Sudoku como PSR

•••

Aluno: Guilherme Zamberlam Pomini
RA: 99345
Disciplina: Inteligência Artificial 1
Professor: Wagner Igarashi
UEM - DIN - Ciência da Computação

O problema do Sudoku

Sudoku - O que é

- Sudoku é um jogo de quebra-cabeça baseado em colocação lógica de números em um tabuleiro. Tradicionalmente o Sudoku é jogado em um tabuleiro com 81 células divididas em 9 linhas, 9 colunas e 9 blocos.

- O jogo tem como objetivo preencher o tabuleiro de forma que nenhuma linha, coluna ou bloco possua números repetidos, que variam de 1 a 9.

- Pode ser estendido para qualquer ordem de N^2 x N^2, sendo a mais comum N = 3

Sudoku - Exemplo

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

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

Sudoku - Computacionalmente

- A resolução do Sudoku é feita através de um estado inicial do tabuleiro parcialmente preenchido de forma que seja possível induzir os próximos passos a partir do estado existente.

- Computacionalmente resolver um Sudoku de ordem N^2xN^2 é um problema NP-Completo, não havendo uma solução em tempo polinominal conhecida. Uma solução simples para um Sudoku é avaliar através de força bruta todas as possibilidades de preenchimento para as células vazias.

Problemas de Satisfação de Restrições

PSR - Definição

Um Problema de Satisfação de Restrição (PSR) é todo problema que possui um conjunto de variáveis, um conjunto de valores possíveis para as variáveis (o domínio de cada variável) e um conjunto de restrições, possuindo como objetivo encontrar um conjunto de atribuições que satisfaçam todas as restrições.

PSR - Atributos Básicos

1. Estado Inicial: uma associação de valores às variáveis, podendo ser parcial ou total

2. Função Sucessor: responsável por atribuir um valor a uma variável desde que não entre em conflito com as restrições

3. Teste de Objetivo: verificar se o estado atual é uma solução do problema

4. Custo do Caminho: constante para cada passo

PSR - Resolução

Os PSR podem ser resolvidos de várias formas, existindo inúmeras heurísticas e técnicas aplicáveis. No entanto, tipicamente um PSR é resolvido como um problema de busca, sendo as abordagens mais comuns são variações de Backtracking, Propagação de Restrições e Busca Local.

Sudoku modelado como PSR

Sudoku PSR - Atributos

É fácil notar como o problema do Sudoku pode ser facilmente visto como PSR.

A própria natureza do jogo e suas regras são baseadas em restrições e a sua solução é satisfazer todas as restrições.

Sudoku PSR - Atributos

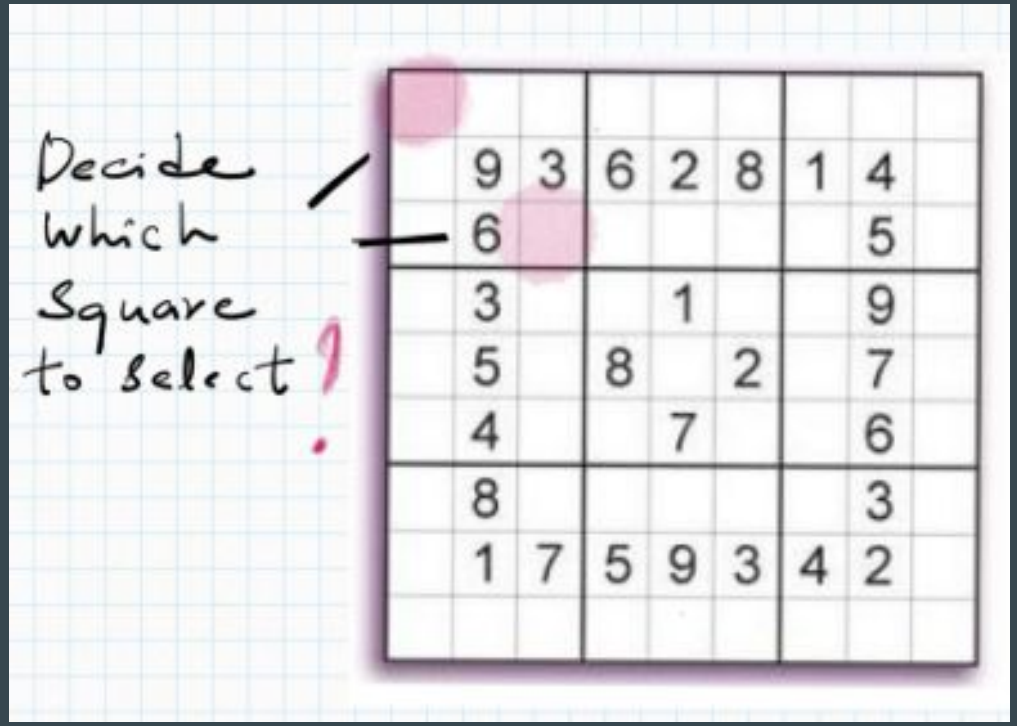
- Conjunto de Variáveis: cada célula vazia do tabuleiro.
- Domínio de Valores: números de 1 a 9 (no caso do sudoku 9x9).
- Estado: uma das possíveis atribuições de valores para as células.
- Solução: um tabuleiro totalmente preenchido que satisfaça todas as restrições
- Restrições:
  - Não pode haver números repetidos nas linhas
  - Não pode haver números repetidos nas colunas
  - Não pode haver números repetidos nos blocos

Implementação

Implementação - Escolha da Variável

- A escolha da variável (célula) em um sudoku é um dos fatores que mais afetam o desempenho
- Deseja-se escolher as variáveis de forma a minimizar a árvore de busca do algoritmo
- Utiliza-se a Heurística da Variável mais Restrita para a escolha da variável
- A Heurística se assemelha ao pensamento humano, escolhendo primeiramente as variáveis que possuem menos valores disponíveis

Implementação - Escolha da Variável



Implementação - Escolha do Valor

- Além da escolha das variáveis, a escolha da ordem dos valores também interfere no tempo de execução
- Uma escolha ótima de valores diminui a chance de entrar em galhos da árvore que não levam a solução ótima
- Como em um Sudoku cada valor tem uma quantidade máxima de vezes que pode aparecer (9 no caso do Sudoku 9x9) uma abordagem é testar primeiro os valores mais utilizados e ainda disponíveis
- Heurística do Valor mais Restrito

Implementação - Ambiente

- A implementação foi feita no Sistema Operacional Windows 10 64 bits 8GB RAM
- O código foi inteiramente feito em Python (versão 3) no Visual Code Studio
- O código foi dividido em 2 classes:
  - A classe Sudoku possui a lógica do jogo Sudoku e as funções auxiliares
  - A classe Main possui os algoritmos de PSR e Backtracking bem como a leitura do arquivo de entrada e a geração da saída
- Para fins de testes é utilizado um arquivo já pronto com Sudokus de nível Difícil

Implementação Algoritmo PSR

```
def resolveSudokuPSR(sudoku, celula = None):
    if sudoku.preenchidos == sudoku.n ** 4:
        return True
    if celula == None:
        celula = sudoku.proximaCelula()
    valoresCelula = sudoku.valoresCelula(celula)
    valoresCelula.sort(key = lambda x: sudoku.quantidade[x], reverse = True)
    for x in valoresCelula:
        sudoku.tabuleiro[celula[0]][celula[1]] = x
        sudoku.preenchidos += 1
        sudoku.quantidade[x] += 1
        if sudoku.preenchidos == sudoku.n ** 4:
            return True
        proximaCelula = sudoku.proximaCelula()
        if proximaCelula != False:
            if(resolveSudokuPSR(sudoku, proximaCelula)):
                return True
        sudoku.tabuleiro[celula[0]][celula[1]] = 0
        sudoku.preenchidos -= 1
        sudoku.quantidade[x] -= 1
    return False
```

Implementação - Algoritmo Backtracking

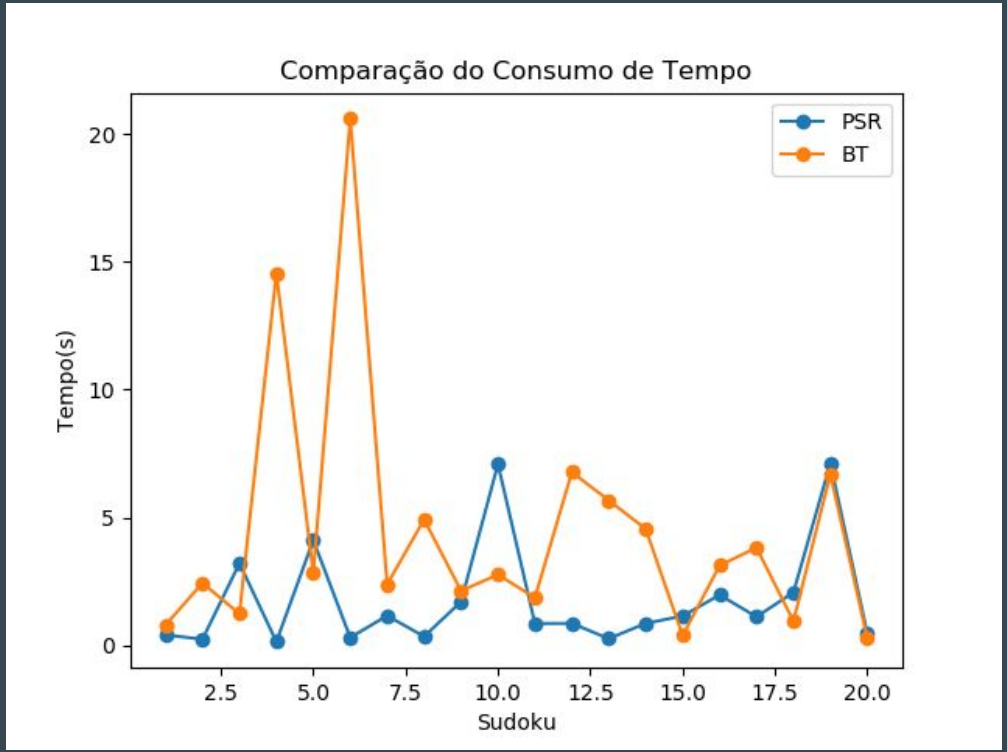
```
def sudokuBackTrack(sudoku, celula = None):
    if celula == None:
        celula = sudoku.proximaBackTrack()
    if celula == True:
        return True
    for i in range(1, sudoku.n ** 2 + 1):
        if sudoku.podeInserir(celula, i):
            sudoku.tabuleiro[celula[0]][celula[1]] = i
            if sudokuBackTrack(sudoku, sudoku.proximaBackTrack()):
                return True
            sudoku.tabuleiro[celula[0]][celula[1]] = 0
    return False
```

Resultados

Resultados

Foram feitos testes de resolução de diferentes Sudokus usando as abordagens PSR e de Força Bruta como forma de comparação. Para 20 tabuleiros de Sudoku diferentes com cada um sendo resolvido duas vezes, uma para cada abordagem.

É possível notar o desempenho superior do PSR, consumindo menos tempo em média, com diferenças de até 200%.



Conclusão

Conclusão

- É possível modelar o problema do Sudoku como um PSR
- A modelagem PSR tem como objetivo diminuir a árvore de busca para o algoritmo base de Backtracking
- A “poda” na árvore de busca gera um ganho significativo de performance
- Devido a natureza da modelagem PSR, é possível induzir que o seu ganho em comparação a abordagem padrão de busca aumenta conforme o tamanho do problema aumenta.

Referências

- Haralick and Elliott 1980] Haralick, R. M. and Elliott, G. L. (1980). Increasing tree search efficiency for constraint satisfaction problems. Artificial intelligence
- [Kemisetti 2018] Kemisetti, A. (2018). Solving sudoku . . . think constraint satisfaction problem.
- [Kumar 1992] Kumar, V. (1992). Algorithms for constraint-satisfaction problems: A survey. AI magazine
- [Seif 2017] Seif, G. (2017). Solving sudoku using a simple search algorithm.