QXD0037 - Inteligência Artificial

Terceiro Trabalho Prático

Profa. Dra. Viviane Menezes
Cortesia do Prof. Marcelo Finger

Data da Entrega: 15.11.2017

1 Objetivos

Os objetivos deste trabalho são três:

- Que o aluno se familiarize com programas provadores de teoremas.
- Que o aluno venha a conhecer a codificação automática de problemas em lógica proposicional.
- Que o aluno se acostume a expor suas ideias por escrito.

2 Regras

1. Espera-se que este trabalho seja feito em equipes de 3 alunos. [1]
2. Cada equipe entregará um relatório através do Moodle pela conta de um dos membros da equipe.
3. Deve ser entregue um único arquivo compactado (formato zip ou tgz), contendo a implementação pedida, a codificação do(s) problema(s) indicados e um relatório sobre o trabalho.
4. O programa deve ser escrito em uma das seguintes linguagens: C, C++, Java, Perl, Python. Só essas. Mais nenhuma outra. Mesmo.
5. Descontar-se-á por erros ortográficos, gramaticais, desvios de estilo e outros impropérios à língua.

[1] Para fins desta disciplina, as equipes de três pessoas contém exatamente 3 pessoas, não existindo instâncias com número superior a 3 pessoas, inferior a 3 pessoas, ou um número qualquer fracionários de pessoas. :-)

3 O Problema SAT

No xadrez, a rainha é tida como peça mais importante do tabuleiro devido a sua versatilidade de ataques. O problema que iremos resolver trata de como posicionar N rainhas num tabuleiro de tamanho $N \times N$, sem que uma rainha ataque as demais.

Uma rainha pode atacar outras que estejam na mesma linha, na mesma coluna ou nas mesmas diagonais, conforme ilustrado abaixo.

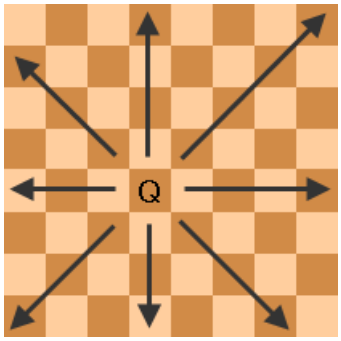

Figura 1: Rainha atacando posições ameaçadas

A idéia é codificar este problema como uma fórmula da lógica proposicional, tal que cada valoração que satisfaz a fórmula representa uma disposição de rainhas no tabuleiro. Sabe-se que este problema possui soluções para $N \geq 4$.

Uma das formas de se fazer isso (mas nem de perto a única) é termos N^2 átomos da forma p_{ij}, com $1 \leq i, j \leq N$. Se tivermos uma valoração v tal que $v(p_{ij}) = 1$, isto representa que há uma rainha na linha i e na coluna j do tabuleiro $N \times N$.

Para a primeira parte do trabalho você deverá realizar os seguintes passos:

(a) Projetar uma fórmula que represente o problema das N-rainhas tal que uma valoração represente uma solução, conforme acima.

(b) Escrever um programa `nrainhas` que recebe como entrada um número $N \geq 4$ e como saída gera o arquivo com a fórmula que representa o problema.

(c) A chamada ao programa deve ser feita de forma clara, da forma

```
$ nrainhas 10
```

para gerar uma fórmula que representa as N-rainhas num tabuleiro 10×10

Nota: esta fórmula deve estar na *forma clausal*, representada de acordo com as convenções do formato `cnf` no formato aceito pelo resolvedor SAT.

4 O Experimento

Um resolvedor SAT é um programa que recebe uma fórmula proposicional e decide se ela é satisfazível ou não. Em geral, ao decidir se é satisfazível, ele também apresenta uma valoração.

Para nosso experimento iremos utilizar um resolvedor SAT de alta potência, o zchaff. O código fonte para este resolvedor pode ser obtido em

`http://www.princeton.edu/~chaff/zchaff.html`

Ao baixar o programa, você será solicitado a dizer as razões pelas quais você está baixando o programa. Você pode escrever o que quiser aí, mas o correto é dizer *coursework for an undergraduation course at the Federal University of Ceara*. Após baixar o pacote, você deve descompactá-lo e compilá-lo segundo as instruções, bastando usar o comando *make* no diretório extraído. Você precisará ter o compilador *g++* instalado.

O zchaff gera uma série de estatísticas, sendo que para nós as mais importantes são o tempo de execução e o número de nós da árvore de prova.

O objetivo do experimento é levantar a curva de performance da resolução do problema das N-rainhas, e plotar um gráfico do tempo de execução (ou número de nós da árvore de prova) em função de N. Note que o número de átomos da fórmula é N^2, portanto talvez você prefira plotar o tempo em função de N^2.

Para gerar este gráfico você deverá ter executado pelo menos 10 problemas para distintos valores de N. Obviamente, estes problemas devem ter sido geradas pelo seu programa desenvolvido na etapa anterior. O valor mínimo de N é 5. O valor máximo de N é tal que a execução não ultrapasse um ou dois minutos (60s–120s).

Note que para este experimento não importa a solução gerada, só o tempo gasto na sua geração. As estatísticas importantes para nós são chamadas de *Num. of Decisions* (tamanho da árvore de prova) e *Total Run Time* (tempo de prova).

Para esta segunda etapa, pede-se gerar um relatório de no máximo 5 páginas (no formato pdf) que cubra os seguintes tópicos:

(a) Contenha um título e o(s) nome(s) do(s) autor(es).

(b) Contenha uma introdução, descrevendo o experimento, incluindo as plataformas de hardware e software em que o experimento foi realizado.

(c) Contenha uma descrição da estratégia na geração da fórmula que representa o problema

(d) Contenha um gráfico de tempo (ou número de nós) $\times$ N (ou N^2) com pelo menos 10 pontos e uma pequena discussão do que pode-se inferir a partir do gráfico.

(e) Contenha uma conclusão que responda a pergunta: o problema das N-rainhas é um problema difícil?

Tanto o relatório quanto o programa devem ser inseridos num único arquivo do tipo zip ou tgz e submetidos através do moodle.

O ponto crucial aqui é determinar o valor máximo de N a ser mostrado. Uma vez que este esteja determinado, é importante determinar o espaçamento entre os valores de N; esta decisão depende de se será utilizado o valor de N ou N^2 nas abscissas.

5 Medindo o Tempo de Execução

Para medir o tempo de execução externamente (ou seja, ignorando o tempo medido pelo programa), use o comando `time`. Este comando, seguido de uma chamada de programa na linha de comando, executa o comando e depois disso mostra o(s) tempo(s) consumido(s) pelo programa. Por exemplo, o comando

```
% time ./zchaff rainha10.cnf > /dev/null

real    0m0.032s
user    0m0.030s
sys     0m0.010s
```

executa o resolvedor SAT em um arquivo chamado `rainha10.cnf`, desviando toda a saída para `/dev/null` (ou seja, não mostra nada da saída do programa). No final, o programa time mostra que o provador de teoremas levou 32ms para executar. Só que, destes, apenas 30ms fora usados pelo programa, (em operação do usuário). O resto foi gasto em ciclos do sistema. O fato que `real` $<$ `user + sys` indica que o programa time não tem uma precisão assim tão grande.

O tempo que importa é o user time, que em geral é um pouquinho maior que o tempo total relatado pelo zchaff.