Revisando Teorias Lógicas de Primeira-ordem a partir de Exemplos usando Busca Local Estocástica

Aline Paes[1] **, Gerson Zaverucha**[1] **, Vítor Santos Costa**[1]

[1]Programa de Engenharia de Sistemas e Computação – COPPE
Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68511 – 21945-970 – Rio de Janeiro – RJ – Brasil

{ampaes,gerson,vitor}@cos.ufrj.br

Abstract. *Theory Revision in First Order Logic (FOL) is the process of improving user-defined or automatically generated theories using a set of examples. So far, the usefulness of Theory Revision systems has been limited by the cost of searching the huge search spaces they generate. This is a general difficulty when learning FOL theories but recent work showed that Stochastic Local Search (SLS) techniques may be effective, at least when learning FOL theories from scratch. Motivated by these results, we propose novel SLS based search strategies for Theory Revision from exemplos in FOL. Experimental results show that introducing stochastic search does lead to better runtime performance and does not decrease accuracy.*

Resumo. *Revisão de teorias lógicas de Primeira-ordem (LPO) é o processo de melhorar teorias geradas automaticamente ou definidas pelo usuário, dado um conjunto de exemplos. A utilização de sistemas de revisão de teorias tem sido limitada pelo custo de buscar em um grande espaço de hipóteses. Esta é uma dificuldade geral no aprendizado de teorias LPO, mas trabalhos recentes mostram que técnicas de busca local estocástica (SLS) são efetivas ao menos quando as teorias são aprendidas a partir do zero. Motivados por estes resultados, propomos neste artigo estratégias de busca baseadas em SLS para revisar teorias LPO a partir de exemplos. Resultados experimentais mostram que utilizar SLS conduz a um tempo de execução melhor sem diminuir a acurácia.*

1. Introdução

A lógica de primeira-ordem é um dos formalismos mais utilizados para representar conhecimento devido a habilidade de representar indivíduos, suas propriedades e relacionamentos de uma forma compacta enquanto ainda permite inferência automática. Uma grande variedade de sistemas de Programação em Lógica Indutiva (ILP) tem sido desenvolvidos para automaticamente aprender teorias lógicas de primeira-ordem, com bons resultados em um grande número de aplicações. A maioria dos sistemas são projetados para aprender teorias a partir do zero, dado um conjunto de exemplos e um conhecimento preliminar

fixo. Existem relativamente menos trabalhos aplicados ao problema de *reparar* teorias incorretas ou incompletas. Exemplos de teorias que poderiam ser reparadas ou melhoradas são aquelas definidas por um especialista no domínio da aplicação, que devem incluir informação útil mas incompleta, assumindo suposições incorretas, ou até mesmo ser inconsistente. Outro exemplo comum ocorre quando novos exemplos não podem ser explicados pela teoria original. Em tais casos os sistemas de ILP ou descartariam a teoria inicial ou a considerariam como parte do conhecimento preliminar que não pode ser modificado. Uma vez que a tarefa de aquisição de conhecimento é difícil e consome muito tempo, e já que a teoria original pode conter informação importante, poderíamos tirar vantagem da teoria inicial como um ponto de partida para o processo de aprendizado. Idealmente esta abordagem deveria diminuir o tempo do aprendizado e resultar em teorias mais acuradas. Vários sistemas de *refinamento de teorias* tem sido propostos objetivando estas metas [Wogulis and Pazzani 1993, Richards and Mooney 1995]. Tais sistemas assumem que a teoria inicial é aproximadamente correta. Assim, somente alguns *pontos* na teoria fazem com que a mesma não modele corretamente o conjunto de exemplos. A idéia portanto é que deve ser mais eficiente buscar por tais pontos na teoria e *revisá-los* do que usar um algoritmo que aprende uma nova teoria a partir do zero. Observe que esses algoritmos de revisão podem ser vistos como uma generalização do aprendizado a partir do zero, como executado pelos sistemas de ILP. Entretanto, em contraste com a maioria dos sistemas de ILP, os algoritmos de revisão de teoria não usam a abordagem de separar–e–conquistar, ao invés eles executam uma busca no espaço de teorias inteiras. A abordagem de separe–e–conquiste, que busca por cláusulas que expliquem exemplos não cobertos, freqüentemente gera hipóteses desnecessariamente longas, com muitas cláusulas.

Sistemas de revisão de teoria operam buscando por *pontos de revisão*, isto é, os pontos que classificam incorretamente algum exemplo e propondo revisões para esses pontos usando *operadores de revisão*. Portanto, revisão de teoria pode ser vista como um processo de busca e assim como em ILP padrão, a revisão de programas lógicos a partir de exemplos busca em um grande espaço de hipóteses, implicando em grandes requerimentos de tempo e armazenamento. Tais espaços de busca crescem rapidamente com o tamanho do conhecimento preliminar e com teorias contendo muitas falhas. Por último, sistemas de revisão de teoria são particularmente ambiciosos já que as hipóteses sao teorias inteiras, o que é um problema reconhecidamente difícil [Bratko 1999]. Portanto, para melhorar os requerimentos da busca, os sistemas de revisão de teoria podem se beneficiar de estratégias de busca mais eficientes, tais como estratégias de busca local estocástica (SLS). Tais métodos têm sido aplicados com sucesso para resolver problemas de otimização combinatória representados de forma proposicional [Rückert and Kramer 2003, Selman et al. 1992, Selman et al. 1996] e recentemente também têm sido aplicados para aprender teorias de primeira-ordem a partir do zero [Paes et al. 2007, Železný et al. 2006], melhorando substancialmente a eficiência da busca em ambos os domínios. Motivados por estes trabalhos e pela explosão combinatória da busca na teoria inteira nós investigamos nesse artigo a relevância de aplicar

SLS em algoritmos de revisão de teoria. Para tanto, nós desenvolvemos uma família de estratégias que executam busca estocástica e a comparamos a um algoritmo que executa somente busca *hill-climbing*.

O artigo está organizado da seguinte forma: nas seções 2 e 3 é fornecido um conhecimento preliminar abordando revisão de teoria e busca local estocástica, respectivamente. As estratégias estocásticas de revisão de teorias são desenvolvidas na seção 4. Antes de concluir o trabalho, resultados experimentais são apresentados na seção 5.

2. Revisão de teorias de primeira-ordem a partir de exemplos

A tarefa de *revisão de teorias* envolve mudar o conjunto de respostas da teoria fornecida, i.e., melhorar suas capacidades de inferência, adicionando respostas perdidas anteriormente através de *generalização* ou removendo respostas incorretas através de *especialização* [Wrobel 1996]. Um sistema de revisão de teoria recebe uma teoria inicial e um conjunto de exemplos. Esta teoria inicial é composta por dois elementos: um componente invariante, o conhecimento preliminar (*background knowledge*), e um componente que pode ser modificado. O conjunto de exemplos é dividido em exemplos positivos e exemplos negativos. Assim, o processo de revisão deve gerar uma teoria final que prove todos os exemplos positivos e nenhum dos exemplos negativos, retornando então uma teoria consistente com o conjunto de dados. Veja que aprender em ILP pode ser visto como um caso particular de revisão de teoria onde o componente que pode ser modificado é vazio.

2.1. Pontos de revisão

Várias cláusulas na teoria podem estar envolvidas na prova de um exemplo negativo assim como várias cláusulas podem ser generalizadas para fazer com que um exemplo positivo passe a ser provado. Podemos então definir dois tipos de pontos de revisão: *pontos de revisão de generalização*, que são literais em uma cláusula responsável pela falha de um exemplo positivo e outros antecedentes que possam ter contribuído para esta falha e *pontos de revisão de especialização* que sao cláusulas usadas na prova de exemplos negativos. O tipo do ponto de revisão determina o operador que precisa ser aplicado no mesmo para tornar a teoria consistente.

2.2. Operadores de revisão

A princípio qualquer operador usado em aprendizado de máquina de primeira-ordem pode ser usado em um sistema de revisão de teoria. Os principais operadores, definidos originalmente em [Richards and Mooney 1995] são: operadores de especialização de *Exclusão de regra*, que remove uma cláusula que participe com sucesso da prova de um exemplo negativo e de *Adição de antecedentes*, que adiciona antecedentes em uma cláusula incorreta; e os operadores de generalização de *Exclusão de antecedentes* que exclui antecedentes que falharam ao se tentar provar exemplos positivos e de *Adição de regra* que deixa a cláusula original na teoria e gera novas baseadas nesta através da exlusão e adição de antecedentes.

2.3. FORTE

Um exemplo de sistema de revisão de teoria é o FORTE (*First Order Revision of Theories from examples*) [Richards and Mooney 1995], cujo algoritmo está reproduzido no Algoritmo 1. FORTE executa uma busca *hill-climbing* no espaço de operadores de generalização e especialização na tentativa de encontrar uma revisão que torne a teoria consistente com os exemplos de treinamento. FORTE funciona da seguinte maneira: primeiro, ele identifica todos os pontos de revisão na teoria corrente e depois ele gera um conjunto de revisões propostas para cada ponto de revisão iniciando daquele com o potential mais alto. *Potencial* é definido como o número de exemplos incorretamente classificados que poderiam se tornar corretamente classificados a partir de uma revisão naquele ponto. Depois de gerada, cada revisão recebe uma *pontuação*, definida como o aumento real na acurácia da teoria. A melhor revisão gerada até o momento é retida. FORTE para de gerar revisões quando o potencial da próxima revisão é menor do que a pontuação da melhor revisão encontrada. Se a melhor revisão realmente melhora a acurácia da teoria, ela é implementada. Este processo iterativo continua até que nenhuma revisão melhore a teoria.

Algorithm 1 Algoritmo FORTE [Richards and Mooney 1995]

```
repita
   gera pontos de revisão;
   ordena pontos de revisão por potencial (do mais alto para o mais baixo);
   para cada ponto de revisão
       gera revisões;
       atualiza melhor revisão encontrada;
   até que o potencial da próxima revisão seja menor que a pontuação da melhor revisão atualizada
   se a melhor revisão melhora a teoria
      implementa a melhor revisão
até que nenhuma revisão melhore a teoria;
```

3. Busca Local Estocástica

Algoritmos de busca local estocástica abandonam a otimalidade em favor de boas soluções, alcançadas usando recursos limitados [Rückert and Kramer 2003]. Esta meta é seguida com sucesso por métodos de checagem da satisfabilidade de fórmulas proposicionais tais como GSAT [Selman et al. 1992] e WalkSAT [Selman et al. 1996]. Eles também têm sido aplicados em tarefas proposicionais codificadas como um problema de satisfabilidade, melhorando substancialmente a eficiência [Chisholm and Tadepalli 2002, Rückert and Kramer 2003]. A próxima seção traz uma breve revisão desses algoritmos.

GSAT e WalkSAT GSAT é baseado em um procedimento guloso com uma componente estocástica. Ele busca por uma associação de veracidade que satisfaz um conjunto de cláusulas proposicionais. O algoritmo GSAT básico inicia com uma associação

de valores gerada aleatoriamente e então repetidamente troca o valor de uma variável que conduza a maior diminuição no número de cláusulas insatisfatíveis, até que ou uma associação satisfatível seja encontrada ou um número máximo de trocas seja alcançado. GSAT pode facilmente ser pego em um mínimo local e a única forma de escapar é reiniciar com uma nova associação gerada aleatoriamente depois que o número máximo de trocas seja alcançado. Este processo é repetido até alcançar um número máximo pré fixado de tentativas.

Outro mecanismo para escapar de mínimos locais é aleatoriamente alternar entre movimentos gulosos e estocásticos, randomicamente selecionado a partir de variáveis que aparecem em cláusulas insatisfeitas. Portanto, o algoritmo WalkSAT primeiro pega uma cláusula não satisfeita pela associação corrente e então pega uma variável dessa cláusula para alterar. Com probabilidade p a variável é escolhida aleatoriamente e com probabilidade $1 - p$ é utilizada uma heurística gulosa.

Existe ainda um estudo recente [Železný et al. 2006] que adaptou uma família de estratégias de busca estocásticas no sistema Aleph [Srinivasan 2001], incluindo buscas baseadas no GSAT, WalkSAT e reinício aleatório rápido. Por questões de espaço não discutiremos estas estratégias neste trabalho.

4. Revisão de teorias de primeira-ordem a partir de exemplos usando uma abordagem estocástica

Com o objetivo de diminuir o tempo de execução dos sistemas de revisão de teorias lógicas de primeira-ordem, principalmente quando muitas revisões são propostas e também almejando escapar de máximos locais, nós propomos a introdução de um componente estocástico no algoritmo de busca de revisões. Embora este componente possa ser introduzido em qualquer algoritmo de revisão de teoria, nós o introduzimos no algoritmo FORTE para implementar e comparar experimentalmente os resultados da nossa abordagem. Similar ao WalkSAT, nossa abordagem executa uma busca local com caminhadas aleatórias, alternando entre movimentos estocásticos e gulosos, com o tipo do movimento a ser executado sendo escolhido de acordo com uma probabilidade fixa p. Nós investigamos nesse trabalho duas abordagens de busca local estocástica. A primeira considera uma busca local estocástica gulosa e como tal tem seu tempo de execução sendo uma variável aleatória. Por causa disso, o algoritmo estocástico guloso é implementado para parar depois que um número máximo de passos tenha sido executado ou quando alcançar uma pontuação máxima. A outra abordagem executa uma busca local estocástica *hill-climbing*. Assim, diferente da abordagem gulosa, mesmo quando a próxima hipótese é escolhida aleatoriamente esta deve pontuar melhor do que a hipótese corrente. O critério de parada também é diferente, com o algoritmo parando sua execução quando nenhuma revisão proposta puder melhorar a pontuação corrente.

Uma diferença importante entre a abordagem do presente trabalho e outros métodos estocásticos é que aqui a hipótese inicial não é inicializada aleatoriamente, já que o ponto de partida é a teoria inicial fornecida ao sistema.

4.1. Dois algoritmos estocásticos de revisão de teoria

Os algoritmos de revisão de teoria usando busca local estocástica gulosa e *hill-climbing*, denominados daqui por diante como SLS_guloso e SLS_hill_climbing, podem ser vistos nos Algoritmos 2 e 3 respectivamente. Ambos iniciam gerando todos os pontos de revisão a partir da teoria inicial e do conjunto de exemplos positivos e negativos, da mesma maneira executada pelo FORTE. Então eles devem escolher entre o movimento estocástico ou guloso. Assim, com uma certa probabilidade p, a revisão a ser implementada é escolhida aleatoriamente a partir de uma lista de todas as possíveis revisões. É importante observar que quando o movimento é estocástico as revisões não são explicitamente geradas, elas são somente listadas a partir dos tipos de pontos de revisão. Portanto, cada ponto de especialização contribuirá com duas possíveis revisões para a lista, referentes aos dois operadores de especialização e cada ponto de generalização contribuirá com duas possíveis revisões para a lista, referentes aos dois operadores de generalização. Logo, a lista será composta por todos os tipos de possíveis revisões que poderiam ser geradas a partir dos pontos de revisão. Para entender melhor como a lista é montada, suponha que a teoria contenha, entre outras, a cláusula $active(A)$:-$logp(A, B), atm(A, C, h, 3, D)$, obtida a partir do *benchmark* de ILP Mutagenesis [Srinivasan et al. 1996]. Agora suponha que apenas esta cláusula na teoria tenha classificado incorretamente tanto exemplos negativos como positivos, sendo então considerada um ponto de revisão de especialização e de um ponto de revisão de generalização. Assim, a lista terá quatro elementos, correspondentes a uma futura aplicação dos operadores de exclusão de regra e adição de antecedentes e ainda adição de cláusulas e exclusão de antecedentes nessa cláusula. Não gerar as revisões antes de escolher aquela que será implementada diminui o tempo de execução do processo de revisão. Depois de escolher um elemento aleatório da lista, a revisão correspondente a esse elemento é gerada e implementada se possível. Se não for possível implementar a revisão (por exemplo, a cláusula não pode ser excluída pois é a única explicando um conceito), outro elemento da lista é escolhido aleatoriamente.

Com probabilidade $1-p$, os algoritmos estocásticos funcionam similar ao FORTE. Como tal, a revisão com a mais alta pontuação é escolhida para ser implementada a partir de um conjunto de todas as possíveis revisões geradas para os pontos de revisão. A geração das revisões inicia naquela com o maior potencial e estas param de ser geradas quando o potencial do proximo ponto de revisão for menor do que a pontuação da melhor revisão.

No SLS_guloso, a revisão é implementada sem verificar se ela melhora a pontuação corrente. As estratégias estocásticas investigadas em [Železný et al. 2006] também procedem de maneira similar, uma vez que elas não podam os elementos que não podem ser refinados de maneira que tenham uma pontuação melhor do que a corrente. O procedimento é executado um número máximo de passos ou até alcançar uma pontuação máxima.

SLS_hill_climbing difere do SLS_guloso em dois momentos. Primeiro, quando o movimento é estocástico ele gera a revisão escolhida aleatoriamente e então verifica se

Algorithm 2 Algoritmo SLS_guloso

```
enquanto pontuação < pontuaçãoMáxima e passos < maxPassos
  gera pontos de revisão;
  com probabilidade p
    para cada ponto de revisão
      lista todas as possíveis revisões;
    proximaRevisão ← uma revisão escolhida aleatoriamente a partir da lista de possíveis revisões;
  caso contrário
    ordena pontos de revisão por potencial (do mais alto para o mais baixo);
    para cada ponto de revisão
      gera revisões;
      atualiza melhor revisão encontrada;
    até que o potencial da próxima revisão seja menor que a pontuação da melhor revisão atualizada
     proximaRevisão ← melhor revisão encontrada;
  implementa proximaRevisão;
  passos ← passos + 1;
```

sua pontuação é melhor do que a pontuação corrente. Se for, a revisão é implementada. Se não for, outra revisão é escolhida até que alguma melhore a pontuação ou a lista não tenha mais elementos. Quando o movimento é guloso, também é requerido que a revisão a ser implementada melhore a teoria, assim como no FORTE. A outra diferença é em relação ao critério de parada. Como declarado anteriormente, o procedimento SLS_hill_climbing para quando não existirem mais revisões que possam melhorar a pontuação corrente, ou seja, o critério de parada é igual ao do algoritmo FORTE original.

5. Resultados experimentais

Para validar os algoritmos de busca estocástica desenvolvidos, utilizamos um benchmark de ILP, o Mutagenesis [Srinivasan et al. 1996]. Uma teoria inicial foi obtida pelo sistema Aleph e erros como alteração de variáveis e exclusão de regras entre outros foram introduzidos na mesma, de forma que a teoria fornecida ao sistema de revisão fosse aproximadamente correta.

Para verificar se o tempo de execução do processo de revisão bem como a acurácia final podem ser melhoradas com técnicas de busca local estocástica, executamos o FORTE e ambas as abordagens estocásticas discutidas acima. Queremos verificar ainda qual das duas abordagens estocásticas possui melhor desempenho. As probabilidades de escolha dos movimentos variaram de 30% a 70% e, para cada probabilidade, o número máximo de passos no SLS_guloso foi fixado em 4 a 10 passos, variando de dois em dois. Os algoritmos estocásticos foram rodados 5 vezes utilizando validação cruzada com 10 *folds* [Dietterich 1998]. Sabemos que o ideal é executar algoritmos estocásticos cerca de 25 vezes, mas deixaremos tal tarefa para trabalhos futuros por questões de tempo. As acurácias finais alcançadas pelo FORTE e pelos algoritmos estocásticos considerando o conjunto de teste e o tempo que estes sistemas levaram para executar são exibidos nos gráficos contidos na Figura 1. Vale lembrar que exibimos as **médias** de acurácia e tempo

Algorithm 3 Algoritmo SLS_hill_climbing

```
repita
   gera pontos de revisão;
   com probabilidade p
      para cada ponto de revisão
         lista todas as possíveis revisões;
      proximaRevisão ← uma revisão escolhida aleatoriamente a partir da lista de possíveis revisões cuja
                       pontuação > pontuação corrente;
   caso contrário
      ordena pontos de revisão por potencial (do mais alto para o mais baixo);
      para cada ponto de revisão
         gera revisões;
         atualiza melhor revisão encontrada;
      até que o potencial da próxima revisão seja menor que a pontuação da melhor revisão atualizada
       proximaRevisão ← melhor revisão encontrada;
       se proximaRevisão melhora a teoria corrente
         implementa proximaRevisão;
até que nenhuma revisão melhore a teoria
```

dos folds das 5 rodadas, neste último caso obviamente apenas para as abordagens estocásticas.

Na Figura 1(a) é possível observar que os maiores valores de acurácia são alcançados quando a probabilidade é fixada em 50%. Esse resultado corresponde ao estudo realizado em [Železný et al. 2006] visto que naquele trabalho a probabilidade da estratégia WalkSAT era sempre fixa em 50%. Valores maiores de probabilidade alcançam uma acurácia bem menor no conjunto de testes, embora o tempo de execução também diminua. Com relação aos diferentes números máximos de passos no SLS_guloso não existe muita diferença no tempo de execução, com os menores valores ocorrendo quando o número de passos é fixado em 6, o que pode indicar que para essa base e para a teoria inicial fornecida, é difícil implementar mais do que 5 revisões. A acurácia entretanto geralmente diminui conforme o número máximo de passos aumenta, o que pode ser um indicativo de *overfitting*.

Podemos observar ainda pelo gráficos que a acurácia não sofre muitas alterações nas três abordagens. Com probabilidade fixada em 50% e número máximo de passos igual a 6 no SLS_guloso, os valores médios alcançados nesta medida são $77,85$ no SLS_hill_climbing e $75,87$ no SLS_guloso. Embora a acurácia alcançada no FORTE seja ligeiramente maior, com o valor de $79,96$, a diferença entre ela e ambas as abordagens estocásticas não é estatisticamente significante, com nível de significância de $p < 0,05$.

Considerando o tempo de execução do processo de revisão podemos observar no gráfico 1(b) que enquanto as abordagens estocásticas não exibiram diferenças significantes de performance mesmo para valores diferentes de probabilidade, o mesmo não se pode dizer da diferença de tempo entre o FORTE e as abordagens estocásticas. A

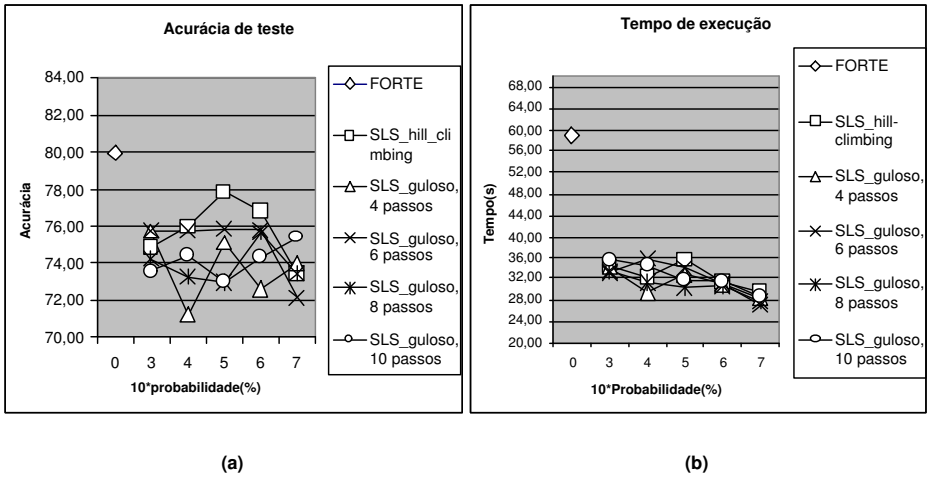

Figura 1. (a) Acurácias alcançadas pelo algoritmo FORTE, SLS_guloso e SLS_hill_climbing no conjunto de teste (b) Tempo de execução do algoritmo FORTE, SLS_guloso e SLS_hill_climbing

diferença de tempo foi reduzida de 60 segundos para aproximadamente 35 segundos com a probabilidade de escolha de movimento sendo 50%, sendo uma diferença de tempo estatisticamente significante considerando $p < 0,05$ como nível de significância. Principalmente por esse último resultado podemos concluir que, embora os resultados sejam para apenas uma importante base, realmente vale a pena introduzir estratégias de busca local estocástica quando estamos revisando teorias lógicas de primeira-ordem a partir de exemplos.

6. Conclusões

Nesse artigo dois algoritmos de busca local estocástica foram desenvolvidos para a revisão de teorias lógicas de primeira-ordem a partir de exemplos: uma abordagem SLS gulosa e uma abordagem SLS *hill-climbing*. Ambos se baseiam no sistema FORTE. Logo, o algoritmo FORTE original e as abordagens estocásticas foram comparados através de um benchmark muito utilizado de ILP, o Mutagênesis. Nesta comparação foi possível observar que o tempo de execução é significativamente diminuído quando o algoritmo é estocástico enquanto a acurácia, apesar de diminuir, não apresenta diferença estatisticamente significante. Entre os algoritmos estocásticos não foi possível eleger um ”melhor”, pois ambos alcançam valores de medidas relativamente iguais.

Os algoritmos dos operadores de revisão que escolhem o antecedente a ser adicionado ou excluído também podem se beneficiar das técnicas de busca estocástica. Isso será abordado em trabalhos futuros por questões de espaço. Ainda como trabalho futuro, pretendemos realizar experimentos mais extensivos com outras bases e comparar as abordagens estocásticas desenvolvidas neste artigo aprendendo teorias do zero com as estratégias de busca estocásticas implementadas no sistema Aleph.

Agradecimentos

O primeiro autor é financiado pela CAPES e os demais pelo CNPq. Vítor Santos Costa foi parcialmente suportado por fundos concedidos ao LIACC pela Fundação para a Ciência e Tecnologia, Program POSI. Agradecemos a Bradley Richards and Raymond Mooney por tornarem o sistema FORTE disponível e a Kate Revoredo por participar de algumas discussões a respeito deste trabalho.

Referências

Bratko, I. (1999). Refining complete hypotheses in ILP. In *Proc. of the 9th ILP*, volume 1634 of *LNAI*, pages 44–55. Springer.

Chisholm, M. and Tadepalli, P. (2002). Learning decision rules by randomized iterative local search. In *Proc. of the 19th ICML*, pages 75–82.

Dietterich, T. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1924.

Paes, A., Železný, F., Zaverucha, G., Page, D., and Srinivasan, A. (2007). Ilp through propositionalization and stochastic k-term DNF learning. In *to appear in Proc. of the 16th ILP*, volume 4455 of *LNAI*. Springer.

Richards, B. L. and Mooney, R. J. (1995). Automated refinement of first-order horn-clause domain theories. *Machine Learning*, 19(2):95–131.

Rückert, U. and Kramer, S. (2003). Stochastic local search in k-term DNF learning. In *Proc. of the 20th ICML*, pages 648–655.

Selman, B., Kautz, H. A., and Cohen, B. (1996). Local search strategies for satisfiability testing. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, October 11-13, 1993. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 26:521–532.

Selman, B., Levesque, H. J., and Mitchell, D. G. (1992). A new method for solving hard satisfiability problems. In *Proc. of the 10th AAAI*, pages 440–446.

Srinivasan, A. (2001). *The Aleph Manual*. http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph.html.

Srinivasan, A., Muggleton, S., Sternberg, M. J. E., and King, R. D. (1996). Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, 85(1-2):277–299.

Železný, F., Srinivasan, A., and Page, D. (2006). Randomised restarted search in ILP. *Machine Learning*, 64(1-3):183–208.

Wogulis, J. and Pazzani, M. (1993). A methodology for evaluationg theory revision systems: results with Audrey II. In *Proc. of 13th IJCAI*, pages 1128–1134.

Wrobel, S. (1996). First-order theory refinement. In Raedt, L. D., editor, *Advances in Inductive Logic Programming*, pages 14–33. IOS Press.