Master's Thesis

Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Pfaffenwaldring 5B
D-70569 Stuttgart

Parallelizstion of a Sudoku Solver

Subtitle

Author Name

Examiner:	Prof. Dr. Ngoc Thang Vu Second Examiner
Supervisor:	Florian Lux
Start:	01.04.2020
End:	30.09.2020

Erklärung (Statement of Authorship)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst habe und dabei keine andere als die angegebene Literatur verwendet habe. Alle Zitate und sinngemäßen Entlehnungen sind als solche unter genauer Angabe der Quelle gekennzeichnet. Die eingereichte Arbeit ist weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen. Sie ist weder vollständig noch in Teilen bereits veröffentlicht. Die beigefügte elektronische Version stimmt mit dem Druckexemplar überein. [1]

(Author Name)

[1] Non-binding translation for convenience: This text is the result of my own work, and any material from published or unpublished work of others which is used either verbatim or indirectly in the text is credited to the author including details about the exact source in the text. This work has not been part of any other previous examination, neither completely nor in parts. It has neither completely nor partially been published before. The submitted electronic version is identical to this print version.

Contents

1 Background

A Sudoku puzzle is a n-by-n grid that contains numbers range from 1 to n. The goal is to fill all the missing numbers to complete the puzzle. There are ruls to the Sudoku puzzle Each number must appear in each row Each number must appear in each column Each number must appear in each subgrid The rules imply no duplicate numbers in any row, column and subgrid. Time to wait for a solution could grow exponentially as the puzzle grows larger and harder.

1.1 Solving Algorithms

2 Related Work

3 Proposed Approach

3.1 sequential approach

3.2 naive brute-force approach

Try every single combination of numbers to each square to find a solution. A blank n-by-n grid has a total of n$^{n} xndifferentpossiblecombinationofsolutions!(searchtreepic)Solvingsudokupuzzlesinthisw$

3.3 back-tracking approach

Similar to the naive algorithm, backtracking picks an empty square, tries all possible numbers and finds one that works (i.e., does not violate the rules). If the number we put into the square is valud, then repeat the procedure for the next empty square. As soon as we get an invalid number for an empty sqaure, we backtrack to the most recent step. eventually. So rather than trying to continuing a solution that can never possibly work which we do with naive algorithm, we're going continue solutions that currently work and if they don't work we backtrack to the last step and try something again. This is going to be a lot faster than trying every single possible combination of solutions as brute-force algorithm did.

3.4 parallel approach

give each thread a puzzle

4 Experiments

print$_b oard find_e mpty is_v alid do not check the same position that we just added in. duplicate solve_b oard$

5 Discussion

6 Summary

List of Figures

List of Tables