Um estudo de comparação entre algorítmos de busca cega e heurística para resolução do Sudoku

Felipe Hauschild Grings[1]

[1]Universidade do Vale do Rio do Sinos (UNISINOS)
91.501-970 – São Leopoldo 93022-750 – RS – Brazil

Abstract. *This paper compares two popular search methodologies used to solve the challenge known as Sudoku, called as blind-search, also known as brute-force, and the other as heuristic search. The heuristic search was based on the perception of the gaming steps of any human playing. From the analysis was proved by results, according to the theory, that the informed search (heuristic) had greatest improvement on several occasions, however it does not improve the performance for all cases, and there may be consequences when the decision results in an unsolved path.*

Resumo. *Este artigo compara duas metodologias de busca popularmete usadas para a resolução do desafio conhecido como Sudoku, sendo eles a busca cega, conhecida também como força bruta, e a outra como busca heurística. A busca heurística foi baseada na percepção da forma de jogo de um ser humano qualquer. A partir da análise dos resultados comprovou-se conforme a teoria que a busca informada (heurística) apresenta grande melhora em diversas ocasiões, porém não mantém tal performance para todos os casos, podendo haver consequências quando a decisão resulta em um caminho sem solução.*

1. Introdução

Sudoku é um quebra-cabeça numérico que consiste em um quadro 9x9 subdividido em "mini quadros" de tamanho 3x3. Cada coluna, linha e "mini quadro" contém números de 1 a 9 cada, onde em um Sudoku original existe apenas uma solução. Sudoku atingiu uma grande popularidade nas últimas décadas, ganhando fama internacional por volta do ano 2005, gerando uma gama de novos puzzles de diferentes tamanhos e regras [3]. O quadro de Sudoku inicia com algumas células preenchidas e imutáveis. Estes valores disponibilizados são o suficiente para encontrar a sua solução. Os valores dados devem ser bem definidos para que não haja nenhuma "consequência lógica". O objetivo é, então, completar a atribuição, inserindo os valores ausentes de tal forma a satisfazer as restrições, manualmente usando a lógica ou, como é cada vez maior a tendência, utilizando soluções automatizadas. Nenhum meio geral ainda é conhecido para determinar o número mínimo de dados necessários para uma atribuição parcial que leva a um solução única [1]. Diversos trabalhos experimentais foram realizados nesta área.

As soluções automatizadas podem ser razoavelmente divididas em duas categorias; buscas não-informadas, no qual existem diversas variações para otimizar a passagem pelos nodos, e algoritmos de otimização de pesquisa heurística (que transformam algumas ou todas as restrições do problema em aspectos de otimização em função de avaliação). O foco desse artigo é a comparação de desempenho entre as duas soluções propostas, considerando desempenho o número de laços executados até encontrar a primeira solução.

2. Conceitos Gerais

Uma busca em largura é um método de busca não-informada que análisa sistematicamente todos os vértices de um grafo direcionado ou não-direcionado. O algoritmo realiza um busca de grande custo computacional pois passa por todas as arestar e vértices do grafo. Ó algoritmo deve garantir que nenhum vértice ou aresta será visitado mais de uma vez, e para isso organiza uma estrutura de dados no formato de fila, comumente conhecido como fronteira. Dessa maneira as visitas são ordenadas em ordem de chegada e um vértice conhecido não será mais visitado.

Tal mecanismo permite que se descubra todos os vértices a uma distância n do vértice raiz antes de qualquer outro vértice de distancia maior que n, sendo n o número de arestas para atingir qualquer outro vértice no grafo considerado. Essa característica do algoritmo permite construir uma árvore de distâncias mínimas (menor número de arestas) entre o vértice raiz e os demais, sendo que o vértice responsável por enfileirar o seu vizinho que será o vértice pai deste na representação em árvore gerada.

O desenvolvimento desses algoritmos sempre levam em considerações dois pontos importantes da computação, sendo eles: Fazer o algoritmo ter um tempo de execução sempre aceitável e Ser a solução ótima ou provavelmente boa para o problema em todos os casos.

Por outro lado, um algoritmo heurístico não cumpre necessáriamente essas prioridades, podendo ser ou o algoritmo que encontra boas soluções a maioria das vezes, porém não há garantia de que sempre vá encontrar a melhor solução e nem que será sempre no menor tempo de processamento, assim levando ao ponto de melhor responsa média em relação aos algoritmos de busca cega.

A pesquisa por heurísticas é uma pesquisa realizada por meio da quantificação de proximidade a um determinado objectivo. Diz-se que se tem uma boa (ou alta) heurística se o objecto de avaliação está muito próximo do objectivo; diz-se de má (ou baixa) heurística se o objecto avaliado estiver muito longe do objectivo. Etimologicamente a palavra heurística vem da palavra grega Heuriskein, que significa descobrir (e que deu origem também ao termo Eureca).

Um algoritmo aproximativo (ou algoritmo de aproximação) é heurístico, ou seja, utiliza informação e intuição a respeito da instância do problema e da sua estrutura para resolvê-lo de forma rápida.

Entretanto, nem todo algoritmo heurístico é aproximativo, ou seja, nem toda heurística tem uma razão de qualidade comprovada matematicamente ou prova formal de convergência. Por este motivo, em várias referências bibliográficas distingue-se os termos algoritmo aproximativo e heurística:

aproximativo é a denominação do algoritmo que fornece soluções dentro de um limite de qualidade absoluto ou assintótico, assim como um limite assintótico polinomial de complexidade (pior caso) comprovado matematicamente; heurística e método heurístico são denominações para o algoritmo que fornece soluções sem um limite formal de qualidade, tipicamente avaliado empiricamente em termos de complexidade (média) e qualidade das soluções.

A heurística é um conjunto de regras e métodos que conduzem à descoberta, à

invenção e à resolução de problemas. Também é uma ciência auxiliar da História que estuda a pesquisa das fontes.

3. Estado da Arte

Diversos autores apresentam o Sudoku como um problema de satisfação de restrição [1]. Nessas abordagens, ao invés de tentar empregar uma função objetivo para determinar um caminho através de um espaço de estado, uma solução é vista como um conjunto de valores que pode ser atribuído a um conjunto de variáveis. Essa abordagens tiram vontagem do quebra-cabeça. O Sudoku demonstrou-se um problema NP-completo para dimensões superiores [1]. Uma grande variedade de abordagens para solução do Sudoku tem sido empregadas nos útlimos anos, como abordagens de satisfação de restrição tem sido empregadas para o Sudoku, como a aplicação de correspondência bipartida e fluxo algoritmos [1], o uso de verificação direta e pesquisa de algoritmos de discrepância limitada [3], e a aplicação da Conjunctive Normal Form (CNF) para produzir codificações diferentes do problema. Uma abordagem de Programação Inteira foi usado em [2], que considerado tanto a criação de quebra-cabeças quanto a solução de quebra-cabeças Um algoritmo de ramificação e limite foi usado para determinar as soluções ideais. Por fim, uma a abordagem interessante foi desenvolvida a partir de algoritmos genéticos [3]. Nesse os autores desenharam crossovers geométricos fortemente relacionado ás restrições do problema. Porém não é a melhor solução por não explorar as restrições dos problemas para reduzir o custo da pesquisa, como relaram os autores.

* A. Método de força bruta: Esta é a solução mais simples e evita complexidade computacional. Esse algoritimo passa por todas as células vazias em uma ordém prédefinida e preenche com alguma possível variável. Caso nenhum possibilidade esteja disponível ele retorna ao estado anterior e altera para a próxima possibilidade da lista. Com isso o algortirmo continua até encontrar a solução.

* B. Recozimento Simulado: Recozimento simulado é um método probabilístico otimizado. Sudoku é condiderado resolvido quando todas as células estão preenchidas com as opções corretas. O método utiliza o preenchimento randômico como forma de otimização da solução. Quando o quadro todo se encontra preenchido, o número de erros contados e os últimos dois digitos são alterados. Após a alteração um novo resultado é obtido e comparado com o resultado anterior, é considerado melhor se conter menos erros que a jogada anterior.

* C. Busca Heurística: A busca heurística desenvolvida utilizada como informação adicional o número de restrições encontradas em cada célula disponível para preenchimento. Quando comparada todas as celulas, seleciona as que tiverem menor opções, assim obtendo uma menor ramificação da arvore e otimizando as próximas buscas. Como destacado anteiormente, ese método não garante melhora em todas as situações, pois há ainda possíveis jogadas na qual a escolha errada dentre as células de menores opções, abra uma novo ramo tão grande quanto o ramo vencedor do jogo, porém a probabilidade de percorrer os caminhos mais longos são reduzidos.

Com a análise e comparação de alguns dos métodos hoje estudados, apresenta-se uma melhor análise nos métodos menos probabilísticos.

3.1. Tipo de aplicação desenvolvida

Para realização do estudo foram desenvolvidas duas aplicações, utilizando a linguagem de programação Python, versão 3.6. O algoritmo recebe como entrada um arquivo .txt que contém em padrão de texto os dados para o Sudoku a ser resolvido. Com isso o algortimo carrega os dados em formatro de matriz filtrando os caracteres de auxilio para visualização humana e transformando os sublinhados em um valor numérico não importante para a solução, no caso o valor escolhido foi ”0”. O código foi separado em diversas funções na qual as mais importantes e relevantes para o problemas são as de análise a matriz. A checagem para conclusão do jogo ou análise dos próximos possíveis passos era realizada a partir de um reastrio por toda a matriz. Após desenvolvido o código para análise do Sudoku enfrenta-se o problema de busca, respectivamente. Para o decorrer da busca foi utilizado o método de busca em largura. Os conceitos deste algoritmo foram aproveitando a técnica de fronteira apresentada por XXXX[1]. A busca em largura foi desenvolvida considerando a próxima jogada sempre como o próximo valor direto, nulo, encontrado na matriz. Dessa forma o algortimo passa necessáriamente por todas as possíbilidades de forma a avançar naturalmente ao objetivo.

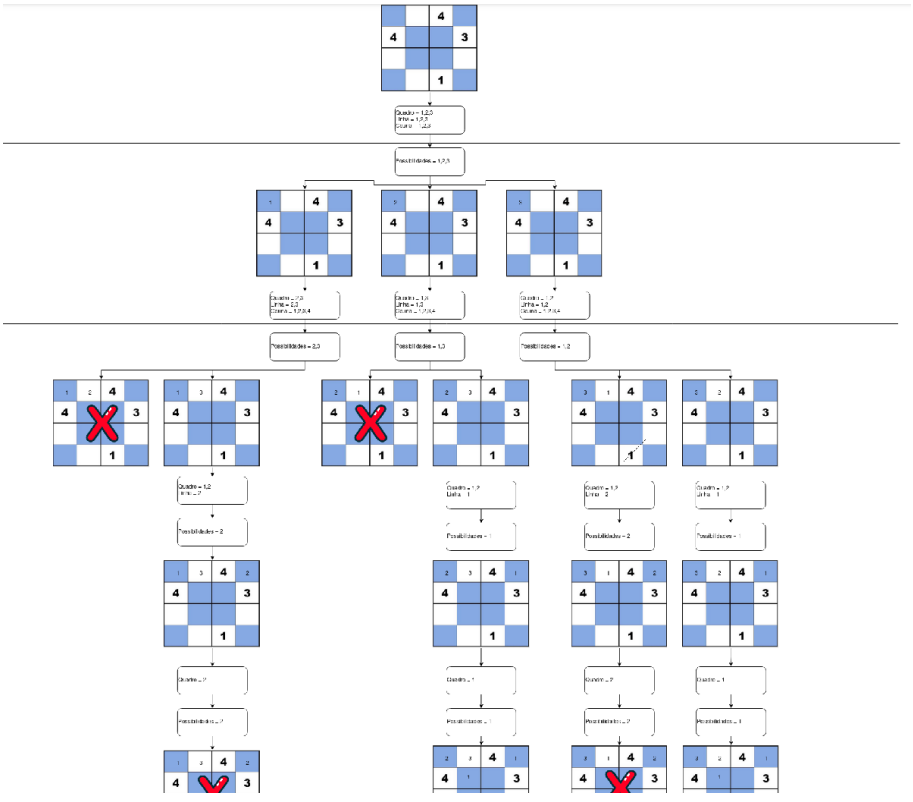

Figure 1. Arvore de possiblidades Sudoku 4x4

Para busca heurística foi realizado um algoritmo baseado na análise da construção da arvore de possibilidades da Figura 1. Nativamente o ser humano ao resolver o desafio busca inicar pelas células de menor possibilidades, reduzindo drásticamente as análises necessárias e as chances de erro futuro. Como visto na Figura 1, possíveis movimentos iniciais podem levar a um jogo longo, porém obrigatóriamente sem solução. Considerando esse conhecimento, foi desenvolvido um trecho de código que análisa diante de

cada jogada qual a melhor próxima jogada. O algortimo analisa o número de restrições de cada célula mutável, considerando que as células alteradas anteriormente não podem mais ser mutadas, e armazena em um terceiro eixo da matriz, transformando-a em um cubo. Esse valor é considerado o peso para cada possiblidade de jogada, então é escolhido a células com maior número de restrições, e não mais próxima seguinte. Com esse valor é possível saber a quantidade de ramificações aquele nodo tem, com isso, utilizando a busca em largura, acaba otimizando em quantidade de jogadas o encontro da solução.

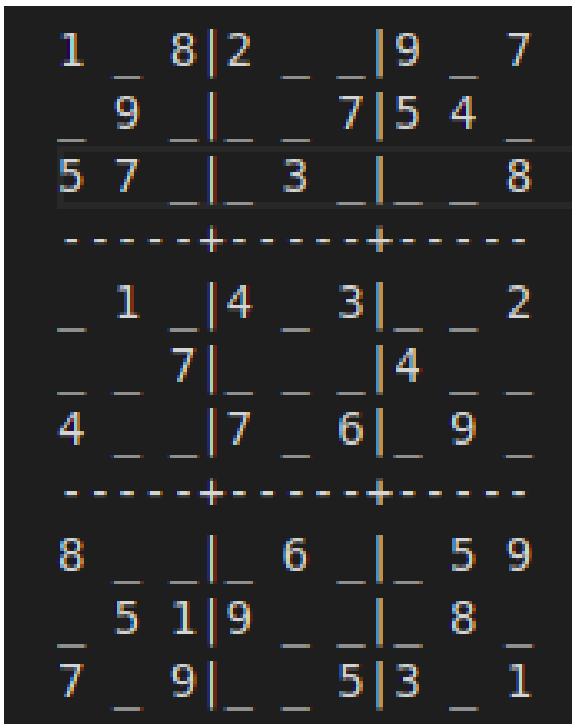

Figure 2. Formato de entrada para Sudoku

4. Análise dos Resultados

Para análise e comparação de resultados entre os dois algortimos foram analisados 7 diferentes jogos de Sudokus.

O algoritmo heurístico apresentou extrema vantagem na maioria dos casos, diminuindo em até 99% o número de ciclos necessários para solução do jogo. A medida que os jogos injetados aumentassem a complexidade o algoritmo heurístico se apresentou cada vez melhorl. Porém para a no Sudoku número 6 houve um resultado não esperado, no qual a relação de eficiência resultado em torno de 90%, decaindo considerávelmente sua performance em relação ao algoritmo de busca cega.

Independe do jogo, dificuldade proposta ou tamanho da matriz, o algorítmo heurístico obteve um resultado acima de 10x. O resultado era previsto dado que ao analisar as restrições de cada célula o número de tentativas sem futuro diminui drasticamente, podendo haver jogos no qual a busca cega melhore o desempenho em relação a normal, porém considerando a média o algoritmo heurístico provou sua eficiência.

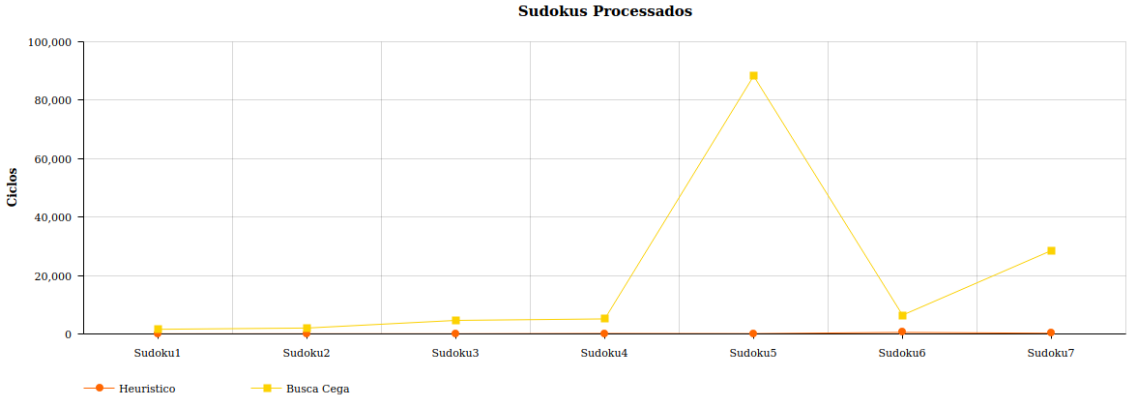

Figure 3. Gráfico de resultados

category	Heutisco	Largura
Sudoku1	46	1593
Sudoku2	52	2019
Sudoku3	52	4632
Sudoku4	153	5144
Sudoku5	105	88290
Sudoku6	530	6437
Sudoku7	236	28547

Figure 4. Tabela de resultados

5. Conclusão

Com a análise dos dados obtidos o algoritmo Heurístico, considerando como peso deciviso para próximas jogadas os número de restrições obtidas por célula, e realizando as jogadas a partir das células com maior número de restrições, mostrou-se grandiosamente maior dentro de todos os testes realizados. Para uma análise mais aprofundada e considerações reais de otimização é necessário a execução de um número maior de jogos.

6. References

[Pillay 2012], [S. K. Jones 2016], [Yato and Seta 2003], [Chatterjee 2014].

References

Chatterjee, S. (2014). *A Comparative Study On The Performance CharacteristicsOf Sudoku Solving Algorithms*. Addison-Wesley, 15th edition.

Pillay, N. (2012). Finding solutions to sudoku puzzles using human intuitive heuristics.

S. K. Jones, P. A. Roach, S. P. (2016). Construction of heuristics for a search-based approach to solving sudoku.

Yato, T. and Seta, T. (2003). Complexity and completeness of finding another solution and its application to puzzles.