Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - USP

TRABALHO 1

Futoshiki 不等式

SCC0218 - Algoritmos Avançados e Aplicações

Prof.: Gustavo Enrique de Almeida Prado Alves Batista

Felipe Scrochio Custódio, 9442688
Gabriel Henrique Campos Scalici, 9292970

São Carlos
Setembro de 2016

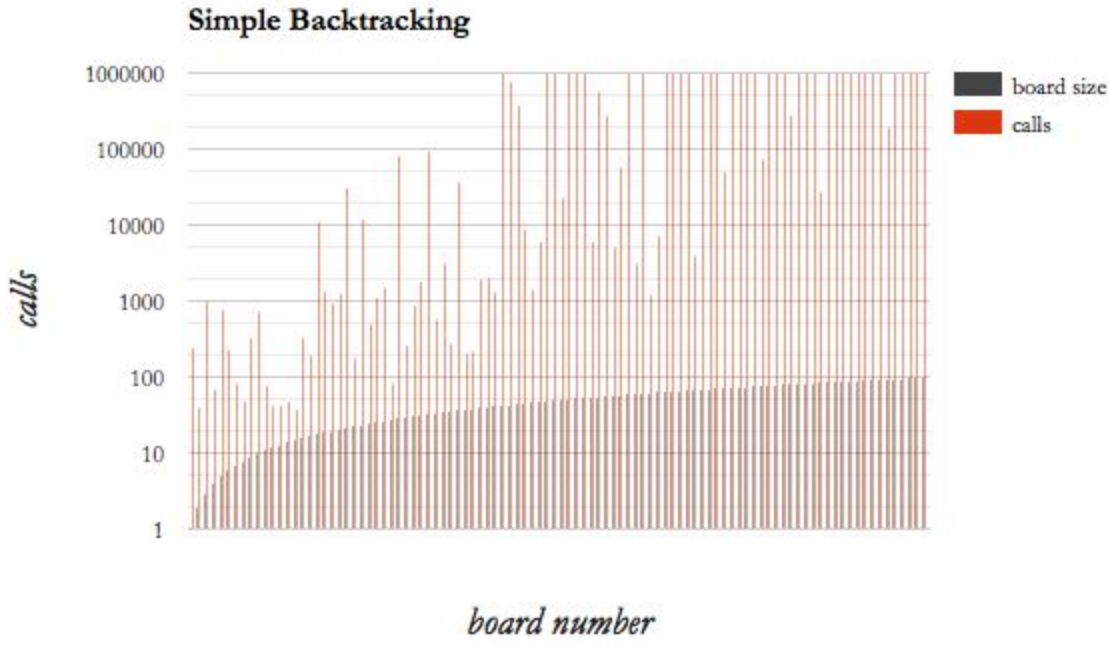
Futoshiki 不等式

Análise de Desempenho

Entrada: 100 tabuleiros

Heurística	Tabuleiros Resolvidos	Tempo Gasto
Backtracking Simples	62/100	1.247850 s
Menor Valor Remanescente	-/-	5.934055 s
Forward Checking	69/100	3.614945 s

- Backtracking Simples:



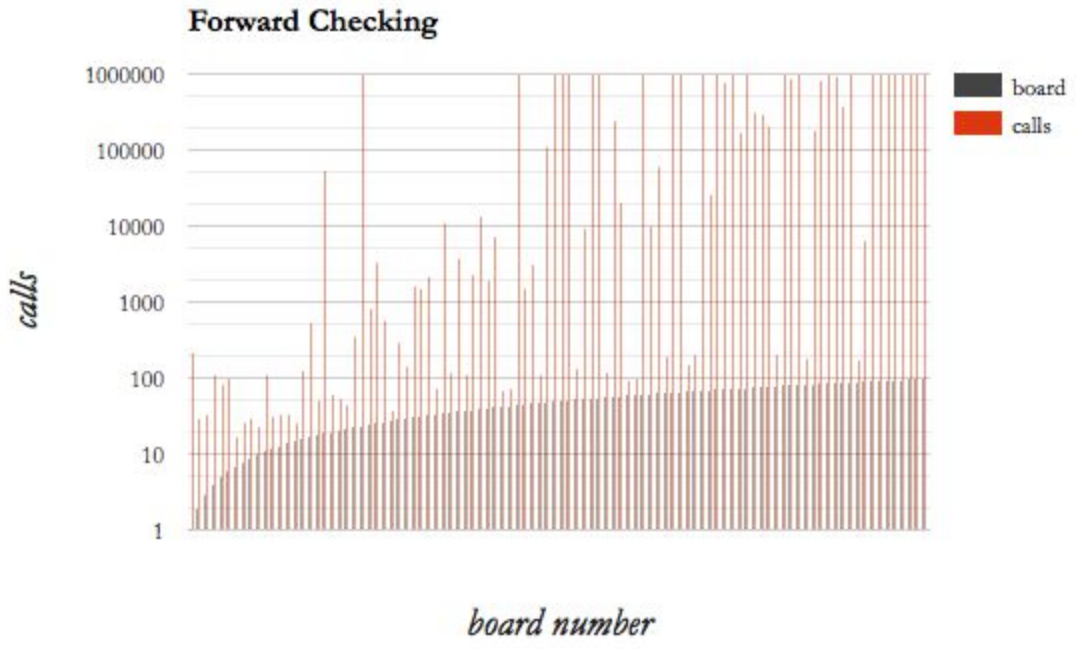
-

A implementação do backtracking simples está muito rápida, porém com os números máximos de chamadas, muitos tabuleiro estouram esse valor, sendo assim, muitos desses tabuleiros estão sem resolução.

Nosso algoritmo de backtracking simples sem heurísticas é basicamente um algoritmo recursivo que começa verificando se o número máximo de atribuições foram atingidas (Exigido na descrição do projeto como no máximo 10^6 atribuições) retornando "*FALSE*", caso ainda não tenha sido atingido o número máximo, é verificado se a análise não chegou no fim do tabuleiro, o que indica que todas as atribuições foram feitas sem conflitos, retornando "*TRUE*" caso tenha percorrido todo o tabuleiro, sendo essa a nossa análise para verificar se o futoshiki foi resolvido corretamente.

Após as duas verificações de parada, o algoritmo analisa todas as células do tabuleiro colocando os valores, caso não tenha um valor inicial (Valor já recebido na leitura do tabuleiro), e verificando se são válidos com a chamada da nossa função "*isValid*", já explicado anteriormente que verifica se não há o número na mesma linha, na mesma coluna e se as restrições do tabuleiro foram respeitadas, caso seja válido, o valor em questão é adicionado, e de forma recursiva, é chamado novamente o algoritmo, porém agora para a célula da próxima coluna do tabuleiro, e caso seja a última célula de coluna, é chamado para a primeira célula da próxima linha.

- Forward checking :



-

A implementação do backtracking com forward checking está rápida, demorando um pouco mais que o backtracking simples, porém, está resolvendo um número maior de tabuleiros, mostrando que as podas ajudam a não estourar em muitos casos o número máximo de chamadas.

Usando as listas, a implementação, é feita de maneira recursiva, de modo que é necessário algum ponto de parada, que assim como no simples deve ser se exceder o número máximo de "*calls*" que são as chamadas.

A lógica funciona basicamente como a implementação sem heurística, a diferença é que agora são usados os valores possíveis armazenados nas listas antes de adicionar um valor, ou testá-lo naquela posição.

Além dessa função, de cortar os valores que estão inválidos, usando a função "*isValid*", é percorrido todas as listas da mesma coluna e linha, removendo os valor que foi adicionado (Atualizando as listas de todas as células afetadas com "*updateList*"), dessa forma, ao remover quando fazemos uma escolha, é possível analisar antes de chegar em determinada célula, que ela não possui mais valores válidos. Caso após a atribuição de algum número, alguma célula fique sem valores possíveis, retorna "*FALSE*" e poda as possibilidades de backtracking.

Caso o valor colocado naquela célula não seja válido pois excluiu todos os valores da lista de outras células, esse valor é ressetado.

A forma de percorrer o tabuleiro (linhas e colunas) é similar ao sem heurística, já explicado anteriormente.

- Forward checking + MVR:

Nosso algoritmo está funcionando corretamente, porém não está analisando as restrições no tabuleiro completamente, e não foi possível identificar porque não está analisando as restrições, se tem essa chamada.

O algoritmo implementado com a junção das duas heurísticas, usou todo o código do algoritmo que tem somente Forward checking, com algumas pequenas modificações para que seja possível a implementação do MVR.

Tais modificações são basicamente, ao remover um dos valores possíveis da coluna e linha ao adicionar um valor, analisa qual das células possui a menor quantidade de valores possíveis (armazenados em "*size*"), escolhendo-a para análise e colocar um valor (heurística MVR).