richardson, M., Domingos, P., (2006), "Markov Logic Networks", *Mach. Learn.*, v. 62, n. 1-2 (Feb.), p. 107–136.