SUDOKU SOLVER USING ANT-COLONY-OPTIMIZATION ALGORITHM

A PROJECT REPORT

Submitted by

SREELEKSHMI PRATHAPAN (TKM20MCA2040)

to

The APJ Abdul Kalam Technological University

In partial fulfillment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Thangal Kunju Musaliar College of Engineering
Kerala

DEPARTMENT OF COMPUTER APPLICATIONS

JULY 2022

DECLARATION

I undersigned hereby declare that the project report on '**SUDOKU SOLVER USING ANT COLONY OPTIMIZATION ALGORITHM**" , submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Prof.Dr.Fousia M Shamsudeen. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Kollam

18/07/2022

Sreelekshmi prathapan

DEPARTMENT OF COMPUTER APPLICATIONS

TKM COLLEGE OF ENGINEERING

CERTIFICATE

This is to certify that, the project report entitled "**SUDOKU SOLVER USING ANT COLONY OPTIMIZATION ALGORITHM**" submitted by **SREELEKSHMI PRATHAPAN (TKM20MCA2040)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Master of Computer Applications, is a bonafide record of the project work carried out by her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor Head of the Department External Examiner

ACKNOWLEDGEMENT

First and foremost, I thank GOD almighty and my parents for the success of this project. I owe sincere gratitude and heart full thanks to everyone who shared their precious time and knowledge for the successful completion of my project.

I express my sincere gratitude to **Dr. T A Shahul Hameed**, Principal of T.K.M College of Engineering for giving me an opportunity to present my project. With a profound sense of gratitude, I would like to express my heartfelt thanks to **Dr. Fousia.M.Shamsudeen**, Head of the Department, for providing me with best facilities and motivated me throughout the work of my project.

I profusely thank all other faculty members in the department and all other members of TKM College of Engineering, for their guidance and inspirations throughout my course of study. I owe my thanks to my friends and all others who have directly or indirectly helped me in the successful completion of this project.

SREELEKSHMI PRATHAPAN

ABSTRACT

In computing and operations research, the ant colony optimization algorithm (ACO) is a probabilistic approach to addressing problems that can often be reduced to locating optimal paths via graphs. In this metaphor, robotic ants stand in for the multi-agent techniques that were modelled after the actions of real ants. In the world of biological ants, pheromone-based communication is often the norm. Combinations of artificial ants and local search algorithms have become the de facto standard method for many optimization tasks requiring graphs, such as vehicle routing and internet routing. In order to approximate solutions to hard optimization problems, a population-based metaheuristic known as ant colony optimization (ACO) can be applied. In ACO, an optimization issue is posed, and a colony of virtual "ants" searches for a solution. In this paper, we apply the ACO algorithm to the puzzle-solving method commonly known as sudoku. This strategy eliminates the need for any additional algorithms beyond the one used to solve the original problem. This one will help us save time and find answers to difficult problems with just a single mouse click..

.

CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

In 1979, a logical-puzzle game known as "Number Place" was introduced under the name Sudoku. In 1984,the puzzle company brought it to Japan's attention. The game was then renamed "Sudoku" as a result. The issue started to be discussed in the West.The first sudoku grid appeared in The Times in 2004, and it was solved for the first issue by a machine for the first time in 1997, thanks to the work of Hong Kong-based judge Wayne Gould. To speed up the process of generating new instances, Gould developed an automated method afterwards.Sudoku is here to stay. It is currently being covered as a global phenomena in several newspapers extra to the current crosswords A 9*9 grid can be further subdivided into nine 3*3 grids to form the barebones version of Sudoku.Although it is obvious that the issue may be expanded to larger grids, for the time being we are just interested in the most well-known. You must use numbers from 1 to 9 to fill in all 9 digits in each row, column, and 3 3 subgrid in this problem. The number of approaches required to complete a Sudoku grid determines its difficulty more so than the number of cell values shown to the player at the outset of the puzzle. Three rows and three columns of cells (or squares) make up an order-n = 3 Sudoku problem, and these cells are separated into three subgrids called boxes. A unit, or row, column, or box in a table is made up of exactly nine cells. The issue can be resolved by permuting the numbers 1 through 9 in every unit, or every row, column, and box.

Each individual cell is a part of exactly three groups, giving it twenty neighbours (16 neighbours in the proper rows and columns plus 4 cells occupying the same box). This gives every cell 30 mutually-similar neighbours. The cell's units are the rows, columns, and box that contain it. The Latin Square Completion problem, of which Sudoku is a reduced form, is NP-complete. Consequently, this problem has been elevated to the status of a useful barometer, and several approaches have been devised to address it. As an alternative to, say, developing a big number of algorithms for playing a certain game, we could investigate methods that have the potential to be used in a wider variety of contexts. While we present our method in the context of Crosswords, we show that because it is generic and doesn't rely on any game-specific "hints," it may be used to a wide range of puzzle games. There are exactly twenty peers (or the other units) in each of the three units that make up a cell.. The row, column, and box in which the cell is placed are known as the units. Sudoku is an NP-complete problem since it is a reduction of the Latin Square Completion issue. Because of this, the subject offers a valuable benchmarking problem, and several methods have been devised to address it. For instance, instead of creating several algorithms to play a particular game, we should explore strategies that are applicable to a larger range of problems. Although the algorithm presented here is proven in the setting of Sudoku, we later demonstrate that its independence that is, game - specific "hints" makes it applicable to a variety of puzzle games.

Though it may at first glance that such puzzle games lack "games are a formidable obstacle for general-purpose AI techniques because of their "real world" relevance; "Puzzle games, 2D arcade games, text adventures, 3D action-adventure games, etc., should all be able to find a voice in a common set of performance metrics and contests.. This is the gold standard for evaluating the breadth and depth of an AI's capabilities and logic." The fact that our method is not "general purpose" does, however, underline the relevance of the gaming sector.

In order to reduce the number of necessary algorithms for the game, the Ant Colony Optimization Algorithm (ACO) is employed. Marco Dorigo came up with the concept of "Ant Colony Optimization" in the 1990s, and the notion was clearly influenced by the grazing habits of ant colonies.. Eusocial insects like ants put the wellbeing of the colony ahead of that of its members. Pheromones, physical contact, and sound all play roles in their communication. Pheromones, which are organic chemical molecules emitted by ants, encourage social behaviour in other members of the same species. These substances, which function like hormones outside the body of the secretor, can have an effect on a person's behaviour. The vast majority of ants are ground-dwellers, and these ants use the soil's surface to produce pheromone trails, which other ants can then follow (smell)..

The idea behind ACO is to travel as little distance as possible while searching for food, just like a colony of ants might. As they forage for food, the ants wander aimlessly away from their colonies. This random investigation reveals alternate ways of getting from the nest to the food supply. Now, the ants only bring back a certain percentage of the food with the appropriate pheromone concentration, and this percentage varies based on the quantity and quality of the meal. Based on the results of these pheromone trials, future ants would be led to the food source by one of several predetermined routes. This chance is obviously affected by the pheromone's concentration and its pace of evaporation. The rate at which pheromones dissipate is a distinct factor in determining the duration of each segment.

I

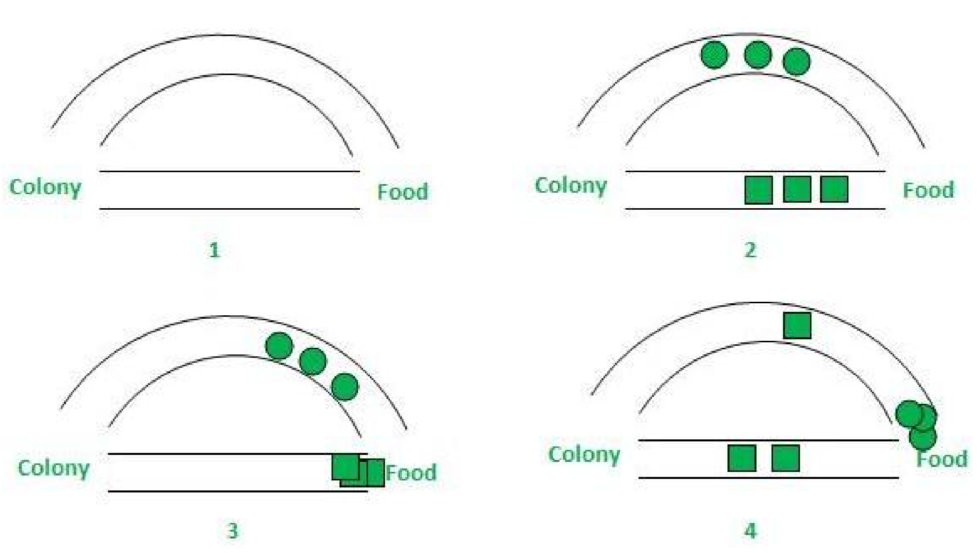


fig1

There are two probable routes from the food supply to the ant nest, as depicted in the diagram above for simplicity's sake.

1.1 PROBLEM STATEMENT

The game of sudoku rewards hours of consistent effort. You get better at playing the more you do it. If you play Sudoku every day, you'll start to notice that you have practically a "sense memory" for the problem grid; you'll start to notice patterns appearing and improve your ability to act more swiftly when an opportunity arises.

The more Sudoku circumstances you encounter via many hours of daily play, the more likely it is that you will be able to solve the puzzle in any circumstance.

The conventional method of backtracking is a helpful algorithm for finding incremental solutions to recursive issues. In most cases, retracing entails starting with one potential answer and moving on to another if the first one doesn't work. This process continues until you find a solution that does. Problem-solving with this algorithmic method requires substantially more time. The algorithm becomes more sophisticated since the hardest issues take the most time to solve.

1.2 OBJECTIVE

The primary goal of this project is to solve sudoku problems with a single technique known as Ant Colony Optimization. On the hardest, massive cases of Sudoku,The state-of-the-art technique is greatly outperformed by an approach based on ant colonies. In comparison to typical backtracking approaches, we demonstrate that our algorithm searches the solution space far more effectively and solves problems in less time.

Chapter 2

Literature Survey

A literature review is an in-depth analysis of previously published works on a given subject. It is common practise to conduct a literature review after establishing a set of research questions in order to guide the subsequent search for and analysis of the literature that might provide answers to those questions. Literature reviews are useful because, by re-examining previous research, novel conclusions can often be drawn. The term literature review refers to a systematic and explanatory overview of a topic's existing body of scholarly writing on the subject. f There are two types of literature reviews that students may be asked to write in college: one is written as a stand-alone assignment for a class, and the other is written as an introduction to or background for a lengthier piece of writing, such as a thesis or research report. The type of review you are writing will influence your choice of emphasis, perspective, and the type of hypothesis or thesis argument you make. Reading published literature reviews or the introductory chapters of theses and dissertations in your field might help you better comprehend the distinctions between these two types of research. Examine their argument structure and take notice of how they deal with the issues..

2.1 PURPOSE OF THE LITERATURE REVIEW

1. It gives readers easy access to research on a particular topic by selecting high-quality articles or studies that are relevant, meaningful, important and valid and summarizing them into one complete report.
2. It provides an excellent starting point for researchers beginning to do research in a newIt ensures that researchers do not duplicate work that has already been done.

3. It can provide clues as to where future research is heading or recommend areas on which to focus.
4. It highlights the key findings.
5. It identifies inconsistencies, gaps and contradictions in the literature.
6. It provides a constructive analysis of the methodologies and approaches of other researchers.

2.2 RELATED WORKS

[1] J. Laire

We start our inquiry into solving Sudoku by employing a "traditional" backtracking technique. The Exact Cover Problem can be explained in terms of a constraint fulfilment problem as follows: a binary matrix's rows where each column adds up to one. The book refers to Algorithm X (DLX), In order to describe Knuth's "brute force" backtracking approach for Exact Cover, we can think of the links in the algorithm as "dancing." As any Sudoku can be converted into an instance of Exact Cover, DLX provides a viable strategy for solving the puzzle..

[2] M. Dorigo and M. Birattari

Suggests a different approach based on limitations and a search strategy. Intriguing methods for resolving Sudoku also include the artificial bee colony algorithm, formal logic, constraint programming, evolutionary algorithms, particle swarm optimization, simulated annealing, tabu search, and entropy minimization. This diversity of solutions illustrates that Sudoku is a challenging, yet conceptually easy, testing ground for evaluating strategies for situations requiring complicated reasoning.

[3] A. Inkala

The prototypical ACO strategy, originally called the "Ant system," was developed to solve the "Traveling Salesman" tracking problem. The issue is that An ant's choice of which edge to follow between two cities is influenced both by the edge's closeness to the ant's current location and by the concentration of pheromones there. This strategy fuses the autocatalytic power of the global pheromone network with a greedy local search heuristic. Every ant maintains a tabu of all the places it has ever been. prohibits its return to any of the specified locations..

Once every city has been visited, an ant's global pheromone production is inversely correlated with the length of its journey; shorter tours produce more pheromone. The global pheromone matrix evaporates when all ants have finished this process, progressively removing any traces of less-than-ideal tours that may have survived over time

[5] T. Mantere

Mantere offers a hybrid ACO/genetic algorithm approach to solving Sudoku by fusing global (evolutionary) search with greedy local (ACO-based) search. , which employed a variation of the artificial bee colony method to solve 9 9 Sudoku problems using a nature-inspired approach. Schiff and Sabuncu give relatively recent work on the application of ACO to Sudoku, however the method's effectiveness in both cases is limited. Although the algorithm was able to solve a number of difficult problems, its runtime performance was rather poor, taking over 6 minutes on average to resolve complex problems

[5] M. Ercsey-Ravasz and Z. Toroczkai

Together with the techniques of Knuth and Norvig, Musliuetal's iterated local search algorithm with constraint programming reflects the state of the art in stochastic search algorithms for the Sudoku problem.

Chapter 3

Methodology

We employ ant colony optimization to solve our sudozu puzzle more quickly and with less complexity (ACO). It is a population-based systematic approch that can be applied to identify roughly correct answers tochallenging optimization issues. In ACO, a group of computer programmes known as "artificial ants" look for viable answers to a given optimization issue.

3.1 SYSTEM ARCHITECTURE

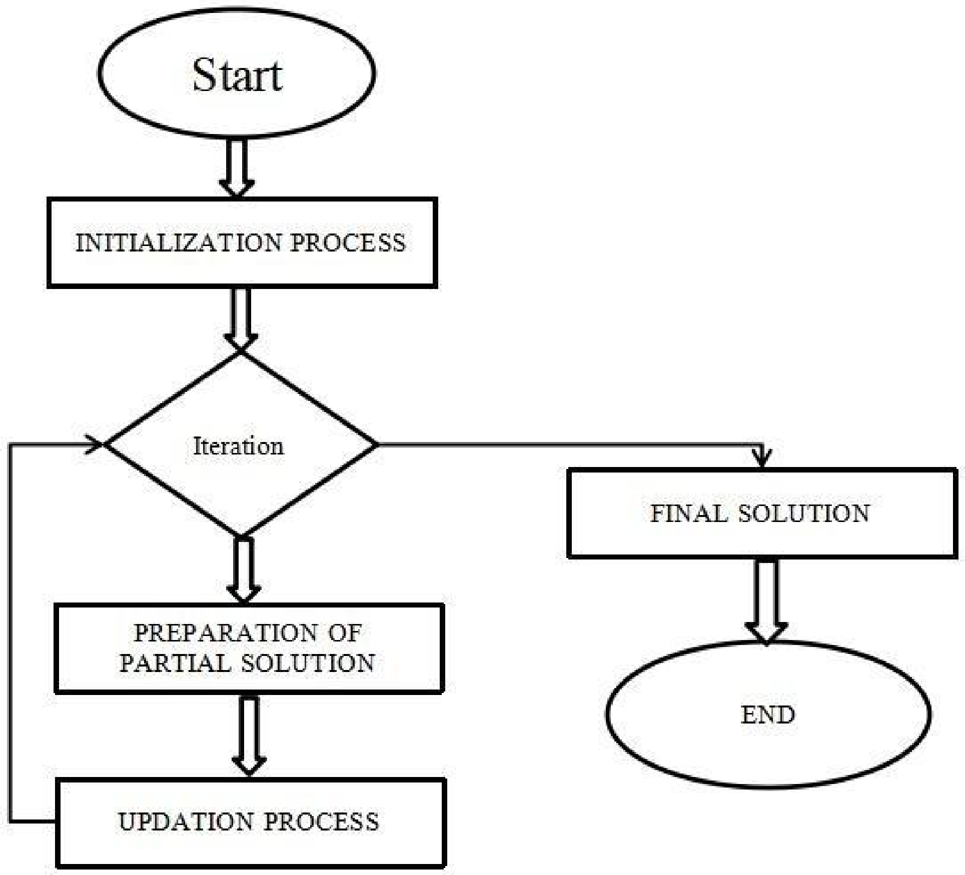


Fig 3.1 Work Flow of the System

Steps for Ant Colony Optimization Process:

The following are the procedures or steps used for optimization ,

Initialization process

Evaluation of the selected subsets

Construction process

Update process

Decision process

- **Initialization Process**

Initialization of the population is commonly done by seeding the population with random values. The fitness value is proportional to the performance measurement of the function being optimized.

- **Evaluation of the selected subsets**

Assign any ant randomly to one feature and visiting features, each ant builds solutions completely. In this step, the evaluation criterion is mean square error (MSE) of the classifier.

- **Construction process**

The basic ingredient of any ACO algorithm is a constructive heuristic for probabilistically constructing solutions . A constructive heuristic assembles solutions as sequences of elements from the finite set of solution components.

- **Update process**

Decrease pheromone concentrations of nodes then, all ants deposit the quantity of pheromone on graph. Finally, allow the best ant to deposit additional pheromone on nodes.

- **Decision process**

The objective of this module is to find the global best ant so far over all iterations in and to record the selected consequent combination of this ant once it is found in an iteration

3.2 ALGORITHMS USED

The Ant Colony Optimization Algorithm (ACO) is the only algorithm used here to solve sudoku.

3.2.1 ANT COLONY OPTIMIZATION ALGORITHM (ACO)

The population-based meta - heuristic optimization ant colony optimization (ACO) can be used to roughly solve complex optimization problems. In ACO, an ensemble of virtual agents dubbed "artificial ants" looks for optimal solutions to a given optimization challenge. When applied to an optimization problem, ACO reduces it to one of locating the optimal route through a weighted graph. In order to solve problems, the artificial ants, hereafter referred to as ants, must traverse the graph. A pheromone model, which is a set of variables associated with graph elements (either nodes or edges) whose values are adjusted at runtime by the ants, biases the process of constructing stochastic solutions.

Through the deployment of artificial ants that navigate a fully connected construction graph, ACO uses algorithms to solve combinatorial optimization issues. Each instantiation choice variable (Xi=vji), also known as a solution component, is denoted by the letters cij. The letter C stands for the totality of all elements of a practical solution. The components C are then connected to either the set of vertices V or the set of edges E to create the construction graph GC (V,E).
Each element has a pheromone trail value (ij) attached to it (cij). It should be noticed that the algorithm iteration (t:ij=ij) regularly affects pheromone values (t) You may express the likelihood of each element in the solution using pheromone values. The ACO algorithm uses and modifies pheromone values during the search.

The ants go from vertex to vertex along the edges of the construction graph, gradually constructing a solution using the knowledge from the pheromone values. When the ants go across the edges or vertices of the components, they also leave some pheromone behind. Depending on the calibre of the detected solution, the amount of pheromone that is deposited may change. The information included in the pheromones directs successive ants to more productive areas of the search field.

The ACO metaheuristic is:

```
Set parameters, initialize pheromone trails
SCHEDULE_ACTIVITIES
  ConstructAntSolutions
  DaemonActions    {optional}
  UpdatePheromones
END_SCHEDULE_ACTIVITIES
```

Ant colony optimization (ACO), a population-based metaheuristic, can be used to approximatively resolve difficult optimization problems. A set of software agents known as artificial ants search for good solutions to an optimization problem in ACO. Using ACO, the optimization problem is transformed into the process of finding the best path on a weighted graph. The artificial ants, henceforth referred to as ants, travel across the graph to piece together solutions. The process of creating stochastic solutions is biassed by a pheromone model, which is a group of variables connected to graph elements (either nodes or edges) whose values are modified at runtime by the ants.

ACO uses artificial ants to solve combinatorial optimization problems by navigating a fully connected construction graph, which is described as follows. The letters cij are used to identify each instantiation choice variable (Xi=vji), also known as a solution component. The collection of all components of a workable solution is represented by the letter C. The construction graph GC is then formed by connecting the components C to either the set of vertices V or the set of edges E. (V,E).

Each element has an associated pheromone trail value (ij) (cij). It's important to keep in mind that pheromone values often change with each iteration of the algorithm (t:ij=ij). (t) Each component's likelihood in the solution can be expressed as a value of a pheromone. The ACO algorithm takes advantage of and adjusts pheromone values while searching.

Based on the pheromone values, the ants move from node to node along the edges of the construction graph, gradually erecting the solution. The ants also leave a certain quantity of pheromone behind when they cross the edges or vertices of the components. Pheromone deposition rates are sensitive to the accuracy of the detected solution. Subsequent ants can use the pheromones' information to focus their efforts in more fruitful parts of the search area.In the Traveling Salesman Problem, the ideal tour length is represented by the formula max=1/(L). It is possible to estimate L using Lbs if one does not know L exactly. The method is reset after a fixed number of iterations if no progress has been seen, and the original value of the trails is set to max.

3.3 TECHNOLOGIES USED

3.3.1 PYTHON 3

Python is a high-level, interpreted scripting language that was developed by Guido van Rossum at the Netherlands' National Research Institute for Mathematics and Computer Science. After the original version was published in the alt.sources newsgroup in 1991, version 1.0 was made available in 1994.

The 2.x series of versions predominated the market since the release of Python 2.0 in 2000 until December 2008. Afterwards, the development team released version 3.0, which featured some minor but important changes that were incompatible with the 2.x series. Some features of Python 3 have been backported to Python 2, making the two versions of the language functionally equivalent. But they still can't coexist in the wild..

With fresh versions for each, Python 2 and Python 3 have both been regularly maintained current. As of this writing, the most recent versions are 2.7.15 and 3.6.5. However, Python2 will no longer be supported as of January 1, 2020, which is the official End Of Life date. If you are new to Python, as you will be in this session, it is advisable that you concentrate on Python 3.

A core development team at the Institute continues to support Python, and the community still refers to Guido as the BDFL (Benevolent Dictator For Life). By the way, Python comes from the British comedy troupe Monty Python's Flying Circus, which Guido undoubtedly enjoyed and still enjoys. Several references to Monty Python's comedic films and sketches can be found throughout the Python documentation.Python is widely used.

Over the past few years, Python's popularity has skyrocketed. Seventh in the 2018 Stack Overflow Developer Survey's rankings of the most popular and sought-after technologies was Python. Top-tier software development companies utilise Python on a regular basis all over the world. Python is the most popular programming language in the world, according to data from Dice, and the Popularity of Programming Language Index lists Python as one of the top skills to have. As a result of Python's acceptability and success as a programming language, developers are in great demand and well-paid.

Python is interpreted (Interpreted)

The source code you write must be translated into machine code, the language of your computer's processor, because many languages are compiled before they can be run. Programs created in an interpreted language are delivered to an interpreter, who promptly executes them. By bypassing the intermediate compilation stage and just inputting your code and running it, you may reduce the length of the development cycle. Interpreted languages may have certain disadvantages, one of which being execution speed. Generally speaking, compiled programmes execute more quickly than interpreted ones since they are written in the processor's native language. For some applications that need a lot of computing, such graphics processing or sophisticated arithmetic calculations, this might be limiting. However, in practise, the bulk of programmes' changes in execution speed are measured in milliseconds, or at most, seconds, and are hardly noticeable to a human user. The speed of coding in an interpreted language is often advantageous for most applications. Many languages are compiled, so before they can be executed, the source code you write must be converted into machine code, the language of your computer's processor. An interpreter receives programmes written in an interpreted language and executes them immediately. A shorter development cycle results from skipping the intermediary compilation phase and simply typing your code and running it. The speed of execution is one possible drawback of interpreted languages. Ones that are compiled into the computer processor's native language typically run faster than interpreted programmes. This could be constricting for some applications that require a lot of computing, such graphics processing or complex math crunching. However, in reality, the change in execution speed for the majority of programmes is measured in milliseconds, or seconds at most, and is barely perceptible to a human user.For the majority of applications, the speed of coding in an interpreted language is usually worth it.

Python is No Cost

The OSI-approved open-source licence under which the Python interpreter was developed makes it completely free to install, use, and distribute—even for commercial gain. The interpreter is available on practically all platforms, including all editions of Unix, Windows, macOS, mobile devices, tablets, and virtually everything else you can think of. There is a version available, even for the 12 OS/2 users left.

Python Can Be Ported

Python code works on every platform that has the Python interpreter installed because it is interpreted rather than compiled into native machine instructions. (This applies to all interpreted languages; Python is not an exception.)

Python is Easy

Python is relatively uncluttered compared to other programming languages, and its designers purposefully left it that way. The amount of keywords or reserved words in a language can be used to determine its level of complexity. These are terms that the compiler or interpreter reserves for unique meaning because they signify certain built-in capability of the language.

Python 2 features 31 keywords, compared to 33 in Python 3. Comparatively, Visual Basic has over 120, C++ has 62, Java has 53, and there are over 120 in Java, but these later instances likely vary a little depending on implementation or dialect. The structure of Python code is clear and straightforward, making it simple to learn and read. As you shall see, the language specification really enforces a readable code structure.

3.3.2LIBRARIES

A Python library is a piece of reused code that you might wish to add to your projects or programmes. When compared to languages like C++ or C, Python libraries are not context-specific. Here, a collection of essential modules is roughly referred to as a "library." Therefore, a library is just a collection of modules. A package management, such as rubygems or npm, can be used to install a library.

Standard Library for Python

The precise syntax, tokens, and semantics of Python are collected in the Python Standard Library. It includes the standard Python distribution. When we started with an introduction, we mentioned this. It manages functions like I/O and other fundamental modules and is built in C. The combination of all these features makes Python the language it is. The core of the standard library is made up of more than 200 core modules. Python already includes this library. However, you may also access a developing collection of several thousand components via the Python Package Index in addition to this library (PyPI)

3.3.3VS CODE

In Visual Studio Code, the simplicity of a source code editor is paired with powerful developer tools, such as IntelliSense code completion and debugging. An editor, first and foremost, gets out of your way. The pleasantly frictionless edit-build-debug cycle allows you to spend more time implementing your ideas rather than dealing with your surroundings.

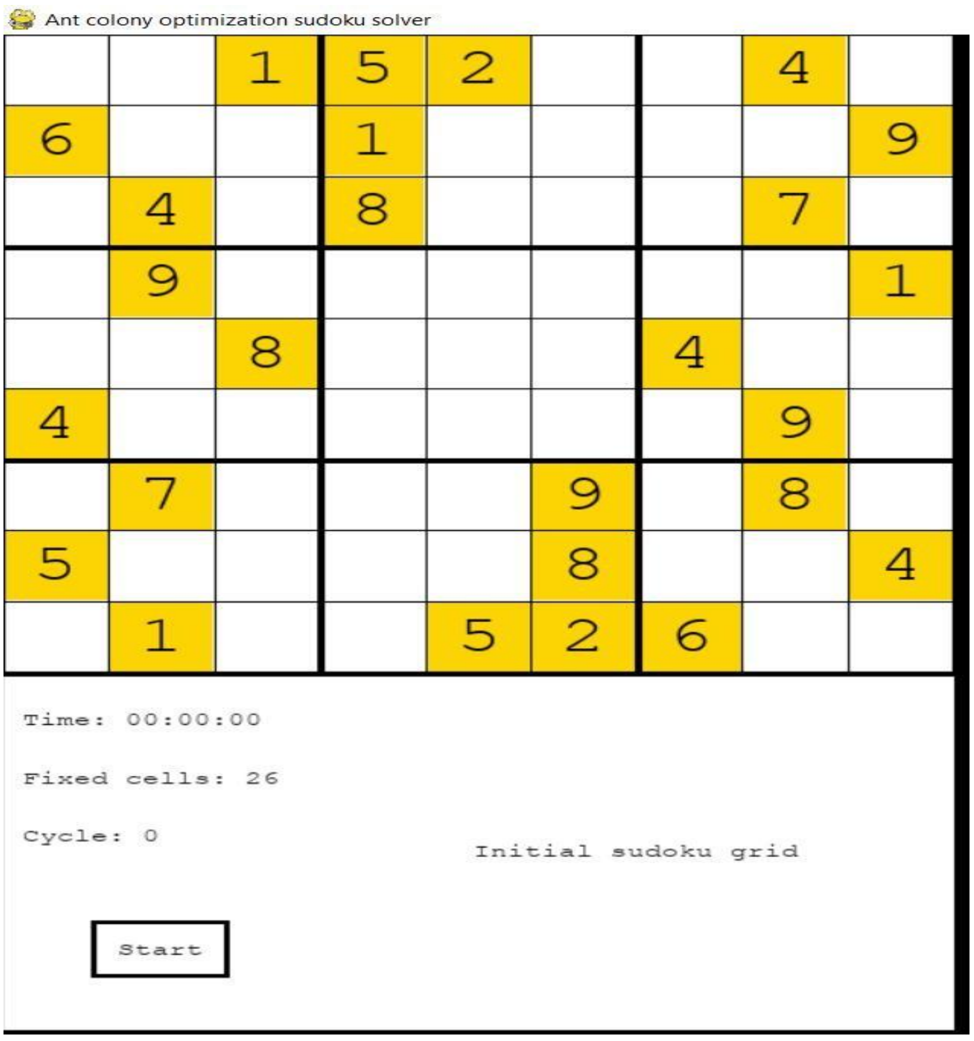
Visual Studio Code's main functionality, a lightning-fast source code editor, makes it perfect for regular use. With its extensive language support and tools like syntax highlighting, bracket matching, auto-indentation, box selection, snippets, and more, Visual Studio Code makes it simple for you to get started right away and be productive. Thanks to simple modification, intuitive keyboard shortcuts, and community-contributed keyboard shortcut mappings, you can navigate your code with ease.

Tools with more code knowledge than just text blocks are typically helpful for serious development. The capabilities that come included with Visual Studio Code include code refactoring, deep semantic code interpretation and navigation, and IntelliSense code completion. Debugging is a talent that the competent use when coding becomes more difficult. Since debugging is typically the one thing coders miss the most in a limited coding environment, we incorporated it. Visual Studio Code has an interactive debugger that lets you step through source code, examine variables, view call stacks, and execute console commands. Additionally, to perform common tasks and speed up everyday operations, VS Code integrates with build and scripting tools.

Chapter 4

Result and Discussion

The bias value for each iteration is the major focus of the outcome. Each iteration of the row and column searches updates the bias value.When the algorithm discovers the current best random value (ant) globally over all iterations, it stores the chosen subsequent combination of this ant.Studies showed our unique technique greatly beats existing methods on the most difficult, big instances of Sudoku, and we present proof that our method enables a much more efficient search of the solution space than standard backtracking algorithms for similar problems. If you're dealing with something easier or smaller



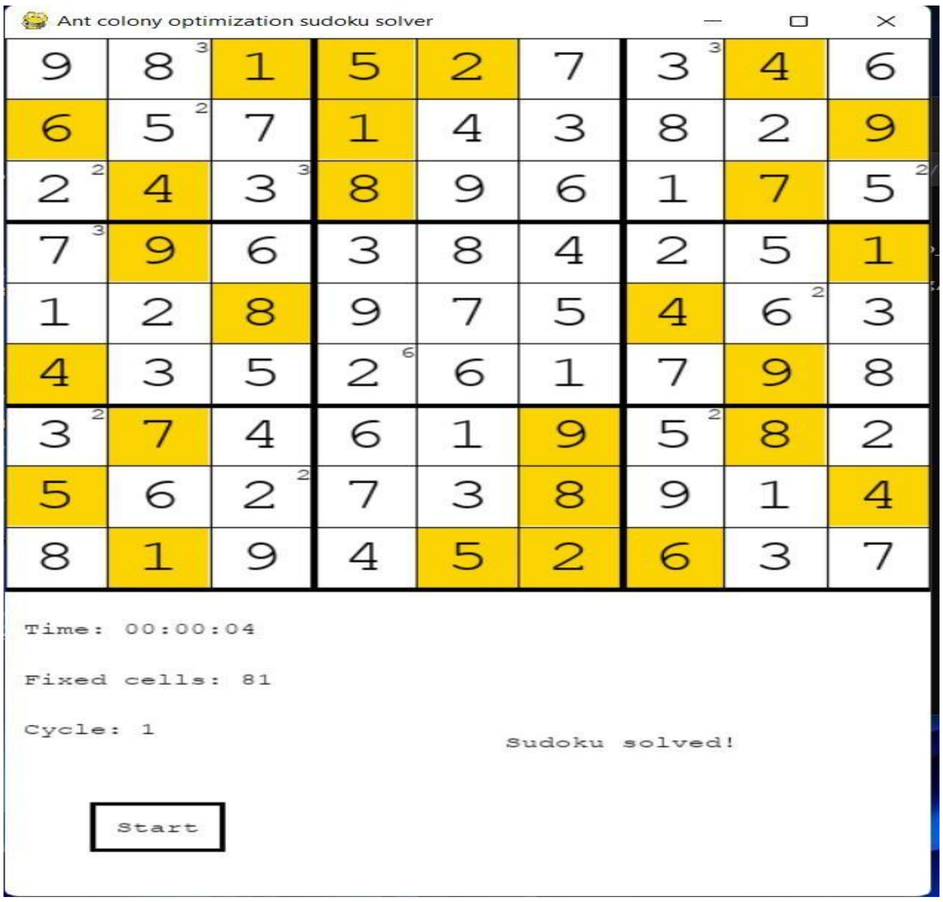
		1	5	2			4	
6			1					9
	4		8				7	
	9							1
		8				4		
4							9	
	7				9		8	
5					8			4
	1			5	2	6		

```
Time: 00:00:00

Fixed cells: 26

Cycle: 0                Initial sudoku grid
```

Fig 4.1 Input Image



Ant colony optimization sudoku solver

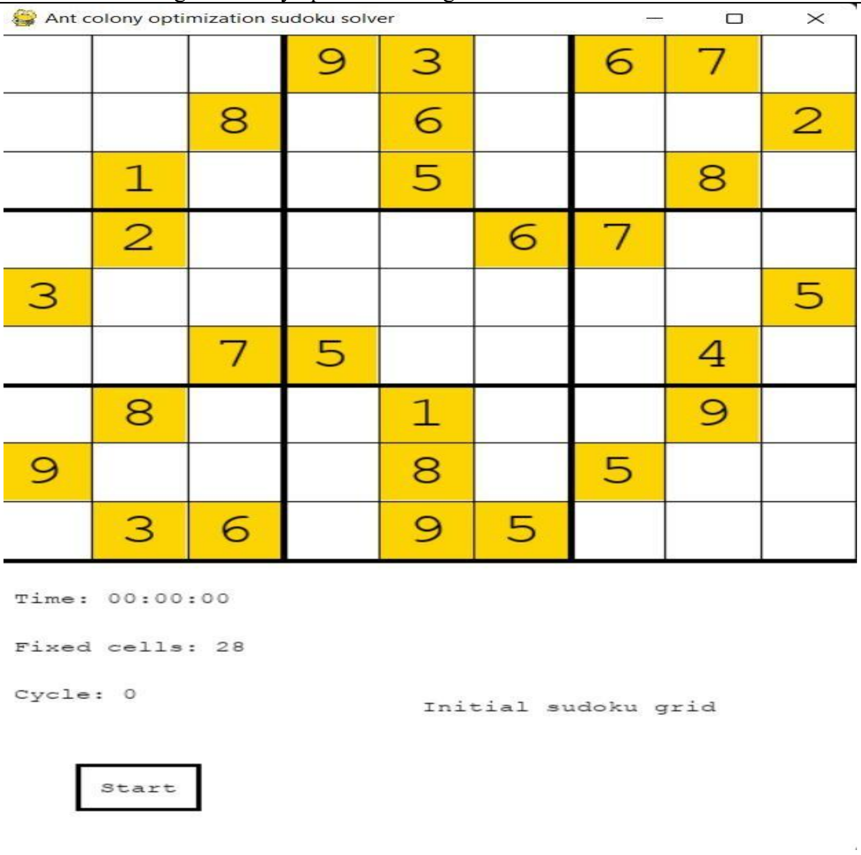
9	8^{3}	1	5	2	7	3^{3}	4	6
6	5^{2}	7	1	4	3	8	2	9
2^{2}	4	3^{3}	8	9	6	1	7	5^{2}
7^{3}	9	6	3	8	4	2	5	1
1	2	8	9	7	5	4	6^{2}	3
4	3	5	2^{6}	6	1	7	9	8
3^{2}	7	4	6	1	9	5^{2}	8	2
5	6	2^{2}	7	3	8	9	1	4
8	1	9	4	5	2	6	3	7

```
Time: 00:00:04

Fixed cells: 81

Cycle: 1                              Sudoku solved!
```

Fig 4.2 solved simple grid



Ant colony optimization sudoku solver

			9	3		6	7	
		8		6				2
	1			5			8	
	2				6	7		
3								5
		7	5				4	
	8			1			9	
9				8		5		
	3	6		9	5			

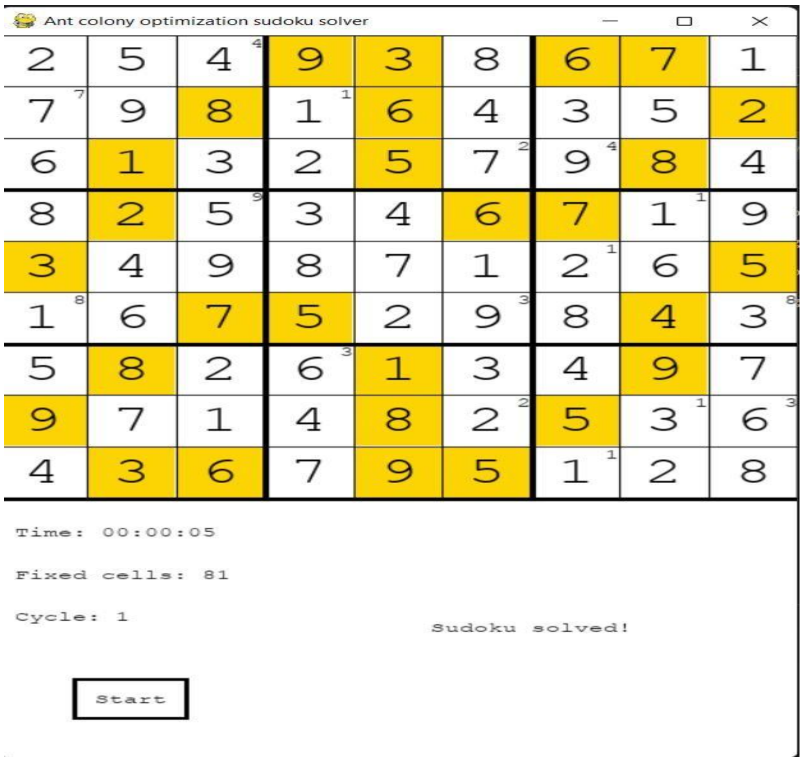
```
Time: 00:00:00

Fixed cells: 28

Cycle: 0                    Initial sudoku grid
```

Start

Fig 4.3 initail grid of really hard grid



Ant colony optimization sudoku solver

2	5	4	9	3	8	6	7	1
7	9	8	1	6	4	3	5	2
6	1	3	2	5	7	9	8	4
8	2	5	3	4	6	7	1	9
3	4	9	8	7	1	2	6	5
1	6	7	5	2	9	8	4	3
5	8	2	6	1	3	4	9	7
9	7	1	4	8	2	5	3	6
4	3	6	7	9	5	1	2	8

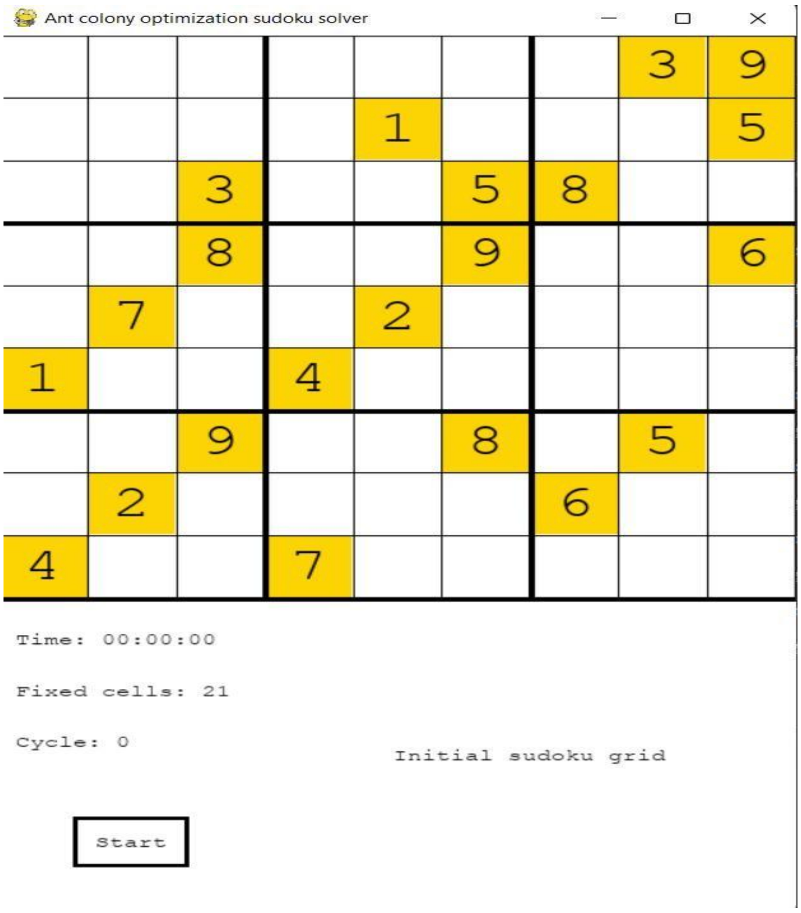
```
Time: 00:00:05

Fixed cells: 81

Cycle: 1                    Sudoku solved!
```

Start

Fig4.4 solved really hard grid



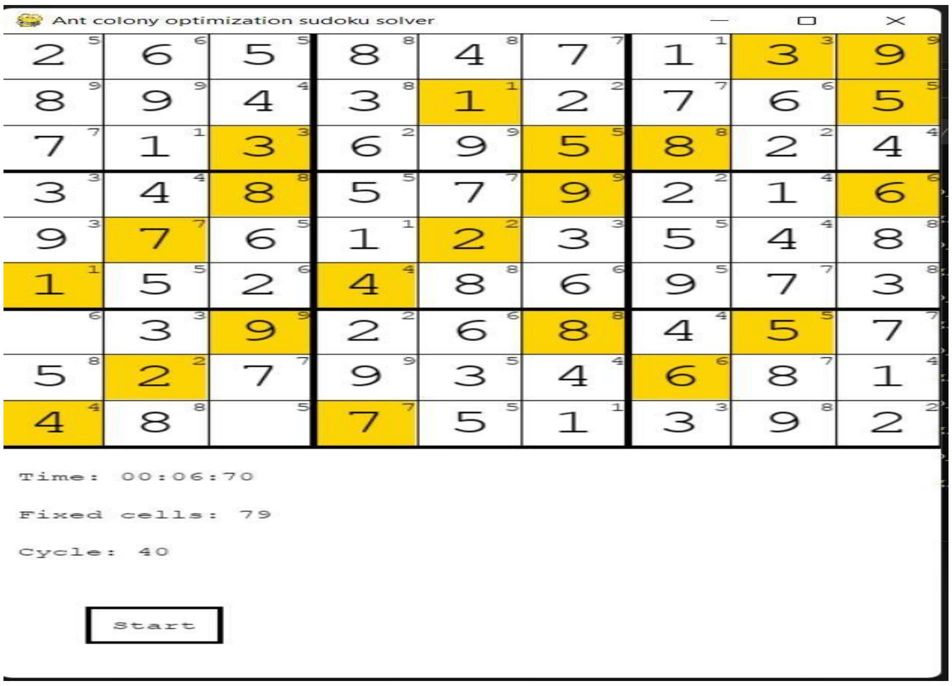
							3	9
				1				5
		3			5	8		
		8			9			6
	7			2				
1			4					
		9			8		5	
	2					6		
4			7					

```
Time: 00:00:00

Fixed cells: 21

Cycle: 0                Initial sudoku grid
```

Fig 4.5 initial grid of golden nuggets



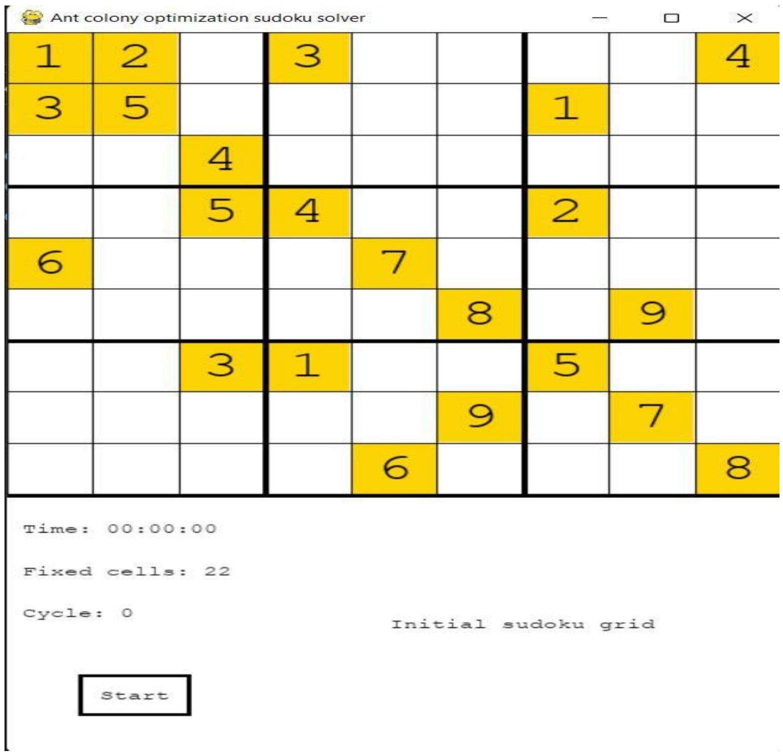
2^{5}	6^{6}	5^{5}	8^{8}	4^{8}	7^{7}	1^{1}	3^{3}	9^{9}
8^{9}	9^{9}	4^{4}	3^{8}	1^{1}	2^{2}	7^{7}	6^{6}	5^{5}
7^{7}	1^{1}	3^{3}	6^{2}	9^{9}	5^{5}	8^{8}	2^{2}	4^{4}
3^{3}	4^{4}	8^{8}	5^{5}	7^{7}	9^{9}	2^{2}	1^{4}	6^{6}
9^{3}	7^{7}	6^{5}	1^{1}	2^{2}	3^{3}	5^{5}	4^{4}	8^{8}
1^{1}	5^{5}	2^{6}	4^{4}	8^{8}	6^{6}	9^{5}	7^{7}	3^{8}
^{6}	3^{3}	9^{9}	2^{2}	6^{6}	8^{8}	4^{4}	5^{5}	7^{7}
5^{8}	2^{2}	7^{7}	9^{9}	3^{5}	4^{4}	6^{6}	8^{7}	1^{4}
4^{4}	8^{8}	^{5}	7^{7}	5^{5}	1^{1}	3^{3}	9^{8}	2^{2}

```
Time: 00:06:70

Fixed cells: 79

Cycle: 40
```

Fig 4.6 solved golden nuggets grid


Ant colony optimization sudoku solver

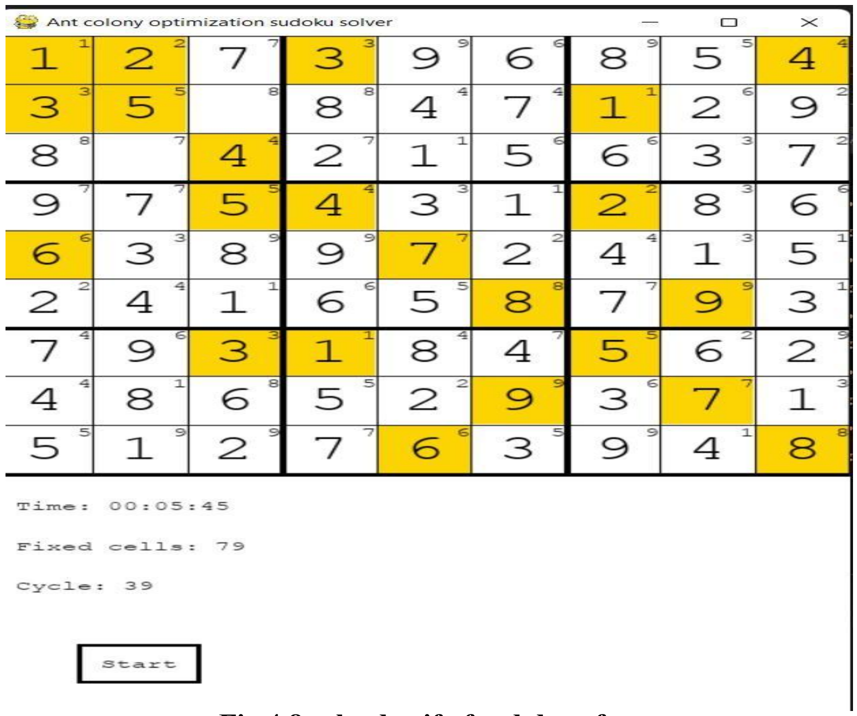
1	2		3					4
3	5					1		
		4						
		5	4			2		
6				7				
					8		9	
		3	1			5		
					9		7	
				6				8

```
Time: 00:00:00

Fixed cells: 22

Cycle: 0                         Initial sudoku grid
```

Fig 4.7 initial grid of red dawrf


Ant colony optimization sudoku solver

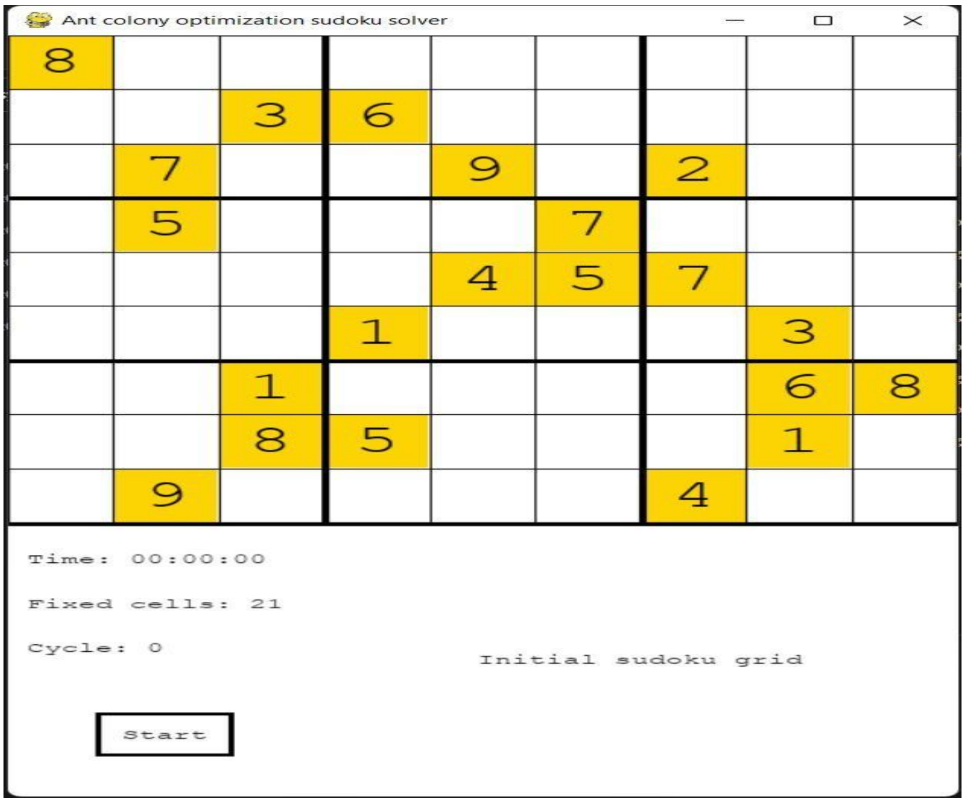
1	2	7	3	9	6	8	5	4
3	5		8	4	7	1	2	9
8		4	2	1	5	6	3	7
9	7	5	4	3	1	2	8	6
6	3	8	9	7	2	4	1	5
2	4	1	6	5	8	7	9	3
7	9	3	1	8	4	5	6	2
4	8	6	5	2	9	3	7	1
5	1	2	7	6	3	9	4	8

```
Time: 00:05:45

Fixed cells: 79

Cycle: 39
```

Fig 4.8 solved grif of red dwarf



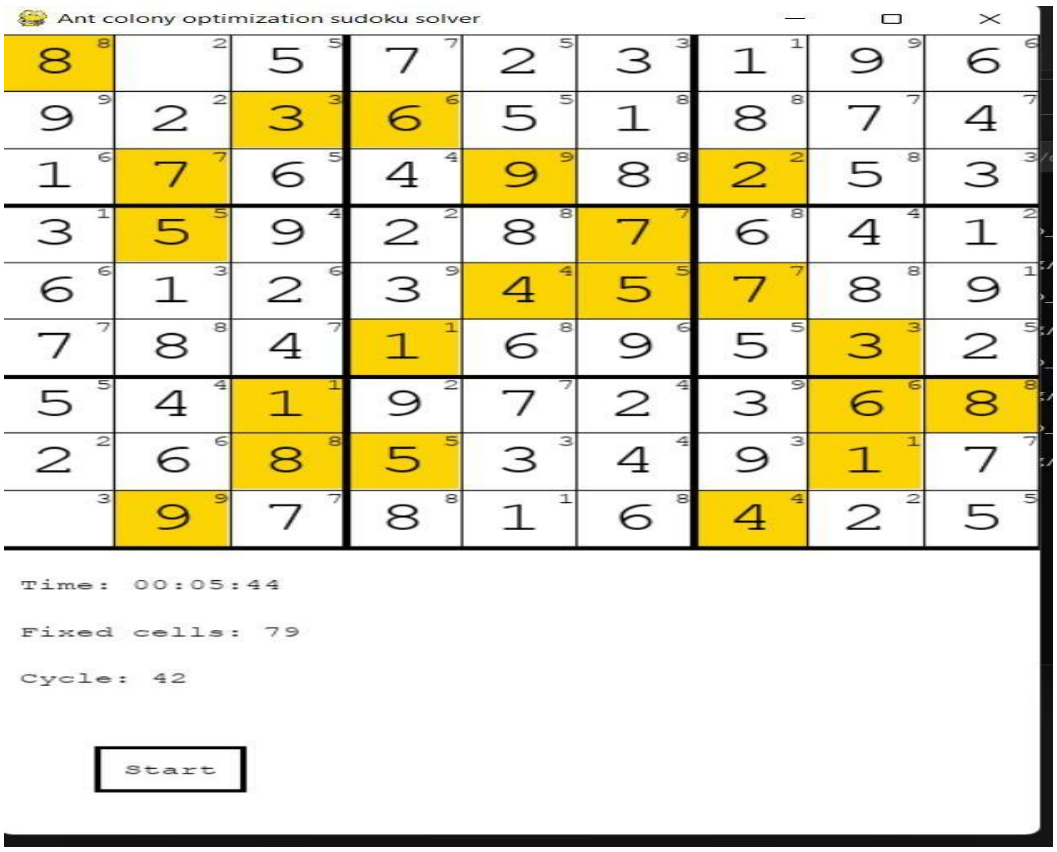
8								
		3	6					
	7			9		2		
	5				7			
				4	5	7		
			1				3	
		1					6	8
		8	5				1	
	9					4		

```
Time: 00:00:00

Fixed cells: 21

Cycle: 0                    Initial sudoku grid
```

Fig 4.9 initial grid of world hardest

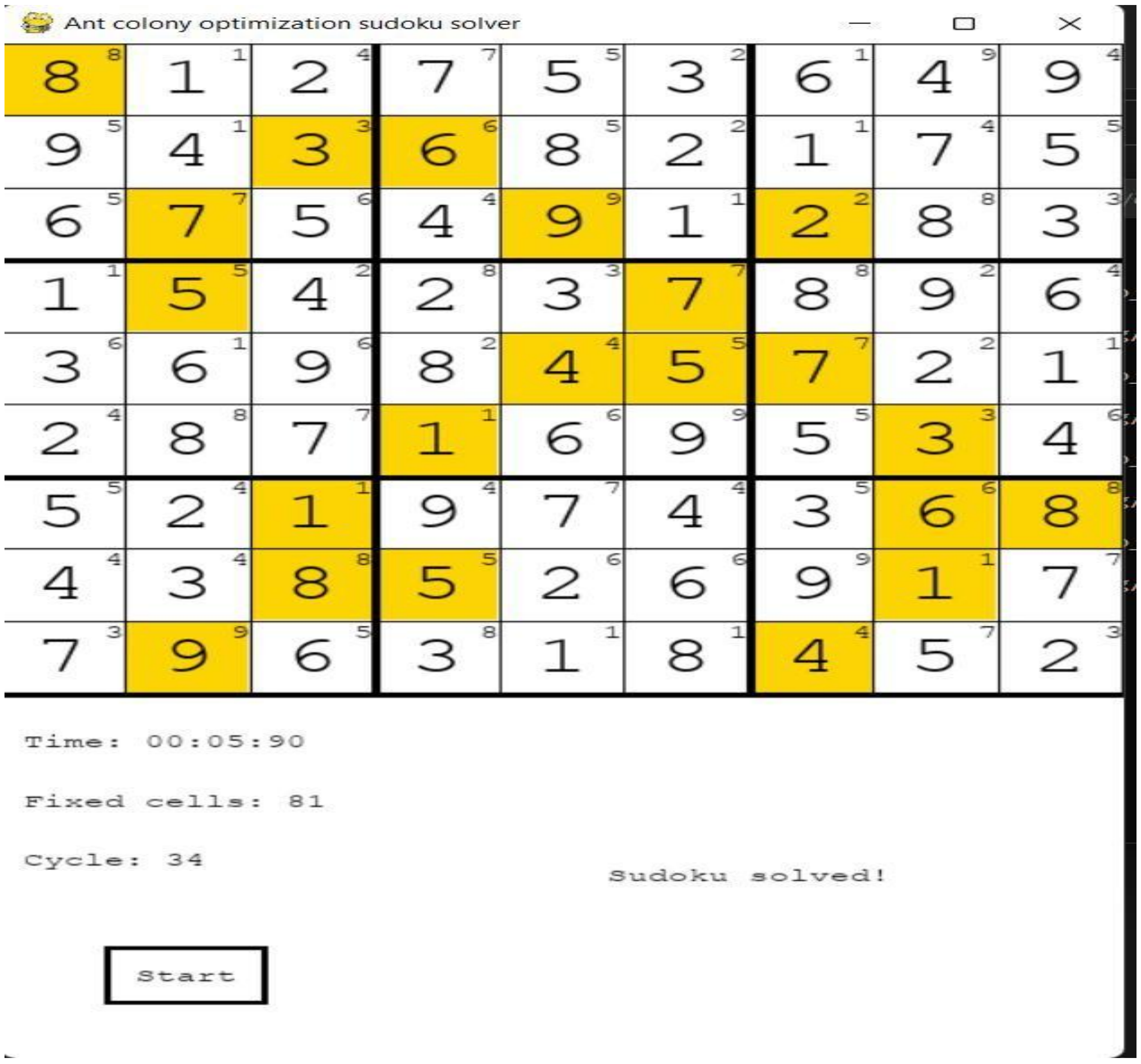


8		5	7	2	3	1	9	6
9	2	3	6	5	1	8	7	4
1	7	6	4	9	8	2	5	3
3	5	9	2	8	7	6	4	1
6	1	2	3	4	5	7	8	9
7	8	4	1	6	9	5	3	2
5	4	1	9	7	2	3	6	8
2	6	8	5	3	4	9	1	7
	9	7	8	1	6	4	2	5

```
Time: 00:05:44

Fixed cells: 79

Cycle: 42
```



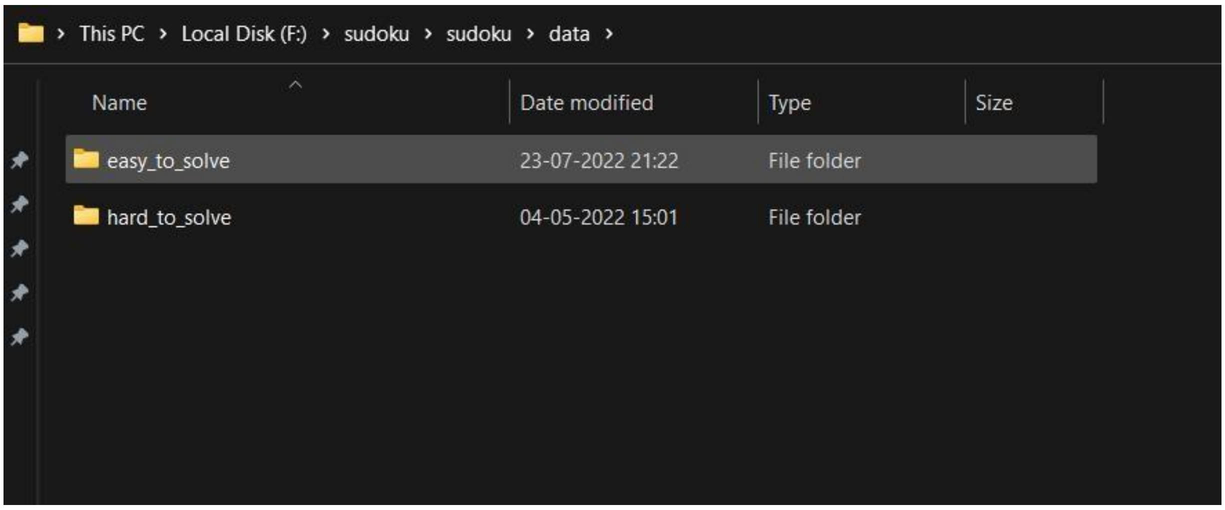
8	1	2	7	5	3	6	4	9
9	4	3	6	8	2	1	7	5
6	7	5	4	9	1	2	8	3
1	5	4	2	3	7	8	9	6
3	6	9	8	4	5	7	2	1
2	8	7	1	6	9	5	3	4
5	2	1	9	7	4	3	6	8
4	3	8	5	2	6	9	1	7
7	9	6	3	1	8	4	5	2

```
Time: 00:05:90

Fixed cells: 81

Cycle: 34
                                Sudoku solved!
```

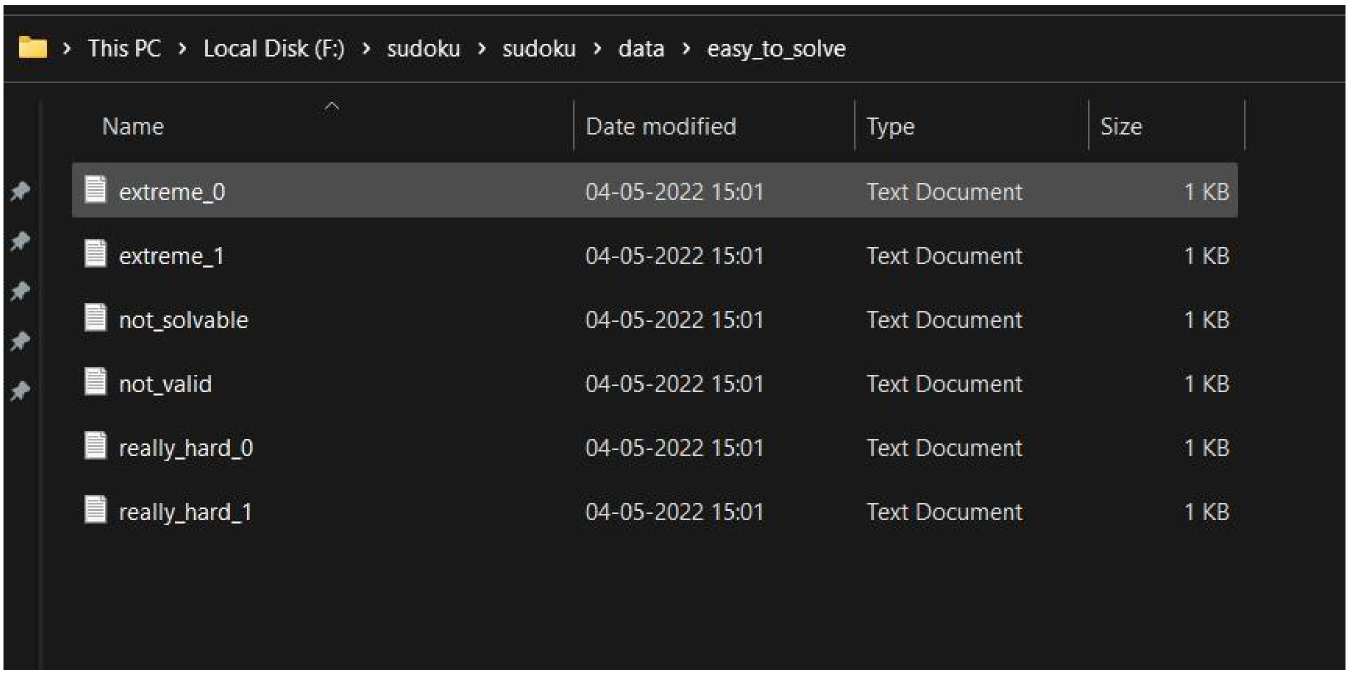
Fig 4.10 Solved grif of worlds hardest



This PC › Local Disk (F:) › sudoku › sudoku › data ›

Name	Date modified	Type	Size
easy_to_solve	23-07-2022 21:22	File folder	
hard_to_solve	04-05-2022 15:01	File folder	

Fig 4.12 data used

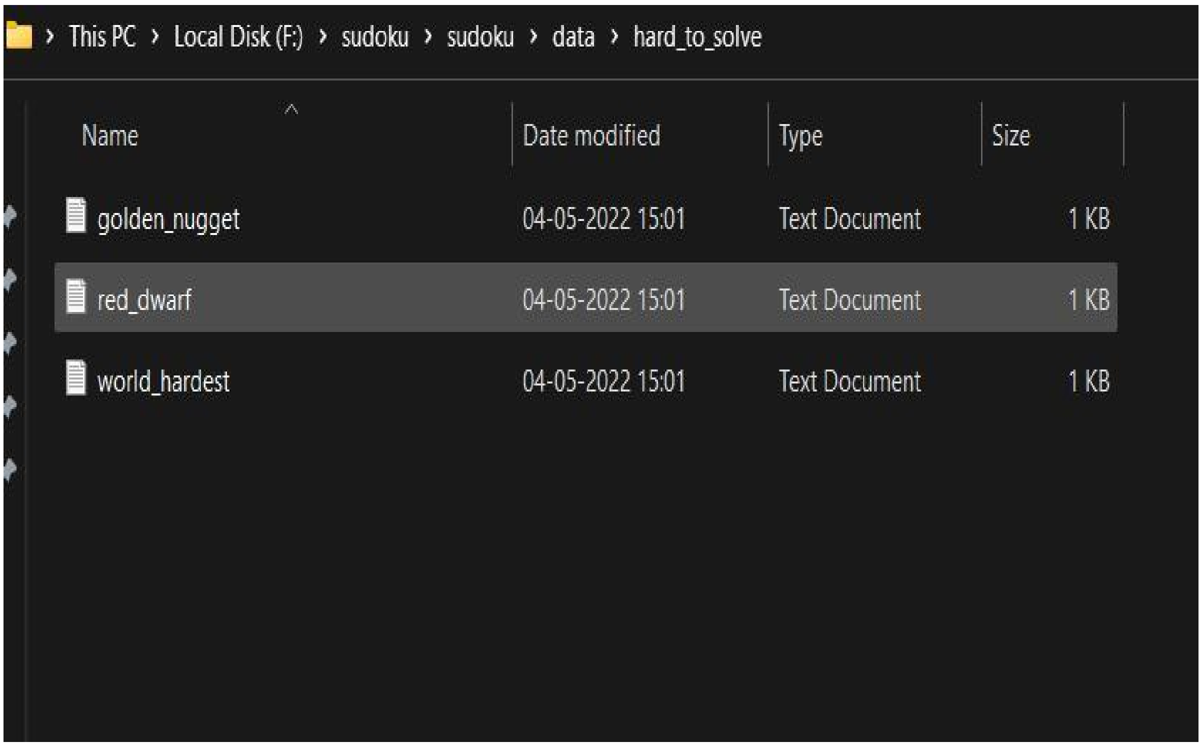


This PC › Local Disk (F:) › sudoku › sudoku › data › easy_to_solve

Name	Date modified	Type	Size
extreme_0	04-05-2022 15:01	Text Document	1 KB
extreme_1	04-05-2022 15:01	Text Document	1 KB
not_solvable	04-05-2022 15:01	Text Document	1 KB
not_valid	04-05-2022 15:01	Text Document	1 KB
really_hard_0	04-05-2022 15:01	Text Document	1 KB
really_hard_1	04-05-2022 15:01	Text Document	1 KB

Fig 4.13 easy_to_solve

888



This PC › Local Disk (F:) › sudoku › sudoku › data › hard_to_solve

Name	Date modified	Type	Size
golden_nugget	04-05-2022 15:01	Text Document	1 KB
red_dwarf	04-05-2022 15:01	Text Document	1 KB
world_hardest	04-05-2022 15:01	Text Document	1 KB

Fig 4.14 hard _to _solve

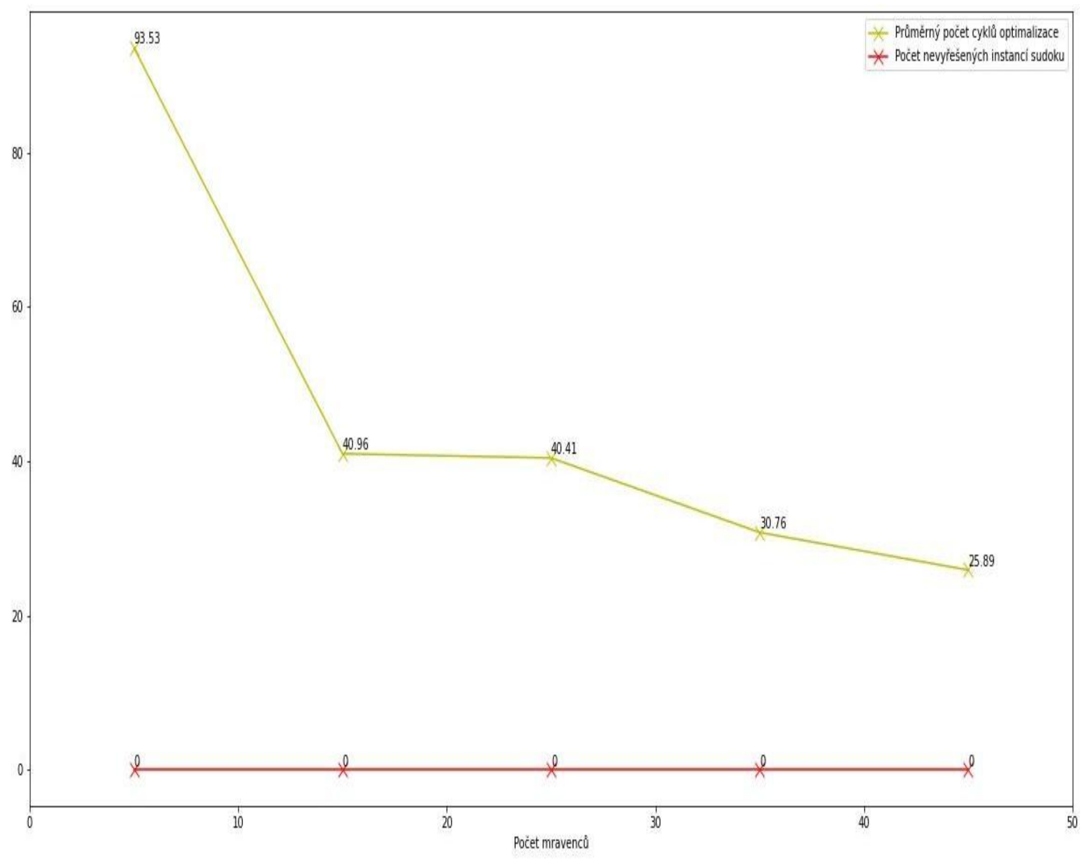


Fig 4.15 OPTMUM NUMBER OF ANTS

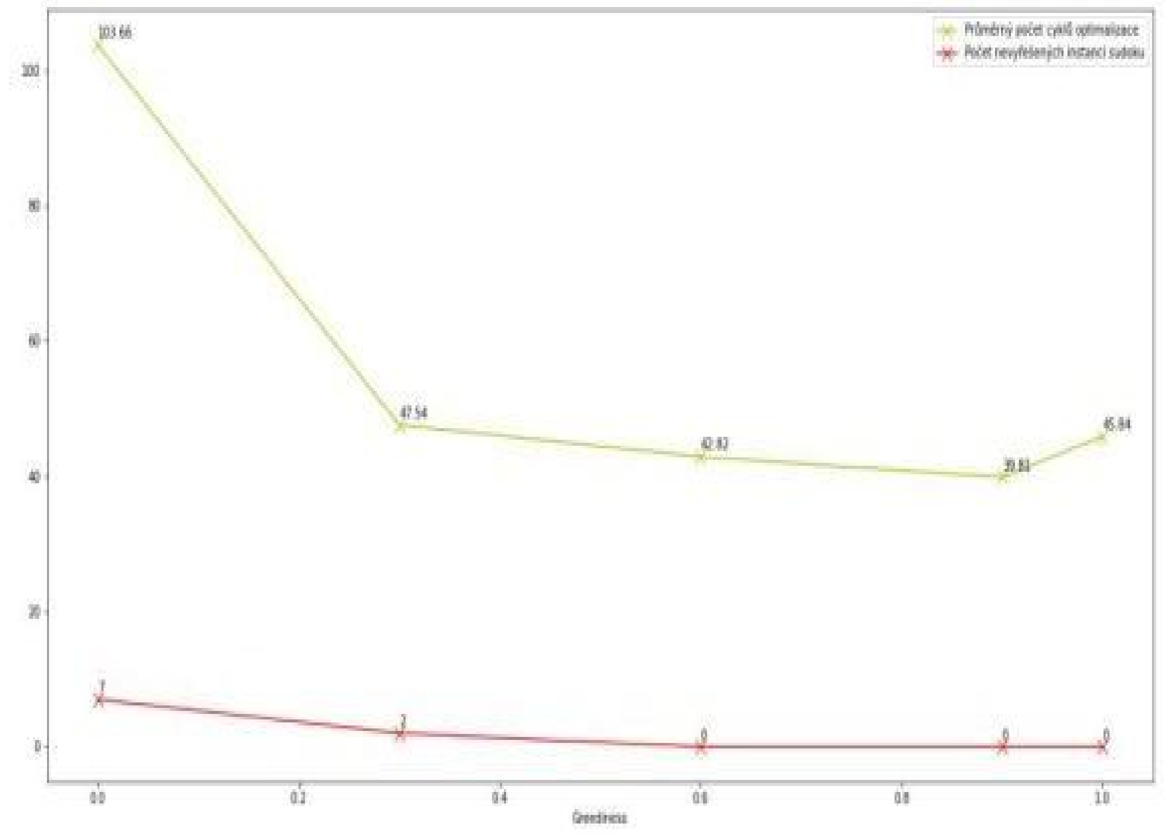


Fig 4.16 Optimal value of the GREEDINESS parameter

Chapter 5

Conclusion

Sudoku's most difficult, large-scale problems can be solved more quickly using our algorithm than using traditional backtracking methods, according to experiments that show the algorithm significantly outperforms existing algorithms.

5.1 **FUTURE ENHANCEMENT**

This work not only paves the way for future research on a general-purpose puzzle solver but also establishes Japanese pencil puzzles as a viable testing ground for a broad variety of algorithms..

Chapter 6

References

[1] J. Laire, “dlx-cpp,” available at https://github.com/jlaire/dlx-cpp, ac- cessed April 23

[2]. M. Dorigo and M. Birattari, “Work-in-progress: solving Sudoku puzzles using hybrid ant colony optimization algorithm,” in 1st International Conference on Industrial Networks and Intelligent Systems (INISCom). IEEE,, pp. 2020

[3] A. Inkala, AI Escargot - The Most Difficult Sudoku Puzzle. Lulu.com, Finland, 2019.

[4]T.Manere, “A hybrid approach for the Sudoku problem: using constraintprogramming in iterated local search,” IEEE Intelligent Systems, vol. 32, no. 2, pp. 52–62, 2017.

[5] M. Ercsey-Ravasz and Z. Toroczkai ,” available at https://github.com/jlaire/dlx-cpp, ac- cessed April 23, 2021