UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO

JOSÉ MARIA CLEMENTINO JUNIOR **11357281**

Projeto 1– SUDOKU

São Carlos

2019

Introdução

O sudoku é um jogo tradicional de tabuleiro quadrado, que possui tamanho n x n, em que n representa a quantidade de linha e colunas, deste modo o número de células é de n^2. O sudoku quando tratado para resolução por meio de técnicas computacional não é uma tarefa trivial.

Desta forma o Sudoku é considerado um problema de satisfação de restrição (PSR), de modo que para atingir a resolução do projeto deve-se atender um conjunto de sub-regras(restrições): conforme a especificação do trabalho, no qual é dado um tabuleiro 9x9 faz-se necessário preencher os valores ausentes:

O novo número atribuído para a célula não pode se repetir: i) na mesma linha (Restrição 1); ii) mesma coluna; iii) e no no mesmo quadrante 3x3 (Restrição 3).

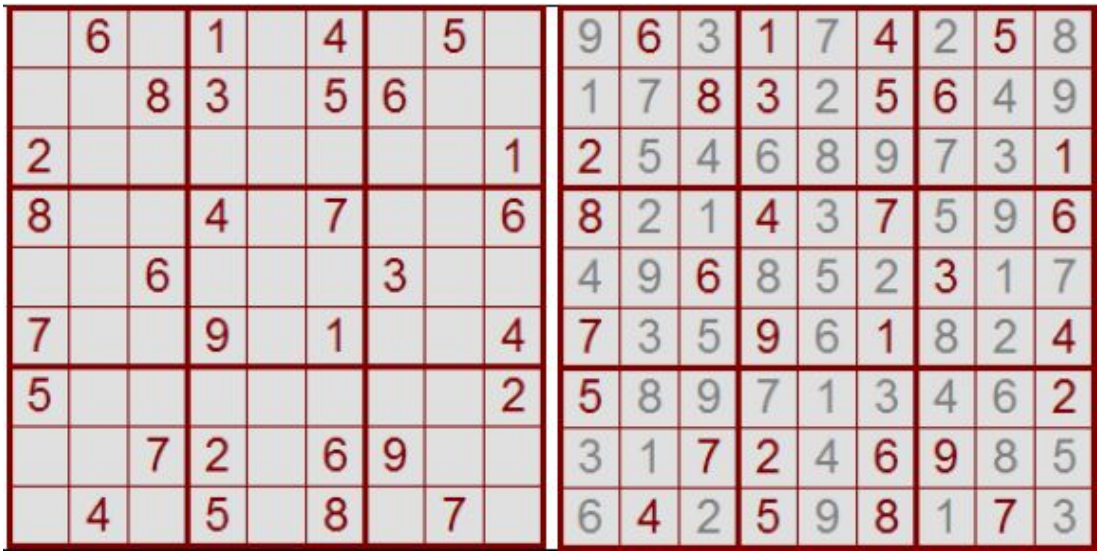

Figura 1: Exemplo de Resolução do Sudoku

Deste modo: vamos contextualizar o jogo do sudoku para o contexto do (PSR), onde:

Variáveis: as variáveis são todas as posições da célula possível

[0,0],[0,1],[0,2],[0,3],[0,4],[0,5],[0,6],[0,7],[0,8],

…. até ….

[8,0],[8,1],[8,2],[8,3],[8,4],[8,5],[8,6],[8,7],[8,8]

Restrições: não pode repetir na mesma linha (Restrição 1), na mesma coluna (Restrição 2) e na mesmo quadrante (Restrição 3).

Domínio: são os possíveis para realizar a atribuição, no caso (1,2,3,4,5,6,7,8,9).

Implementação

Linguagem: Python **Versão:** 2.7

Para executar o programa use: resolverSudoku.py **-i**<nomedoarquivo> **-t**<tipooperação>

Onde <tipooperação>: 1 - Backtracking sem heurística

2 - Backtracking com verificação adiante

3 - Backtracking com verificação adiante e valores mínimos remanescentes.

Modelo de entrada: resolverSudoku.py -i entrada.txt -t 1

Bibliotecas necessárias: time, sys, getopt, unicodecsv, csv e deepcopy.

O algoritmo em seu escopo principal é dividido em duas classes: *backtSimples e backtEmComum.* A *backtSimples* responsável por resolver o sudoku sem a utilização de heurísticas de poda, ela é composta por suas funções, que são apresentadas e comentadas no código. Já a *backtEmComum* é utilizada para resolução do sudoko, por meio da utilização das heurísticas de poda: verificação adiante e verificação adiante e valores mínimos remanescentes. O diferencial entre as duas heurísticas, econtra-se nas seguintes funções: *verifica_pos_vazia_e_troca_valorValido, utilizada* na heurística de verificação adiante, que retornar os possíveis variáveis a serem atribuídos em cada posição ao invés de realizar a recursão com todos os domínios possíveis; *prox_espaco_preencher_Min,* função utilizada para a heurística de poda verificação adiante com e valores mínimos remanescentes, diferentemente da heurística anterior, para cada célula além de buscar os valores possíveis a serem atribuídos ele aplica uma ordem de prioridade de iniciar o processo de backtracking à partir da variável que contém menor número de domínio. São utilizadas demais funções para leitura de entrada saída de resultados para análises que podem ser encontradas comentadas no link do código.

Análise dos Resultados

Para realização da análise dos resultados foi testado uma entrada com 55 jogos de Sudokus à serem resolvidos. A seguir é apresentado o desempenho de cada um dos algoritmos em função da mesma entrada:

- **Algoritmo 1: Backtrackin sem Heurística:**
  - **Resolvidos:** 40
  - **Tempo:** 733,21 segundos = 12,22 minutos.
- A**lgoritmo 2 : Backtrackin + verificação adiante:**
  - **Resolvidos:** 51
  - **Tempo:** 553,14 segundos ou 9,21 minutos.

- **Algoritmo 3: Backtrackin + verificação adiante + valores mínimos remanescentes:**
  - **Resolvidos:** 55
  - **Tempo:** 78,85 segundos ou 1,31 minutos.

É importante lembrar que os Sudokus considerados **não resolvidos** são aqueles que ultrapassaram 10^6 (1000000) tentativas de atribuições.

A seguir são apresentados os gráficos em relação ao número de atribuições e ao tempo para cada sudoku.

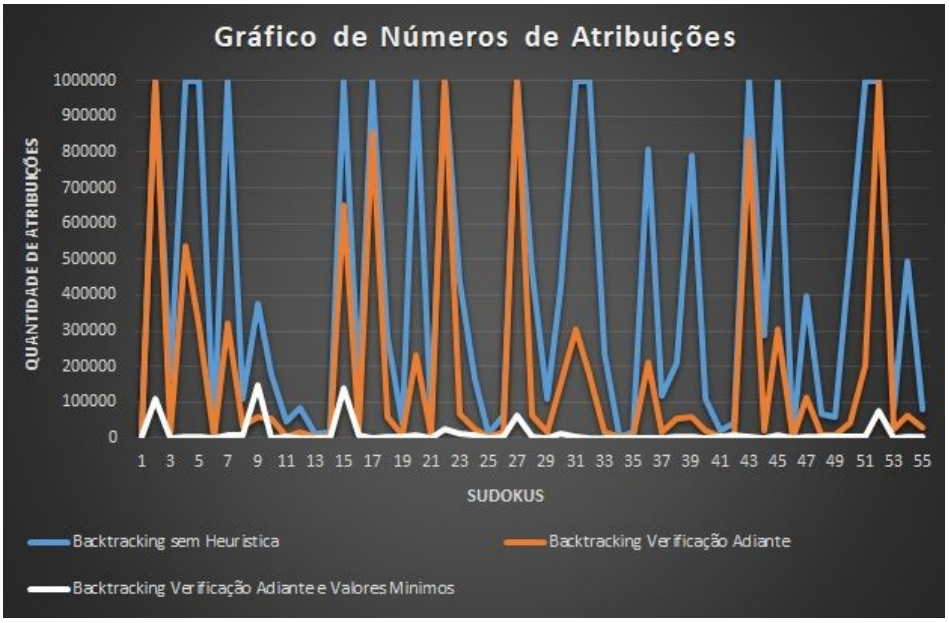

Figura 2: Número de atribuições x Sudokus Link

Figura 3: Tempo x Sudokus Link

Conclusão

É possível observar que existem diferenças significativas quanto ao uso de heurísticas de poda para resolução do problema especificado. Foi possível observar que o algoritmo 3, demonstrou melhores resultados em relação aos algoritmos 1 e 2, tanto em relação ao número de atribuições (Figura 2) quanto ao tempo (Figura 3). Também pode-se observar que o algoritmo 2 mesmo demonstrando em resultados melhores(de modo geral) em relação ao número de atribuições e tempo quando comparado ao algoritmo 1, em algumas resoluções de sudokus apresentou um número superior ao algoritmo 1 de atribuições e tempo. A explicação pode-se atribuir que em determinados sudokus é necessário a atualização de todos os valores das variáveis que ainda não receberam um valor.

Referências:

http://www.lcad.icmc.usp.br/~jbatista/scc210/ForcaBruta.pdf

https://www.geeksforgeeks.org/backtracking-algorithms/

https://spin.atomicobject.com/2012/06/18/solving-sudoku-in-c-with-recursive-backtracking/