Universidade de São Paulo – USP

Instituo de Ciências Matemáticas e de Computação

Disciplina: Projeto de Algoritmo

Professor: Gustavo Batista

Aluno: Pedro Henrique da Costa Ulisses

Projeto 1: Backtracking

1 – Implementação

O algoritmo *backtracking* e as estratégias de podas foram implementadas em C/C++. No arquivo que contem o código contém 3 constantes, TAMANHO, MVR e VERIFICACAO\_ADIANTE. A constante TAMANHO define o tamanho do sudoku, e as constantes MVR e VERIFICACAO\_ADIANTE que são uma *flags* para ligar/desligar o uso dessas estratégias de poda.

A implementação desenvolvida é capaz de ler uma entrada em um formato pré-definido, onde a primeira linha é um número inteiro que fornece o número de jogos de sudoku para serem resolvidos. As demais linhas contem os jogos de sudoku disposto em matrizes 9x9 e separados por uma linha em branco. Cada matriz é preenchida com valores de 0 a 9, onde o zero representa posições sem atribuições, ou seja, posições que o algoritmo vai atribuir valores para resolver o jogo.

O *backtracking* foi desenvolvido de forma recursiva e nesta função possui os trechos de códigos responsáveis por realizar as estratégias de poda com verificação adiante e consulta dos mínimos valores remanescentes. A consulta dos mínimos valores remanescentes retorna a posição, linha e coluna, que possui a menor quantidade de valores possível para ser atribuído e em caso de empate é realizada uma busca para identificar qual das posições possui mais restrições, ou seja, qual posição mais interfere nas outras posições. A poda utilizando a verificação adiante verifica se a última atribuição realizada torna qualquer valor incompatível com pelo menos uma posição sem um valor atribuído.

A cada atribuição realizada é verificado se o valor é compatível, ou seja, se o valor não se repete na linha, nem na coluna e nem na grade 3x3. É realizada a contagem do número de atribuições para cada jogo a ser resolvido assim como o tempo gasto para resolver cada jogo.

Foi levado em consideração o critério de parada quando o número de atribuições atingir 106 o algoritmo não progride mais nas buscas e sai das chamadas recursivas.

2 – Resultados

Um arquivo contendo 95 jogos de sudoku foi utilizado para verificar o desempenho do *backtracking*, *backtracking* com verificação adiante e *backtracking* com verificação adiante e com a heurística mínimo valor restritivo (MVR). A quantidade de atribuições e o tempo gasto para resolver cada jogo foram as métricas utilizadas para avaliar o desempenho dos algoritmos e das heurísticas adotadas.

Tabela 1 Porcentagem de jogos resolvidos com menos de 106 atribuições

|  |  |
| --- | --- |
| Algoritmos | % |
| *Backtracking* | 36 |
| *Backtracking* + verificação adiante | 62 |
| *Backtracking* + verificação adiante + MVR | 100 |

O desempenho dos algoritmos é apresentado nas tabelas 1 e 2. Utilizando apenas o backtracking foi possível resolver 35 do total de 95 jogos, os demais jogos atingiram a quantidade de 106 atribuições. Com a utilização do backtracking juntamente com a verificação adiante a quantidade de jogos resolvidos foi maior e com o acréscimo de mais uma heurística de poda, a MVR, o algoritmo conseguiu resolver todos os 95 jogos em uma quantidade de atribuições menor que o limite estabelecido. Na tabela 2 são apresentados o tempo e o número de atribuições médio que o backtracking com cada uma das heurísticas de poda atinge para resolver cada jogo.

Tabela 2 Tempo médio e o número médio de atribuições para resolver cada jogo

|  |  |  |
| --- | --- | --- |
| Algoritmos | Tempo (s) | Número de atribuições |
| *Backtracking* | 0,070942857 | 399633,8 |
| *Backtracking* + verificação adiante | 5,272491525 | 232629,6 |
| *Backtracking* + verificação adiante + MVR | 1,2696 | 39896,79 |

Com os resultados apresentados nas tabelas 1 e 2, conclui-se que a heurística utilizando verificação adiante juntamente com MVR é a estratégia de poda mais indica para resolver os jogos de sudoku, pois consegue resolver um jogo com um número menor de atribuições e em um baixo tempo em relação ao tempo atingido pelos demais algoritmos.