

Resolvendo Sudoku com Inteligência Artificial

Matheus Teixeira Fernandes

João Paulo Pizani Flor

José João Junior

12 de setembro de 2010

1 Introdução

O estudo da Inteligência Artificial se concentra na resolução de problemas computacionais para os quais:

- Não existe solução algorítmica eficiente
- São realizados (acredita-se) bem pelos seres humanos
- São realizados bem por seres vivos (processos naturais)

Uma classe de problemas tipicamente tratada por métodos de Inteligência Artificial é a dos problemas NP, em especial os NP-Completo[5]. Tais problemas são caracterizados por terem soluções facilmente *verificáveis* porém difíceis de serem encontradas. Alguns exemplos de problemas NP-Completo são: subset sum[3], clique (subgrafo completo) e o problema da mochila[2].

Neste trabalho, especificamente, foram utilizadas técnicas de Inteligência Artificial (métodos de busca) para a solução do quebra-cabeça numérico Sudoku[4]. O “tabuleiro” de Sudoku é quadriculado e com dimensões quadráticas, sendo que o tamanho mais comum de tabuleiro de Sudoku é 9x9. Inicialmente apenas algumas das casas do tabuleiro estão preenchidas com números de 1 a 9.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figura 1: Tabuleiro de Sudoku em um estado “inicial”

O objetivo do Sudoku é preencher totalmente o tabuleiro, também com números de 1 a 9, sendo que não podem existir repetições de números em uma linha, em uma coluna ou em uma “célula” (região 3x3). Um exemplo de tabuleiro Sudoku resolvido encontra-se abaixo.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Figura 2: Tabuleiro de Sudoku resolvido

2 Estratégias adotadas

Há várias estratégias (algoritmos) para a solução do Sudoku, a maior parte delas aplicável também a outros problemas de IA. As mais notórias[1] são:

Busca cega por profundidade Consiste em se construir uma *árvore de estados*, onde cada nodo consiste de uma configuração válida no tabuleiro de sudoku, e a relação entre um nodo e seus nodos filhos é um movimento válido (um número colocado no tabuleiro). Um ou mais nodos folha de tal árvore serão soluções do problema. A árvore é construída *um ramo por vez*, ou seja, um caminho completo da raiz até a folha é feito em cada iteração, antes de se tentar um novo caminho partindo-se da raiz.

Busca cega por altura Bastante similar ao algoritmo de busca cega por profundidade, sendo que a única diferença com relação a este é que a construção da árvore é realizada *um nível por vez*.

Busca por satisfação de restrições Sob essa ótica, as soluções para o problema são vistas como um conjunto de objetos cujo estado precisa satisfazer uma certa coleção de *restrições*, sendo tais restrições predicados sobre o estado de um objeto. A busca comumente é realizada utilizando-se de *backtracking*. Um outro detalhe importante nesse método é a ordem com que as variáveis de estado serão analisadas. Dependendo da ordenação das variáveis o tempo de busca pode variar significativamente. Duas heurísticas de ordem para as variáveis são:

Escolher primeiro a variável mais restringida

Escolher primeiro a variável implicada em mais restrições

As estratégias adotadas no nosso trabalho foram: Busca cega por profundidade, Busca por satisfação de restrições com variável mais restringida e por satisfação de restrições com variável implicada em mais restrições. Cabe ressaltar que nossa versão do algoritmo de busca em profundidade é ITERATIVA, e portanto utiliza um pilha explícita para controle de fluxo, ao invés de utilizar a própria pilha de chamada de funções do computador.

3 Estrutura do trabalho

A programação do trabalho foi feita seguindo-se a metodologia MVC (Model View Controller), para que os códigos de interface gráfica e lógica de resolução ficassem devidamente segregados. Toda a parte de interface gráfica com o usuário (GUI) foi desenvolvida utilizando-se o *framework* C++ Qt, e está localizada no subdiretório sudokuQT. Maiores detalhes sobre os requisitos para executar o trabalho estão disponíveis na próxima seção.

Já o código que modela um tabuleiro de Sudoku e que efetivamente implementa os métodos de busca escolhidos pelo nosso grupo estão nos subdiretórios `include/` e `src/`. Em `include/` encontram-se os cabeçalhos C++ com a declaração de todas as classes, cuja respectiva implementação encontra-se em um arquivo de mesmo nome, porém em `src/`. As classes programadas e suas funcionalidades são descritas a seguir:

Sudoku Um objeto da classe Sudoku representa um tabuleiro do jogo. Ela é implementada utilizando-se uma matriz bidimensional de números inteiros positivos. A classe Sudoku fornece métodos para obter o valor e preencher uma determinada posição, assim como para verificar a corretude do tabuleiro (se as restrições são atendidas).

SudokuLoader A classe utilitária SudokuLoader tem métodos que permitem o carregamento de um tabuleiro a partir de um arquivo de texto, sua interpretação e a criação de um objeto da classe Sudoku que o represente.

Solver É a *superclasse* para todos os métodos de busca implementados. Tem métodos para resolver um jogo, calcular o tempo e calcular a memória utilizada no processo de solução. Como uma classe interna à classe Solver também está presente o modelo de árvore utilizado em nosso trabalho.

BlindSolver Esta subclasse de Solver implementa o método de busca cega por profundidade, e utiliza como estrutura de suporte ao percorrimento da árvore de estados uma fila.

RestrictionSatisfactionSolver Implementa o método de busca por satisfação de restrições, favorecendo sempre a escolha da variável mais restringida.

RestrictionSatisfactionSolverPlus Subclasse de Solver que realiza a busca também por satisfação de restrições, porém fazendo uma escolha preferencial da variável implicada em um maior número de restrições.

4 Instruções para execução

O arquivo binário executável do programa já está disponível no subdiretório `sudokuQT/`, e seu nome é `"sudokuQT"`. Sendo assim, não é necessária compilação. Porém, como o trabalho foi desenvolvido usando-se o framework QT para interface gráfica, existe uma dependência desse framework, e pode ser necessária a instalação do mesmo para que o trabalho seja executado.

Felizmente, na distribuição Ubuntu Linux a instalação do QT é bastante simples, pois há pacotes do QT no repositório padrão do sistema. Para realizar a instalação basta executar num terminal o seguinte comando:

```
sudo aptitude install libqt4-{core,designer,dev,gui}
```

Referências

- [1] Algorithmics of sudoku. http://en.wikipedia.org/w/index.php?title=Algorithmics_of_sudoku&oldid=383815040. Link permanente p/ a revisão referenciada.
- [2] Knapsack problem. http://en.wikipedia.org/w/index.php?title=Knapsack_problem&oldid=383777109. Link permanente p/ a revisão referenciada.
- [3] Subset sum problem. http://en.wikipedia.org/w/index.php?title=Subset_sum_problem&oldid=381947959. Link permanente p/ a revisão referenciada.
- [4] Sudoku. <http://en.wikipedia.org/w/index.php?title=Sudoku&oldid=382990789>. Link permanente p/ a revisão referenciada.
- [5] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, page 158. ACM, 1971.