# Solving Sudoku by Sparse Signal Processing 

Master's Degree Project<br>Stockholm, Sweden February 2015

# KTH ROYAL INSTITUTE OF TECHNOLOGY 

## KTH Royal Institute of Technology

The Department of Electrical Engineering
Author: Muhammad Mohsin Abbasi
Email Address: mmabbasi@kth.se
Study Programme: Master in Wireless Systems, 120 Credits
Supervisor: Magnus Jansson
Date: January 19, 2015

## Abstract

Sudoku is a discrete constraints satisfaction problem which is modeled as an underdetermined linear system. This report focuses on applying some new signal processing approaches to solve sudoku and comparisons to some of the existing approaches are implemented. As our goal is not meant for sudoku only in the long term, we applied approximate solvers using optimization theory methods. A Semi Definite Relaxation (SDR) convex optimization approach was developed for solving sudoku. The idea of Iterative Adaptive Algorithm for Amplitude and Phase Estimation (IAA-APES) from array processing is also being used for sudoku to utilize the sparsity of the sudoku solution as is the case in sensing applications. LIKES and SPICE were also tested on sudoku and their results are compared with I1-norm minimization, weighted I1-norm, and sinkhorn balancing. SPICE and I1-norm are equivalent in terms of accuracy, while SPICE is slower than I1-norm. LIKES and weighted I1-norm are equivalent and better than SPICE and I1-norm in accuracy. SDR proved to be best when the sudoku solutions are unique; however the computational complexity is worst for SDR. The accuracy for IAA-APES is somewhere between SPICE and LIKES and its computation speed is faster than both.

## Sammanfattning

Sudoku är ett diskret bivillkorsproblem som kan modelleras som ett underbestämt ekvationssystem. Denna rapport fokuserar på att tillämpa ett antal nya signalbehandlingsmetoder för att lösa sudoku och att jämföra resultaten med några existerande metoder. Eftersom målet inte enbart är att lösa sudoku, implementerades approximativa lösare baserade på optimeringsteori. En positiv-definit konvex relaxeringsmetod (SDR) för att lösa sudoku utvecklades. Iterativ-adaptiv-metoden för amplitud- och fasskattning (IAA-APES) från gruppantennsignalbehandling användes också för sudoku för att utnyttja glesheten i sudokulösningen på liknande sätt som i mättillämpningen. LIKES och SPICE testades också för sudokuproblemet och resultaten jämfördes med I1-norm-minimiering, viktad I1norm, och sinkhorn-balancering. SPICE och I1-norm är ekvivalenta i termer av prestanda men SPICE är långsammare. LIKES och viktad I1-norm är ekvivalenta och har bättre noggrannhet än SPICE och I1norm. SDR visade sig ha bäst prestanda för sudoku med unika lösningar, men SDR är också den metod med beräkningsmässigt högst komplexitet. Prestandan för IAA-APES ligger någonstans mellan SPICE och LIKES men är snabbare än bägge dessa.

## Acknowledgement

First of all I am thankful to Allah Almighty for giving me abilities to complete this task. There is no power and strength except with Allah.

Secondly I am thankful to my project supervisor professor Dr.Magnus Jansson for encouraging me and motivating me with his ideas to accomplish my goal. His constant concern, innovative ideas, motivation and constant help has made me to reach to the final goal of my project.

Finally I would like to take this opportunity to thank my parents whose prayers are always with me. I am thankful to them for encouraging me morally to get the higher education and supporting me financially to complete it. My sweet wife who stood beside me and kept my spirit high during the entire course, and she is as happier as me on accomplishment of this task. My Siblings and my friends, in one or other way, are all part of this.

I thank you all and wish you all the best.

## Contents

1 Introduction ..... 1
1.1 Aim of the Project ..... 1
1.2 Methodology ..... 2
1.3 Report Organization ..... 2
1.4 Related Work ..... 2
2 Sudoku as a Linear System ..... 4
2.1 Problem Formulation ..... 4
3 Algorithms' Description ..... 7
3.1 L1-Norm minimization ..... 7
3.2 Reweighted I1-norm minimization .....  8
3.2.1 Algorithm ..... 8
3.3 SPICE and LIKES. ..... 9
3.3.1 SPICE ..... 9
3.3.1.1 Problem formulation ..... 9
3.3.1.2 Algorithm ..... 10
3.3.2 LIKES ..... 11
3.3.2.1 Algorithm ..... 11
3.4 Semi Definite Relaxation ..... 12
3.5 Iterative Adaptive Approach based on Weighted Least Squares ..... 14
3.5.1 Data Model ..... 14
3.5.2 IAA-APES ..... 15
3.5.2.1 Algorithm ..... 16
3.6 Sinkhorn Balancing ..... 16
4 Puzzles ..... 17
5 Results and Discussions ..... 18
5.1 Puzzle Set 1 ..... 18
5.2 Puzzle Set 2 ..... 21
5.3 Puzzle Set 3 ..... 22
6 Conclusion and Future Work ..... 25
7 References ..... 26
Appendix ..... 28

## List of Figures

Figure 1: Array processing example ..... 14
Figure 2: Execution time for first 20 puzzles (in seconds) ..... 20
Figure 3: Increase in computation time with puzzle size ..... 20
Figure 4: Singular value plot for incorrectly solved puzzle ..... 24
Figure 5: Singular value plot for correctly solved puzzle ..... 24

## List of Tables

Table 1: Example $9 \times 9$ sudoku ..... 4
Table 2: Comparison of methods for number of incorrectly solved entries of the puzzle set 1 ..... 18
Table 3: Comparison table for number of solved puzzles in puzzle set 1 ..... 19
Table 4: Comparison for puzzle set 2 ..... 21
Table 5: Puzzle set 2, number of solved puzzles ..... 22
Table 6: Comparison table for puzzle set 3 ..... 22
Table 7: Total puzzles solved for puzzle set 3 ..... 23
Table 8: Puzzle set 1 ..... 28
Table 9: Puzzle set 2 ..... 31
Table 10: Puzzle set 3 ..... 34

## Terminologies

LDPC: Low Density Parity Check codes
BP: Belief Propagation

SPICE: Sparse Iterative Covariance-based Estimation

LIKES: Likelihood-based Estimation of Sparse Parameter

SDR: Semi Definite Relaxation

IAA: Iterative Adaptive Approach
IAA-APES: Iterative Adaptive Approach for Amplitude and Phase Estimation
SVD: Singular Value Decomposition

## 1 Introduction

Sudoku is an $N \times N$ logical based puzzle in which $n=N^{2}$ entries are arranged in such a way that the arrangement satisfies given clues and constraints. For example considering a $9 \times 9$ puzzle as shown in Table 1, it is required to fill each empty box in such a way that each row each column and each $3 \times 3$ box should contain each digit 1 to 9 once. The initial given boxes are called clues. Based on total number of given clues and their locations, the puzzles can be categorized from easy to hard.

The sudoku puzzle is a discrete constraints satisfaction problem and the reason why it got interest in signal processing community is that it has ties to decoding error correcting codes which is also a discrete constraints satisfaction problem. In fact, the methods used for error correcting codes like Low Density Parity Check (LDPC) Belief Propagation (BP) decoding can directly be apply on sudoku as a sudoku puzzle has a tanner graph representation. Therefore a good decoding algorithm can be applied to sudoku and alternatively a good sudoku solver can be applied to the problem of decoding error correcting codes which results in an interesting interplay between sudoku and error correcting codes.

There are two main solution approaches for sudoku puzzles. First approach is to get the exact solution of the sudoku puzzles. Sudoku solving algorithms which use this approach are efficient as they tend to get exact unique or all solutions of the puzzle but their complexity increases as the puzzle size increases. The other group of sudoku solvers does not guarantee the exact solution satisfying all constraints, rather they tend to get optimum approximate solution of the puzzle by relaxing some of the hard constraints and in most cases they get to the exact optimum solution. Relaxing constraints may result in a lower computational complexity and hence fast processing speed.

### 1.1 Aim of the Project

It has been noticed from the literature that sudoku can be modeled as an underdetermined linear system [1], and that the traditional calculus and mathematical approaches can be used to solve sudoku. Our aim for this project is to use optimization theory methods to see their effects on sudoku puzzle problem. In particular sparsity of the sudoku system is aimed to be utilized as a tool for this project along side of the convex optimization strategies. Our aim is to analyze and implement the following techniques to sudoku problem and to compare the accuracy and computational complexity to solve the puzzle.

- SPICE [2]
- LIKES [2]
- SDR [3]
- IAA-APES [4]

Some of the existing approaches for sudoku are also implemented and are compared with the above techniques. These are as following:

- I1-norm minimization [5]
- Weighted I1-norm minimization [5] [6]
- Sinkhorn balancing [7]


### 1.2 Methodology

This thesis report involves in-depth literature study in optimization theory and to use it to sudoku puzzles. The literature taken is from highly recognized books and from research papers. The knowledge gained is implemented in matlab. After deep understanding of the algorithms, they are modeled to suit the sudoku puzzles taken from different resources, the puzzles are generated, algorithms are implemented in matlab and the results are drawn as mentioned in section 5. The generated puzzles are also included in the Appendix of the report in order to make it possible to reproduce the results or to compare the results for further research.

### 1.3 Report Organization

Section 1 introduces the problem definition and related work. Section 2 discusses the problem formulation of the sudoku and how it is being used as an optimization problem. Section 3 describes the detailed description of the algorithms used in this project. Section 4 describes the puzzles used for the results. Section 5 contains the results of the algorithms used. Section 6 concludes the report. Section 7 shows the related references and Appendix contains the puzzles used for this project

### 1.4 Related Work

Sudoku can be solved by logical elimination techniques as described in [8]. Back tracking, exact cover problem and brute force algorithms as described in [9] [10] are techniques which tend to get to the exact solution of the problem. Guessing, back tracking and brute force algorithms visit empty cells in some order, they assigned a value which is legal for that cell, and then recursively check if this value leads to a sudoku solution by filling the next empty cell values. If for some cell value there is no legal choice the algorithm goes back to the previous cell (hence backtracking) discards its current value and changes it to other legal value if exists, if again no value is allowed here, the algorithm goes back one more cell and does the procedure again until all cells are filled with legal values. These Algorithms guarantee the solution if the sudoku puzzle is a valid puzzle. The most efficient method in this category is Dancing Links [11], an exact cover problem which proved to be the most efficient algorithm for solving sudoku. Dancing links is a recursive, back tracking depth first search algorithm which implements the Knuth's $X$ algorithm [12] in an efficient and fast way. In sudoku case the sudoku problem is divided into a matrix of 0's and 1's and the goal is to find the subset of rows which have exactly one element ' 1 ' in each column. Sudoku is also
solved as constraint satisfaction problem [13]. In [1], sudoku constraints are mathematically described as integer linear system of equations which are then solved by integer linear programing. These algorithms guarantee the one or all solutions of the sudoku puzzle however the complexity increases exponentially as the puzzle size tends to increase.

Other category of sudoku solution is suboptimal solutions as they do not guarantee the exact solution all the time, but in these methods some of the hard constraints of the puzzles are relaxed so as to improve the computational complexity. These methods are where traditional calculus and signal processing approaches are used to solve sudoku. There are number of approximate methods for solving sudoku based on randomly assigning numbers for each cell, calculating errors and then shuffling the numbers around the whole sudoku grid to reduce the error. The approaches for shuffling the numbers include genetic algorithm [14], simulated annealing [15] and tabu search [16]. Sinkhorn balancing [7] is an approximate probabilistic solver that tends to solve all but most difficult puzzles by projecting matrix onto doubly stochastic matrix. A description of message passing algorithm (Belief Propagation) for solving sudoku is given in [17]. BP only solves easy puzzles due to the loopy nature of the sudoku in its tanner graph representation. In [1] it is shown that the sudoku problem can be written as an underdetermined system of linear equations which has a sparse solution. Therefore techniques to exploit sparsity and linear equality are used in I1-norm minimization [5] which is first traditional calculus based optimization technique for sudoku solution. Sudoku was formulated as an optimization problem over a set of probabilities in [18] and they proposed an entropy minimization approach to solve not all but most of the puzzles having varying difficulties.

## 2 Sudoku as a Linear System

### 2.1 Problem Formulation

Let $S$ be an $N \times N$ sudoku puzzle. The contents of cell $n$ of $S$ can be represented as $S_{n}=$ $\{1,2, \ldots, N\}$ for $n=\left\{1,2, \ldots, N^{2}\right\}$.

Let $i_{n}=\left[I\left(S_{n}=1\right), I\left(S_{n}=2\right), \ldots, I\left(S_{n}=N\right)\right]^{T}$ is an indicator vector where $I\left(S_{n}=k\right)$ is an indicator function which is given as

$$
I\left(S_{n}=k\right)=\left\{\begin{array}{l}
1, \text { if } S_{n}=k \\
0, \\
\text { Otherwise }
\end{array}\right.
$$

Let $x=\left[i_{1}, i_{2}, \ldots, i_{N^{2}}\right]^{T}$ of size $N^{3}$. Taking the $9 \times 9$ sudoku puzzle for example, there are four types of constraints that need to be satisfied in order to solve sudoku puzzle.

Row constraints: Each row of $S$ should contain all digits from 1 to 9 .
Column constraints: Each column of $S$ should contain all digits 1 to 9 .
Box constraints: Each $3 \times 3$ box of $S$ should contain all digits 1 to 9 .
Cell constraints: Each cell of $S$ should be filled.
In addition to these, there are some clues given for each puzzle which must also be satisfied. Each of the above constraints and the clues can be written as a linear combination to the elements of $x$.

For example, the row constraints of the puzzle in Table 1 can be expressed as,

| 9 | 4 |  | 3 | 5 | 1 |  |  | 2 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 7 | 8 | 5 | 4 | 2 | 6 | 3 | 1 |  |
| 2 | 3 | 1 | 7 | 8 | 9 | 5 | 6 | 4 |
|  | 7 | 3 |  | 9 | 2 |  | 8 | 1 |
| 6 | 1 | 9 |  |  | 7 |  |  | 5 |
|  | 2 |  | 5 | 1 |  | 7 | 9 |  |
| 3 | 5 | 4 | 9 | 6 | 8 |  | 2 | 7 |
|  | 9 | 7 | 2 | 3 |  | 6 | 5 | 8 |
| 8 | 6 | 2 | 1 | 7 | 5 | 9 | 4 | 3 |

Table 1: Example $9 \times 9$ Sudoku

$$
\begin{gathered}
{\left[\begin{array}{lllll}
I_{9 \times 9} & I_{9 \times 9} & I_{9 \times 9} & I_{9 \times 9} & I_{9 \times 9} \\
I_{9 \times 9} & I_{9 \times 9} & I_{9 \times 9} & I_{9 \times 9} & 0_{9 \times 648}
\end{array}\right] x=\left[\begin{array}{c}
1 \\
1 \\
\vdots \\
1
\end{array}\right]} \\
{\left[0_{9 \times 81} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} 0_{9 \times 567}\right] x=\left[\begin{array}{c}
1 \\
1 \\
\vdots \\
1
\end{array}\right]}
\end{gathered}
$$

$$
\left[0_{9 \times 648} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9} I_{9 \times 9}\right] x=\left[\begin{array}{c}
1 \\
1 \\
\vdots \\
1
\end{array}\right]
$$

Similarly for column constraints, we can write,

$$
\begin{gathered}
{\left[I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72}\right] x=\left[\begin{array}{c}
1 \\
1 \\
\vdots \\
1
\end{array}\right]} \\
{\left[0_{9 \times 9} I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72} \ldots I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 63}\right] x=\left[\begin{array}{c}
1 \\
1 \\
\vdots \\
1
\end{array}\right]} \\
\vdots \\
{\left[0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} 0_{9 \times 72} I_{9 \times 9} \ldots 0_{9 \times 72} I_{9 \times 9}\right] x=\left[\begin{array}{c}
1 \\
1 \\
\vdots \\
1
\end{array}\right]}
\end{gathered}
$$

Similarly the box constraints can be written. For example for box 1 we can write,

$$
\left[J_{9 \times 27} 0_{9 \times 54} J_{9 \times 27} 0_{9 \times 54} J_{9 \times 27} 0_{9 \times 540}\right] x=\left[\begin{array}{l}
1 \\
1 \\
\vdots \\
1
\end{array}\right]
$$

where,

$$
J_{9 \times 27}=\left[I_{9 \times 9} I_{9 \times 9} I_{9 \times 9}\right]
$$

Similarly for box 2 ,

$$
\left[0_{9 \times 27} J_{9 \times 27} 0_{9 \times 54} J_{9 \times 27} 0_{9 \times 54} J_{9 \times 27} 0_{9 \times 513}\right] x=\mathbf{1} \text { and so on. }
$$

Now for cell constraints, for example, the constraint that $1^{\text {st }}$ and $2^{\text {nd }}$ cells should be filled can be written respectively as,
[11111111110000 $\ldots 000] x=1$ and $[00000000011111111100 \ldots 00] x=1$
Finally the clues can also be written as linear combination of x . For example the clue that cell 2 contains value 4 can be written as $i_{2}=\left[\begin{array}{lllll}0 & 0 & 1000 & 0 & 0\end{array}\right]^{T}$ i.e.

$$
\text { [ } 00000000000001000 \ldots \text {... } 0 \text { ] } x=1 \text {. }
$$

By combining all of above constraints, we can write the Sudoku problem in more generic form as

$$
A x=\left[\begin{array}{c}
A_{\text {row }}  \tag{1}\\
A_{\text {col }} \\
A_{\text {box }} \\
A_{\text {cell }} \\
A_{\text {clue }}
\end{array}\right] x=\left[\begin{array}{c}
1 \\
1 \\
\vdots \\
1
\end{array}\right]=b
$$

Where $A_{\text {row }}, A_{\text {col }}, A_{\text {box }}, A_{\text {cell }}, A_{\text {clue }}$ are the matrices associated with different constraints. Size of matrix $A$ is $\left(4 N^{2}+C\right) \times N^{3}$ where $C$ denotes number of clues.

In the rest of this report, we will investigate the type of problem in equation (1) and will use different methods to solve it.

## 3 Algorithms' Description

### 3.1 L1-Norm minimization

From equation (1), we have an under determined linear system $A x=b$ which leads to infinite number of solutions. However not all the solutions are sudoku solutions. In fact, for the sudoku having a unique solution there is one and only solution $x_{s}$ which consists of only 0's and 1's. Therefore we have to look for one particular solution among many solutions. This problem leads us to optimization theory literature which strives to find the optimal solution through some cost function.

Recently emerged field of compressive sensing uses the notion of sparse signal processing. A signal is sparse if it has few none zero elements. The idea in compressive sensing is to reconstruct a signal from far fewer measurements than the traditional methods used. As we can show that sudoku solution is the sparsest solution of $A x=b$, we can use the ideas presented in the literature of compressive sensing to get to the solution, where we can find that one of the ideal cost function for finding the sparsest solution of (1) is I1-norm.

As described in [5], it can be shown easily that the solution $x_{s}$, which is unique solution of sudoku, is the sparsest solution of equation (1).

As we can see that $x_{s}$ is $N^{2}$ sparse, that is there are $N^{2}$ none-zero elements out of $N^{3}$ elements. Therefore $\left\|x_{s}\right\|_{0}=N^{2}$, where $\left\|x_{s}\right\|_{0}$ represents the number of none zero elements in $x_{s}$. Suppose we have some solution $x_{p}$ such that $x_{s} \neq x_{p}$ and

$$
x_{p}=\left[\begin{array}{c}
i_{p 1} \\
i_{p 2} \\
i_{p 3} \\
\vdots \\
i_{p N^{2}}
\end{array}\right]
$$

Assume $\left\|x_{p}\right\|_{0}<N^{2}$
It implies that at least one of the vectors $i_{p 1}$ to $i_{p N^{2}}$ must be all 0 's but this will violate the cell constraints of our sudoku which states that all cells must be filled. Hence $x_{p}$ will not be a feasible solution. So all feasible solution of (1) must satisfy the equation

$$
\|x\|_{0}=\sum_{j=1}^{N^{2}}\left\|i_{j}\right\|_{0} \geq N^{2}
$$

This implies that the least number of none zero elements in a feasible solution is $N^{2}$ and it will exists only when $x=x_{s}$. Therefore $x_{s}$ is the sparsest solution of equation (1).

To find the sparsest solution of (1) we will consider the following optimization problem.

$$
\begin{equation*}
\text { minimize }\|x\|_{0} \text { sub. to } A x=b \tag{2}
\end{equation*}
$$

As equation (2) is combinatorial problem and solving it in a mathematical tractable way is NP-hard problem. Literature in compressive sensing is trying to prove its equivalence to alternative convex minimization problem which is I1-norm minimization problem. The equivalence exists under certain conditions based on the matrix $A$ used.

$$
\begin{equation*}
\text { minimize }\|x\|_{1} \text { sub. to } A x=b \tag{3}
\end{equation*}
$$

Where $\|x\|_{1}=\sum_{j=1}^{N^{3}}\left|x_{j}\right|$ is the 1 norm of $x$.
Most of the time solution to (3) leads to solution of (2) but it does not guarantee the correct solution all the time, because presently known conditions for equivalence of (2) and (3) does not hold for sudoku. However solution to (3) solves most of the sudoku puzzles and moreover as (3) is mathematically tractable problem we used (3) to get to the solution. Equation (3) can easily be expressed as a linear programing model and we have many standard software packages to solve it. We used the linear programing package as in [19] for this thesis work.

### 3.2 Reweighted l1-norm minimization

Due to limited accuracy of I1-norm minimization, research has been carried out to look for some methods comparable to I1-norm minimization or which can outperform I1-norm method. Numerical results in this thesis and also from some other related work like in [5], it has been shown that in most situations reweighted I1-minimization outperforms I1-norm method. Reweighted 11 is basically a weighted some of series of I1-norm minimization. The idea is to assign some weights to iteration 1, then do the I1-norm minimization using those weights, and then use the solution to form new weights for the next iteration. Weights are used to reduce the small components as small as possible by minimizing the weighted I1-norm. Complete algorithm as mentioned in [5] is as below.

### 3.2.1 Algorithm <br> Initialize weights $W^{0}=I$ for $0^{\text {th }}$ iterations. Where $I$ is identity matrix of size $N^{3} \times N^{3}$

Solve the weighted I1-minimization problem for the solution $x^{i}$ as below, where super-index $i$ represents $\mathrm{i}^{\text {th }}$ iteration number.
where,

$$
\min _{x}\left\|W^{i-1} x\right\|_{1} \text { sub. to } A x=b
$$

$$
W_{k, k}^{i-1}=\frac{1}{\left|x_{k}^{i-1}\right|+\epsilon}, \quad 0<\epsilon<1
$$

represents the diagonal element $(k, k)$ of $W$ in iteration $i-1$.
$\epsilon$ can be tuned to different values to obtain the sparsest solution. We used $\epsilon=0.5$
Iterations are terminated when a certain threshold is achieved. We used a threshold of $10^{-4}$. That is when $\left\|x^{i}-x^{i-1}\right\|^{2}<10^{-4}$ the iterations are terminated.

### 3.3 SPICE and LIKES

As we have seen that for our sudoku case the solution is the sparsest solution of equation (1). Therefore we have opened a door to exploit different methods to get to the solution of sudoku using the literature in sparse parameter estimation. One of the methods we have presented is l1-norm minimization and its iterative version. Here we are introducing another two methods of sparse parameter estimation and their results on sudoku. One is Sparse Iterative Covariance-based Estimation (SPICE) and the other one is Likelihood-based Estimation of Sparse parameters (LIKES). For more details of these methods we refer to [2]. Here we will modify the problem presented in [2] to suit sudoku problem presented in equation (1).

### 3.3.1 SPICE

In sparse parameter estimation most of the methods are parameter based methods which depends on selection of one or more user parameters which is usually a difficult task. SPICE is a newly proposed method in [2] which does not have this drawback. SPICE is based on statistically sound covariance based selection criteria. It is shown in [2] that SPICE has more accuracy than I1-norm. Here in this report we are implementing SPICE and will present the results based on sudoku problem.

### 3.3.1.1 Problem formulation

Here we present the problem formulation of SPICE and then we will modify it to sudoku case.
SPICE considers the following linear model as described in [2].

$$
y=\sum_{k=1}^{M} a_{k} x_{k}+e=\left[a_{1}, \ldots, a_{M} I\right]\left[\begin{array}{l}
x \\
e
\end{array}\right]=B \beta
$$

Where $x=\left[x_{1}, \ldots, x_{M}\right]^{T}$ is unknown sparse parameter vector.
$\beta=\left[x^{T} e^{T}\right]^{T}, B=\left[a_{1}, \ldots, a_{M} I\right]$ and $e$ is a noise term.
SPICE makes a working assumption that elements of $\beta$ are random uncorrelated variables having zero means and variances denoted by $p_{k}$ for $x_{k}$ and $\sigma_{j}$ for $e_{j}$ with k from 1 to M and j from 1 to N .

We can write the covariance matrix for $y$ as

$$
\boldsymbol{R}=E\left(y y^{T}\right)=E\left(B \beta \beta^{T} B^{T}\right)=B E\left(\beta \beta^{T}\right) B^{T}=B P B^{T}
$$

Where,

$$
P=\left[\begin{array}{cccccccc}
p_{1} & 0 & \ldots & & & & 0 \\
0 & \ddots & & & & & \vdots \\
\vdots & & \ddots & & & & \\
\vdots & & & p_{M} & & & \vdots \\
& & & & \sigma_{1} & & \\
0 & \ldots & \ldots & \ldots & \ldots & 0 & \sigma_{N}
\end{array}\right]
$$

For our sudoku problem, as we do not have noise $e$ we can modify the above mentioned problem as following.

$$
\beta=x, B=\left[a_{1}, \ldots, a_{M}\right]=A
$$

If we consider $y=b$ as in equation (1), Then SPICE problem can be written exactly as our sudoku problem in equation (1). That is

$$
y=B \beta \Rightarrow b=A x
$$

Therefore $R=A P A^{T}$ where,

$$
P=\left[\begin{array}{ccc}
p_{1} & 0 & \ldots \\
0 & \ddots & 0 \\
\vdots & 0 & p_{M}
\end{array}\right]
$$

Then as mentioned in [2], SPICE estimation merit function can be written as following weighted covariance fitting criteria

$$
\left\|R^{-\frac{1}{2}}\left(R-y y^{T}\right)\right\|^{2}
$$

Where ||. || represents Frobenius norm for the matrices.
It has been verified in [2] that,

$$
x=P A^{T} R^{-1} b
$$

From the definition of Frobenius norm, we know that $\|A\|_{F}=\left[\operatorname{Tr}\left(A^{T} A\right)\right]^{1 / 2}, \operatorname{Tr}=\operatorname{Trace}$.

We can write,

$$
\left\|R^{-\frac{1}{2}}\left(R-b b^{T}\right)\right\|^{2}=\operatorname{Tr}\left[\left(I-b b^{T} R^{-1}\right)\left(R-b b^{T}\right)\right]=\operatorname{Tr}(R)+\|b\|^{2} b^{T} R^{-1} b-2\|b\|^{2}
$$

Where,

$$
\operatorname{Tr}(R)=\sum_{k=1}^{M} p_{k}\left\|a_{k}\right\|^{2}
$$

So the SPICE minimization problem will become,

$$
\min \left(\left\|R^{-\frac{1}{2}}\left(R-b b^{T}\right)\right\|^{2}\right)=\min _{p} b^{T} R^{-1} b+\sum_{k=1}^{M} w_{k}^{2} p_{k}
$$

Where the weights,

$$
w_{k}=\frac{\left\|a_{k}\right\|}{\|b\|}
$$

In [2], methods for solving above convex minimization problems are discussed in general form. Here we are using CA-based solver discussed in [2] for sudoku problem. The algorithm we used is as below

### 3.3.1.2 Algorithm

$$
\begin{gathered}
p_{k}^{i}=\left|\beta_{k}^{i}\right| / w_{k} \\
\beta_{k}^{i+1}=p_{k}^{i} a_{k}^{T} R^{-1}(i) b ; \quad R(i)=B P(i) B^{T}
\end{gathered}
$$

where $k=1,2, \ldots, M$ and $i=1,2, \ldots$
This iterative algorithm converges to the solution of the SPICE minimization problem. The initialization is done as following.

$$
\begin{equation*}
\beta_{k}^{0}=\frac{a_{k}^{T} b}{\left\|a_{k}\right\|^{2}}, \quad k=1,2, \ldots, M \tag{4}
\end{equation*}
$$

### 3.3.2 LIKES

LIKES tends to minimize the following negative log likelihood function.

$$
f(p)=\ln |R|+b^{T} R^{-1} b
$$

The second term $b^{T} R^{-1} b$ is a convex function of $p$ as seen in SPICE, however $\ln |R|$ is not a convex function rather it is a concave function and therefore it is hard to find its global minimum. An iterative algorithm to solve above minimization problem is discussed in [2] which decreases $f(p)$ at each iteration and hence expected to converge at least locally.

Let $\tilde{p}$ be an arbitrary point in the parameter space, and $\tilde{R}$ is its covariance matrix, then by using the majorization principals it has been shown in [2] that $\ln |R|$ can be written as:

$$
\ln |R| \leq \ln |\tilde{R}|-N+\operatorname{tr}\left(\tilde{R}^{-1} R\right)=\ln |\tilde{R}|-N+\sum_{k=1}^{M} \widetilde{w}_{k}^{2} p_{k}
$$

Where,

$$
\begin{equation*}
\widetilde{w}_{k}^{2}=a_{k}^{T} \tilde{R}^{-1} a_{k} \tag{5}
\end{equation*}
$$

Hence we can write,

$$
f(p) \leq b^{T} R^{-1} b+\ln |\tilde{R}|-N+\sum_{k=1}^{M} \widetilde{w}_{k}^{2} p_{k}=g(p)
$$

Note that $f(\tilde{p})=g(\tilde{p})$
So it can be derived from above equation that we can decrease $f(p)$ from $f(\tilde{p})$ to $f(\hat{p})$ such that

$$
f(\hat{p})<g(\hat{p})<g(\tilde{p})=f(\tilde{p})
$$

So LIKES minimizes $g(p)$ in order to minimize $f(p)$.
As we can see that $g(p)$ is under a constant a SPICE like convex minimization problem and hence by choosing $\tilde{p}$ and updating it iteratively we expect to converge to minimum of $f(p)$ using a convex minimization problem. The algorithm consists of the following step

### 3.3.2.1 Algorithm

Inner step: using the recent estimated $\tilde{p}$ and its covariance $\tilde{R}$ solve the SPICE minimization problem and get its solution $\hat{p}$.

Outer step: Set $\tilde{p}=\hat{p}$ and repeat inner step.
We used same initialization as in SPICE depicted in equation (4) and the weights are then updated adaptively as in equation (5).

Due to the maximum likelihood characteristics of LIKES and also the fact that it uses adaptive weights it has been shown in [2] that the LIKES estimates are more accurate than the SPICE. We have used this algorithm for sudoku and our results also show the improvements in LIKES accuracy.

### 3.4 Semi Definite Relaxation

In recent years Semi Definite Relaxation (SDR) got interest in signal processing as a very powerful computationally efficient optimization technique for very difficult optimization problems in particular to none convex quadratically constraint quadratic programs [3]. In this section we are going to reformulate our sudoku problem in such a way that it becomes SDR problem. So let us write our problem of interest in a different way.

From equation (1), we have $A x=b$ where $x \in\{0,1\}$.
Let us reformulate the problem such that, $\tilde{A} \tilde{x}=\tilde{b}$ where $\tilde{x} \in\{-1,1\}$. We may write $A x=b$ as,

$$
\begin{equation*}
A(2 x-1)=2 b-A .1 \tag{6}
\end{equation*}
$$

If we let $\tilde{x}=2 x-1$, then $\tilde{x} \in\{-1,1\}$ if $x \in\{0,1\}$.
So our reformulated problem will be,

$$
\begin{equation*}
\tilde{A} \tilde{x}=\tilde{b} \tag{7}
\end{equation*}
$$

Where $\tilde{A}=A, \tilde{x}=2 x-1$ and $\tilde{b}=2 b-A .1$
Now our goal is to solve equation (7) by SDR approach.
Our goal is to minimize $\|\tilde{b}-\tilde{A} \tilde{x}\|_{2}^{2}$, which can be written in matrix form as

$$
\begin{equation*}
\|\tilde{b}-\tilde{A} \tilde{x}\|_{2}^{2}=\|M z\|_{2}^{2} \tag{8}
\end{equation*}
$$

Where $M=\left[\begin{array}{cc}\tilde{b} & -\tilde{A}\end{array}\right]$ and $z=\left[\begin{array}{l}1 \\ \tilde{x}\end{array}\right]$ and $\|.\|_{2}$ now represents Frobenius norm.
By the definition of Frobenius norm, we know that $\|A\|_{F}=\left[\operatorname{Tr}\left(A^{T} A\right)\right]^{1 / 2}$.
Therefore,

$$
\left.\|M z\|_{2}^{2}=\operatorname{Tr}\left((M z)^{T} M z\right)\right)=\operatorname{Tr}\left(z^{T} M^{T} M z\right)
$$

By using the circular shift property of trace it will become

$$
\|M z\|_{2}^{2}=\operatorname{Tr}\left(M^{T} M z z^{T}\right)
$$

Now we can see that the objective function is linear in the matrix $z Z^{T}$. if we assume $Z=z Z^{T}$, that will make Z a rank 1 positive semi definite matrix. So our minimization problem will be translated to following:
> minimize $\left(\operatorname{Tr}\left(M^{T} M Z\right)\right)$ subject to the following 3 conditions

1. Rank of $Z$ is 1
2. $Z>0$, that is $Z$ is positive semi definite
3. As $Z=z Z^{T}$, where $z=\left[\begin{array}{l}1 \\ \widetilde{x}\end{array}\right]$ and $\tilde{x} \in\{-1,1\}$ therefore diagonal of $Z$ is all 1's.

That is,

$$
\begin{equation*}
\operatorname{minimize}\left(\operatorname{Tr}\left(M^{T} M Z\right)\right) \operatorname{subj} \text { to } \operatorname{Rank}(z)=1, Z>0, \operatorname{and} \operatorname{diag}(Z)=1 \tag{9}
\end{equation*}
$$

Until now we have achieved nothing as equation (9) is still hard to solve. However the only problem in (9) is the constraint $\operatorname{Rank}(z)=1$ which is a non-convex function. If we drop this constraint then we will get the following relaxed version of the problem

$$
\begin{equation*}
\operatorname{minimize}\left(\operatorname{Tr}\left(M^{T} M Z\right)\right) \text { subj to } Z>0 \text {, and } \operatorname{diag}(Z)=1 \tag{10}
\end{equation*}
$$

Problem (10) is known as an SDR problem of problem (8) and it can be effectively solved in numerically reliable and efficient fashion. SDR can be handled easily by the readily available software package. We used CVX package [19].

To retrieve $z$ from $Z$ we used Singular Value Decomposition (SVD) of $Z$ and took the singular vector corresponding to the largest singular value for reconstruction as it is its best rank 1 approximation.

SVD is theorem based on the linear algebra, which says that any rectangular matrix A can be written as a product of three matrices as below.

$$
A_{m n}=U_{m m} S_{m n} V_{n n}^{T}
$$

Where columns of $U$ are orthonormal Eigen vectors of $A A^{T}$, coloums of V are orthonormal Eigen vectors of $A^{T} A$, and S is the diagonal matrix containing square roots of the eigenvalues of $A A^{T}$ or $A^{T} A$ in its diagonal in descending order.

For matrix Z in our case, the SVD of Z leads to the U matrix containing Eigen vectors of $Z Z^{T}$, which is equivalent to $\left(z z^{T}\right)^{2}$, the reconstruction for ' $z$ ' is straight forward according to the Eigen values definition. According to Eigen values and Eigen vector properties, if $\lambda$ is the Eigen value of $Z Z^{T}=$ $\left(z Z^{T}\right)^{2}$ then Eigen value of $\left(z z^{T}\right)$ will be square root of $\lambda$ with the same corresponding Eigen vector. As multiplying Eigen value with its Eigen vectors will reconstruct matrix Z. Then according to Eigen values properties, multiplying square root of Eigen values with Eigen vectors will reconstruct z. However in our case since the matrix was assumed to be rank 1, the best approximation for the rank 1 matrix is to use Eigen vector corresponding to the largest Eigen value. Since by definition, for rank 1 matrix all the information should lie in its largest Eigen value while other Eigen values should be zero. Hence to reconstruct $z$ we multiplied square root of largest Eigen value $S(1,1)$, with the corresponding Eigen vector in U .

After having ' $z$ ', since we know that the first component of ' $z$ ' should be 1 in our case, as it is in the problem formulation above. We normalized vector ' $z$ ' by dividing its components with first element.

Finally we took the sign of the values of vector ' $z$ ', ignoring the magnitude. As our original vector was consisting of +1 and -1 only. So negative sign for a value in ' $z$ ' corresponds to -1 and positive sign approximates +1 .

### 3.5 Iterative Adaptive Approach based on Weighted Least Squares

As we are using sparse signal processing techniques to solve the sudoku problem, here we are using another technique from sparse signal literature. The one of the challenge of sparse signal processing techniques is they are much computationally heavy and slow, and the techniques we have discussed earlier also have varying computational complexity for our sudoku case. Here we tried another method which we assumed should work faster. The method we are going to use is from the literature of array processing and is described in [4]. Here we will explain a bit about the mathematical model of the algorithm and then we will use it for our sudoku case.

Array processing is used for sensing location and waveforms of the sources by combining the signal from different sources using a set of sensors separated spatially (e.g. an array of antennas). The goal is to enhance the signal by combining received signals in such a way that the noises and other unwanted interferences are suppressed. If the environment in which the sources are sensed is stationary only for a limited time, then the number of snapshots available can be very few which makes it challenging to localize the sources and also the near field sources which are near to each other, spatially and with respect to Doppler shifts, are difficult to discriminate. In [4], it is described that the array processing scenario can be modeled in sparse signal representation as the number of actual sources is much smaller than the number of potential source points that are considered. Moreover sparse based techniques can work for as low snapshot as one, so it is considered to solve the array processing localization using sparse signal based technique which they named as Iterative Adaptive Approach for Amplitude and Phase EStimation (IAA-APES).

### 3.5.1 Data Model

Consider the wave field generated by $K$ narrow band sources located at angle $\Theta_{1}, \theta_{2}, \ldots, \Theta_{k}$ from the sensor array as shown in figure.


Figure 1: Array Processing example

By using the complex envelop representation the array output can be represented as (in sense of superposition),

$$
x(t)=\sum_{k=1}^{K} a\left(\theta_{k}\right) s_{k}(t)+n(t)
$$

Where:
$x(t)$ is the signal received by the array.
$a\left(\theta_{k}\right)$ is the steering vector of the array towards the direction of source $k$.
$n(t)$ is the noise vector.
$s_{k}(t)$ is signal emitted by the $k^{t h}$ source as received by the first sensor of the array.

In vector forms, we can write it as

$$
x(t)=A(\theta) s(t)+n(t)
$$

Let us say we have only one snapshot (that is the case in our sudoku model) we may write

$$
X=A s+n
$$

Also if we assume there is no noise (as is the case in our sudoku model) we can skip the term $n$, and hence the model becomes

$$
X=A s
$$

By comparing it to sudoku model we can see that if we assume $X=b$ as in equation (1), A as in equation (1) and $s$ as $x$ (our required solution) then we can use this array processing model to reach to the solution $x$ which will eventually be our sudoku solution.

### 3.5.2 IAA-APES

IAA-APES is a non-parametric algorithm based on weighted least squares. Let us modify the model presented in [4] for our sudoku case.

Let $P$ be a $K \times K$ diagonal matrix, whose diagonal contains the power at each angle on the scanning environment. Then $P$ can be expressed as, for single snapshot case,

$$
P_{k}=\left|s_{k}\right|^{2}
$$

Also we do not have noise in the sudoku case. Then we need to minimize the following least square cost function:

$$
\sum_{n=1}^{N}\left\|y(n)-s_{k}(n) a\left(\theta_{k}\right)\right\|^{2}
$$

[4] contains the details of the algorithm derivations. We will present only the algorithm here to suit our sudoku case. For our case we have only one snapshot so $\mathrm{N}=1$.

Therefore our cost function for sudoku case is

$$
\left\|y-s_{k} a\left(\theta_{k}\right)\right\|^{2}
$$

The complete algorithm is presented in the next section.

### 3.5.2.1 Algorithm

$$
\widehat{P_{k}}=\frac{1}{\left(a^{T}\left(\theta_{k}\right) a\left(\theta_{k}\right)\right)^{2}}\left|a^{T}\left(\theta_{k}\right) y\right|^{2}, \quad k=1,2, \ldots, K
$$

Repeat

$$
R=A(\theta) \hat{P} A^{T}(\theta)
$$

for $k=1$ : K

$$
\begin{aligned}
& \hat{s}_{k}=\frac{a^{T}\left(\theta_{k}\right) R^{-1} y}{a^{T}\left(\theta_{k}\right) R^{-1} a\left(\theta_{k}\right)} \\
& \hat{P}_{k}=\left|\hat{s}_{k}\right|^{2}
\end{aligned}
$$

End for: Until convergence (15 iterations approximately are enough)

Comparing it with our sudoku case in equation (1)
$\mathrm{b}=\mathrm{y}, \mathrm{A}$ as in (1) and $\mathrm{s}=\mathrm{x}$

### 3.6 Sinkhorn Balancing

A slightly different algorithm than those we have already discussed is presented in [7]. The problem formulation for Sinkhorn balancing algorithm is different from those presented earlier. Sinkhorn balancing computes the solution based on a probabilistic representation of the sudoku puzzle. Sinkhorn balancing is presumed to solve all puzzles but the most difficult ones. We have included this method in our report to compare its results to our presented methods for the sudoku. For more details of this algorithm we refer to [7]. Here we only present the results of this algorithm on our Sudoku puzzles. We are thankful to Todd Moon for providing us the matlab code for Sinkhorn balancing.

## 4 Puzzles

In this thesis work we used the sudoku generator from [20] which creates sudoku of varying difficulty level and tries to create as hard as possible sudoku puzzles. It also gives the solution of the puzzles by using some simple and complex algorithms which guarantees to get to the solutions. One of the methods used for these solutions is as described in [9]. The method used in getting these solutions in sudoku generator are not mathematical but interactive as people used to do it on paper like guessing and back tracking. One of the popular algorithms of such category to solve the sudoku on computer is Dancing Links which uses programming techniques and data structures to solve the sudoku. As in this thesis work we are comparing different mathematical algorithms for the solution and we are interested in the performance of the algorithm in terms of the entries solved, it is good idea to have the solution of these sudoku in advance using some of above mentioned techniques to compare the results. It would at least give a good comparison of the solved entries for the puzzles which have unique solutions. As choosing the puzzles in [20] helped us having a solution in advance and mostly it tries to create varying level of difficulty with unique solution we are presented puzzle set 1 for these sudoku. Puzzle set 2 is taken from the web generator [21] with different difficulty levels and puzzle set 3 [22] also provides puzzles with multiple solutions without guarantee of uniqueness. Table 8, 9 and 10 in the Appendix contain the puzzles for these 3 sets respectively.

## 5 Results and Discussions

Here we will present different puzzle sets with different observations and conclusion.

### 5.1 Puzzle Set 1

Table 2 shows the results from the 50 randomly generated $9 \times 9$ puzzles as in [20]. This puzzle generation method tries to generate as hard as possible puzzles and also ensures that the puzzle has a unique solution. The used puzzles are shown in Appendix. Here we present the results of all our methods on these puzzles to see the comparison. Numeric entry in the table shows the number of unsolved or number of incorrectly solved entries out of total of 81 entries for each puzzle. As this puzzle generation method also gives the solution in advance, this is being matched with the results to get the incorrect entries.

Table 2: Comparison of methods for number of incorrectly solved entries of the puzzle set 1

|  | Linear <br> program <br> (L1) | Weighted <br> I1 method | SDR | Sinkhorn | SPICE | LIKES | IAA |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| PUZZLE 1 | 50 | 0 | 7 | 16 | 50 | 47 | 0 |
| PUZZLE 2 | 0 | 0 | 0 | 0 | 0 | 0 | 7 |
| PUZZLE 3 | 32 | 0 | 0 | 6 | 32 | 30 | 0 |
| PUZZLE4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 5 | 47 | 0 | 1 | 23 | 47 | 43 | 28 |
| PUZZLE 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 7 | 41 | 29 | 18 | 17 | 41 | 33 | 12 |
| PUZZLE 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 10 | 42 | 0 | 0 | 8 | 42 | 41 | 24 |
| PUZZLE 11 | 48 | 39 | 16 | 17 | 48 | 40 | 40 |
| PUZZLE 12 | 28 | 19 | 0 | 12 | 28 | 0 | 12 |
| PUZZLE 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 16 | 46 | 38 | 0 | 18 | 46 | 34 | 8 |
| PUZZLE 17 | 40 | 35 | 0 | 18 | 40 | 37 | 34 |
| PUZZLE 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 20 | 0 | 0 | 0 | 0 | 0 | 0 | 13 |
| PUZZLE 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 23 | 47 | 0 | 0 | 11 | 48 | 0 | 0 |
| PUZZLE 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 25 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| PUZZLE 26 | 0 | 0 | 0 | 0 | 0 | 0 | 34 |
| PUZZLE 27 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| PUZZLE 28 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| PUZZLEE 29 | 16 | 0 | 0 | 0 | 0 | 25 |  |
| PUZZLE 30 | 36 | 0 | 0 | 0 | 0 | 0 |  |
|  |  | 0 | 0 | 0 | 0 |  |  |


| PUZZLE 31 | 26 | 0 | 0 | 6 | 26 | 26 | 22 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| PUZZLE 32 | 51 | 40 | 0 | 16 | 51 | 41 | 37 |
| PUZZLE 33 | 43 | 21 | 0 | 17 | 43 | 42 | 18 |
| PUZZLE 34 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 35 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| PUZZLE 36 | 50 | 43 | 0 | 20 | 50 | 46 | 43 |
| PUZZLE 37 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 38 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 39 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 40 | 0 | 0 | 0 | 0 | 0 | 0 | 7 |
| PUZZLE 41 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 42 | 0 | 0 | 0 | 0 | 0 | 0 | 21 |
| PUZZLE 43 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 44 | 45 | 38 | 0 | 19 | 45 | 42 | 34 |
| PUZZLE 45 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 46 | 37 | 0 | 0 | 14 | 37 | 33 | 31 |
| PUZZLE 47 | 43 | 31 | 0 | 20 | 43 | 36 | 30 |
| PUZZLE 48 | 34 | 24 | 0 | 16 | 34 | 24 | 26 |
| PUZZLE 49 | 42 | 23 | 0 | 13 | 42 | 39 | 25 |
| PUZZLE 50 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 3 shows the number of solved puzzles out of total of 50 puzzles. The puzzles here are the same as are in Table 2.

Table 3: Comparison Table for Number of solved puzzles in puzzle set 1

|  | L1 | Weight I1 | SDR | Sinkhorn | SPICE | LIKES | IAA |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Total <br> Puzzels <br> solved | 29 | 38 | 46 | 28 | 29 | 31 | 28 |

Summarizing Table 2 and Table 3.

As expected the SPICE and I1-norm minimization are almost the same in terms of their performance.

LIKES is not much of improvement over SPICE in terms of number of solved sudoku as in Table 3. But the numbers of solved entries for LIKES are much better than those in SPICE see Table 2 for the corresponding columns of SPICE and LIKES. Also here weighted I1 norm minimization behaves better than the LIKES and as expected LIKES performs better than I1-norm minimization. IAA method performs somewhere around LIKES and SPICE in terms of accuracy, as we can see in Figure 3 that the execution speed for IAA is faster than those of SPICE and LIKES.

Sinkhorn method shows us the least accurate results than the other methods. Sinkhorn only can solve half of the puzzles and mostly for the hard puzzles the sinkhorn was unable to solve all the entries. It is also clear that the puzzles solved by sinkhorn are also solved by all the other methods. Therefore Sinkhorn is the lowest performer of our comparison table.

The best performer of our tried methods is SDR which solves more than $90 \%$ of the puzzles and those which are not solved have also the least number of unsolved entries than the other methods. So idea of using SDR is successful in terms of the accuracy of the method. However the huge difference lies in terms of computational complexity. In fact the computational complexity for SDR is worse than all of the other methods, which is kind of a dilemma for using SDR. However the accuracy is highest among all and it solves mostly the hard to extremely evil puzzles.


Figure 2: Execution time for First 20 puzzles (in seconds)


Figure 3: Increase in computation time with puzzle size

We also tried to get an idea to know how the computation speed increases with the increase of puzzle size. For this purpose we test randomly generated puzzles from different web resources of size $4 \times 4,6 \times 6$ and $9 \times 9$ respectively. We found the computation speed for each method to solve 20 set of puzzles of the above mentioned sizes and took the average computation speed for each method. Finally this average speed is plotted as shown in Figure 3. Vertical axis is the log scale of the average computation speed taken on 20 set of puzzles of each size.

### 5.2 Puzzle Set 2

Here we present other 40 puzzles generated by the web generator [21] with different level of difficulties. The puzzles used can be found in Appendix. Table entry 0 represents that puzzle has been solved with no errors while entry ' $X$ ' represents that puzzle was not solved completely.

Table 4: Comparison for puzzle set 2

|  | Difficulty Level | Linear program (L1) | Weighted I1 method | SDR | Sinkhorn | SPICE | LIKES | IAA |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| PUZZLE 1 | Easy | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 2 | Easy | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 3 | Easy | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 4 | Easy | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 5 | Easy | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 6 | Easy | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 7 | Easy | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 8 | Easy | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 9 | Easy | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 10 | Easy | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 11 | Medium | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 12 | Medium | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 13 | Medium | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 14 | Medium | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 15 | Medium | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 16 | Medium | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 17 | Medium | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 18 | Medium | 0 | 0 | 0 | 0 | 0 | 0 | X |
| PUZZLE 19 | Medium | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 20 | Medium | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 21 | Hard | 0 | 0 | 0 | 0 | 0 | 0 | X |
| PUZZLE 22 | Hard | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 23 | Hard | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 24 | Hard | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 25 | Hard | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 26 | Hard | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 27 | Hard | X | 0 | X | X | X | 0 | 0 |
| PUZZLE 28 | Hard | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 29 | Hard | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 30 | Hard | 0 | 0 | 0 | 0 | 0 | 0 | X |
| PUZZLE 31 | Evil | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 32 | Evil | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 33 | Evil | 0 | 0 | 0 | 0 | 0 | 0 | 0 |


| PUZZLE 34 | Evil | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| PUZZLE 35 | Evil | 0 | 0 | 0 | 0 | 0 | 0 | X |
| PUZZLE 36 | Evil | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 37 | Evil | 0 | 0 | 0 | 0 | 0 | 0 | X |
| PUZZLE 38 | Evil | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 39 | Evil | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZZLE 40 | Evil | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 5: Puzzle set 2, number of solved puzzles

|  | LP | WLP | SDR | SINKHORN | SPICE | LIKES | IAA |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Easy | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| Medium | 10 | 10 | 10 | 10 | 10 | 10 | 9 |
| Hard | 9 | 10 | 9 | 9 | 9 | 10 | 8 |
| Evil | 10 | 10 | 10 | 10 | 10 | 10 | 8 |

Most of the puzzles generated by the web generator for puzzle set 2 are easily solved by each of the methods. However there exits only 1 strange puzzle that is not solved by any, actually this puzzle has multiple solutions and in the next puzzle set we will look into more detailed results of multi solution sudoku.

### 5.3 Puzzle Set 3

Following puzzles are generated from web generator [22]. This generator mostly generates puzzles without guarantee of uniqueness as opposite to puzzle set 1 which generates puzzles with unique solutions. Here the solved puzzles by the algorithm are mostly distinct from the one we have from the puzzle generator. Here we will focus more on SDR and will see how rank deficiency effects the accuracy as depicted in Figure 4 \& 5. Type of solution shows the number of solutions exists for the puzzles. To find the number of solutions a puzzle contains we used the web source [23]. Tick sign indicates that puzzle is solved successfully while cross sign represents failure.

Table 6: Comparison Table for Puzzle Set 3

|  | Difficult <br> y Level | Linear <br> progra <br> m (L1) | Weighte <br> d I1 <br> method | SDR | Sinkhorn | SPICE | LIKES | IAA | Type of <br> Solution |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| PUZZLE 1 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |
| PUZZLE 2 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |
| PUZZLE 3 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |
| PUZZLE 4 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |
| PUZZLE 5 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |
| PUZZLE 6 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |
| PUZZLE 7 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |
| PUZZLE 8 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |
| PUZZLE 9 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |
| PUZZLE 10 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |
| PUZZLE 11 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |


| PUZZLE 12 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | unique |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| PUZZLE 13 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 14 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 15 | Easy | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 16 | Medium | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 4 |
| PUZZLE 17 | Medium | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 6 |
| PUZZLE 18 | Medium | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 19 | Medium | X | $\checkmark$ | X | $\checkmark$ | X | $\checkmark$ | $\checkmark$ | 5 |
| PUZZLE 20 | Medium | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 21 | Medium | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 22 | Medium | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 23 | Medium | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 24 | Medium | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 25 | Medium | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 26 | Medium | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 27 | Medium | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 28 | Medium | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 4 |
| PUZZLE 29 | Medium | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 7 |
| PUZZLE 30 | Medium | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 18 |
| PUZZLE 31 | Hard | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 9 |
| PUZZLE 32 | Hard | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 33 | Hard | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 34 | Hard | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 14 |
| PUZZLE 35 | Hard | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 6 |
| PUZZLE 36 | Hard | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 3 |
| PUZZLE 37 | Hard | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 54 |
| PUZZLE 38 | Hard | X | $\checkmark$ | X | X | X | X | $\checkmark$ | 13 |
| PUZZLE 39 | Hard | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 40 | Hard | X | $\checkmark$ | X | X | X | $\checkmark$ | X | 6 |
| PUZZLE 41 | Evil | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 12 |
| PUZZLE 42 | Evil | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 102 |
| PUZZLE 43 | Evil | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 54 |
| PUZZLE 44 | Evil | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 23 |
| PUZZLE 45 | Evil | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | Unique |
| PUZZLE 46 | Evil | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 103 |
| PUZZLE 47 | Evil | X | X | X | X | X | $\checkmark$ | $\checkmark$ | 67 |
| PUZZLE 48 | Evil | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 145 |
| PUZZLE 49 | Evil | X | $\checkmark$ | X | X | X | $\checkmark$ | $\checkmark$ | 3 |
| PUZZLE 50 | Evil | X | $\checkmark$ | X | X | X | X | X | 92 |

Table 7: Total Puzzles solved for puzzle set 3

|  | L1 | Weight I1 | SDR | Sinkhorn | SPICE | LIKES | IAA |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Total <br> Puzzels <br> solved | 28 | 49 | 28 | 29 | 28 | 48 | 48 |

In the multi solution environment, IAA, LIKES and weighted I1 proved to be the best however SDR is not as optimal. Sinkhorn also performed poorly for this puzzle set.

Figure 4 and 5 show the singular value plots for unsolved puzzle 27 of puzzle set 2 . It can be seen that matrix rank is not 1 as the singular component other than 1 are not 0 . As in SDR we skip the constraints of matrix rank to be one, so here we can see that the more better the rank 1 approximation holds the more accurate the result is. Following figures show this scenario for one correct and one incorrectly solved puzzles.


Figure 4: Singular Value Plot for incorrectly solved puzzle

Following figure shows singular values for Puzzle 36, a solved puzzle. As the rank is approximately 1 as the first singular values is more dominant and others are approximately near zero.


Figure 5: Singular value plot for correctly solved puzzle

## 6 Conclusion and Future Work

Inspired by [1] and [5] we presented sudoku as a system of underdetermined linear system of equations. This system is being solved with a number of approximate methods and it is been presented in the report that the approximate solution techniques for solving sudoku suffers in accuracy as compared to the exact solutions but the gain is that the computations complexity for approximate solvers is better when the puzzle size increases. Accuracy is the key for solving sudoku or any other puzzle, however as our goal is not meant for sudoku only in the long term, the research presented in the report can be useful to extend to some of the more practical communication and signal processing areas as solving sudoku has ties with scenarios in the communication world like decoding error correcting codes.

We showed numerically that how mathematical techniques used in signal processing and optimization theory can solve many representative puzzles. Semi Definite Relaxation (SDR) proved to be the best in accuracy for the puzzles with unique solutions by solving more than $90 \%$ puzzles. SPICE and 11 -norm minimization techniques are equivalent in accuracy, in fact for our puzzle sets used SPICE and I1-norm have exactly same accuracy in terms of number of solved puzzles. Weighted I1norm minimization and LIKES are equivalent in the accuracy and they are better than I1-norm and SPICE with increased cost of computation speed. When the puzzle solutions are not unique, SDR performance seems to be degraded mainly because the rank 1 approximation has more error for these puzzles. LIKES and weighted I1-norm performed best for these puzzles and they mostly converge to one of the possible solutions.

The real challenge in using SDR is that the computation speed is worst and when the size of the puzzle is increased the result for solving a puzzle can take several minutes and hours. LIKES, weighted 11 also work slowly due to their iterative nature. To achieve the goal of SPICE and LIKES with increased computation speed, the Iterative Adaptive Algorithm (IAA) was presented which can provide accuracy somewhere near SPICE and LIKES with better computation speed. Results were compared with Sinkhorn balancing whose accuracy seems to be the worse of all.

One can further research on the possibility of better rank 1 approximation of SDR to get the better results. Sparsity is desirable in many applications. The sparse solution representation of sudoku can be further utilized to suit many practical areas. As mentioned earlier many practical scenarios in telecommunication and signal processing exists which are similar to sudoku puzzles, work can be done to utilize the good algorithms working for sudoku to those areas and vice versa.

## 7 References

[1] A. C. Bartlett and A. N. Langville, "An integer programming model for the Sudoku problem," Preprint, available at http://www. cofc. edu/langvillea/Sudoku/sudoku2. pdf, 2006.
[2] P. Stoica and P. Babu, "SPICE and LIKES: Two hyperparameter-free methods for sparseparameter estimation," Signal Processing, vol. 92, no. 7, pp. 1580-1590, 2012.
[3] Z. q. Luo, W. k. Ma, A. C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," Signal Processing Magazine, IEEE, vol. 27, no. 3, pp. 20-34, 2010.
[4] T. Yardibi, J. Li, P. Stoica, M. Xue, and A. B. Baggeroer, "Source localization and sensing: A nonparametric iterative adaptive approach based on weighted least squares," Aerospace and Electronic Systems, IEEE Transactions on, vol. 46, no. 1, pp. 425-443, 2010.
[5] P. Babu, K. Pelckmans, P. Stoica, and J. Li, "Linear systems, sparse solutions, and Sudoku," Signal Processing Letters, IEEE, vol. 17, no. 1, pp. 40-42, 2010.
[6] Y. B. Zhao and D. Li, "Reweighted \ell_1-Minimization for Sparse Solutions to Underdetermined Linear Systems," SIAM Journal on Optimization, vol. 22, no. 3, pp. 10651088, 2012.
[7] T. K. Moon, J. H. Gunther, and J. J. Kupin, "Sinkhorn solves sudoku," Information Theory, IEEE Transactions on, vol. 55, no. 4, pp. 1741-1746, 2009.
[8] T. Davis, "The mathematics of Sudoku," [Online] available at http://www. geometer. org/mathcircles/sudoku. pdf, 2006.
[9] M. Mepham, "Solving sudoku," Daily Telegraph [Online]. Available: http://www. sudoku. org. uk/PDF/Solving_Sudoku. pdf., 2005.
[10] "Sudoku Solving Algorithms," (2014) Retrieved January 19, 2015, from http://en.wikipedia.org/wiki/Sudoku solving algorithms
[11] D. E. Knuth, "Dancing links," Millenial Perspectives in Computer Science, vol. 18, no. arXiv: cs/0011047, p. 4, 2009.
[12] "Knuth's Algorithm X," (2014). Retrieved January 19, 2015, from http://en.wikipedia.org/wiki/Algorithm X
[13] H. Simonis, "Sudoku as a constraint problem,", In CP Workshop on modeling and reformulating Constraint Satisfaction Problems, 12 ed Citeseer, 2005, pp. 13-27.
[14] J. Almog, "Evolutionary computing methodologies for constrained parameter, combinatorial optimization: Solving the Sudoku puzzle," in Proc. IEEE AFRICON, 2009, pp. 1-6.
[15] R. Lewis, "Metaheuristics can solve sudoku puzzles," Journal of heuristics, vol. 13, no. 4, pp. 387-401, 2007.
[16] "Tabu Search," (2014). Retrieved January 19, 2015, from http://en.wikipedia.org/wiki/Tabu search
[17] T. K. Moon and J. H. Gunther, "Multiple constraint satisfaction by belief propagation: An example using Sudoku," in Mountain Workshop IEEE, 2006, pp. 122-126.
[18] J. Gunther and T. Moon, "Entropy minimization for solving Sudoku," Signal Processing, IEEE Transactions on, vol. 60, no. 1, pp. 508-513, 2012.
[19] M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab software for disciplined convex programming," 2008 [Online]. Available: http://stanford.edu/~boyd/cvx
[20] [Online]. Available: http://www.mathworks.se/matlabcentral/fileexchange/28168-sudokugenerator
[21] [Online]. Available: http://www.websudoku.com/
[22] [Online]. Available: http://www.mathworks.se/matlabcentral/fileexchange/13846-solve-and-create-sudoku-puzzles-for-different-levels/all_files
[23] [Online]. Available: http://www.sudoku-solutions.com/

## Appendix

Table 8: Puzzle Set 1

| Puzzle1 |  |  |  |  |  |  |  |  | Puzzle2 |  |  |  |  |  |  |  | Puzzle3 |  |  |  |  |  |  |  |  | Puzzle4 |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 3 |  | 9 |  |  | 8 | 4 |  |  |  |  |  |  |  | 3 |  | 8 |  |  | 7 | 6 |  |  |  | 1 |  |  |  |  | 9 |  |  |  | 1 |
|  | 7 |  | 3 |  |  |  | 5 |  | 3 |  |  |  |  | 7 |  |  |  | 1 | 7 |  |  |  |  |  |  |  |  |  |  | 2 |  |  | 7 | 3 |
|  |  |  |  |  | 1 |  | 3 |  | 4 |  | 8 |  |  |  |  |  |  |  | 9 | 2 |  | 5 | 7 |  | 6 | 1 |  |  |  |  | 7 | 6 | 5 | 2 |
|  | 5 |  |  | 4 |  |  |  |  | 6 |  |  | 2 |  | 31 |  |  | 5 |  | 1 |  |  |  |  | 9 |  | 9 | 8 |  | 5 |  |  | 1 |  |  |
|  |  |  |  |  |  | 9 |  | 4 |  |  |  |  |  |  | 2 |  |  | 7 | 2 |  |  |  |  |  |  | 7 | 4 |  |  |  |  |  | 6 |  |
| 6 |  |  |  |  |  |  | 2 |  |  | 5 | 9 |  |  | 1 | 8 |  |  |  |  | 3 |  |  |  | 5 |  |  |  |  | 2 |  | 6 |  |  |  |
|  | 4 | 5 |  |  |  |  | 8 | 6 |  |  |  |  | 4 |  | 1 |  |  |  |  | 4 |  |  |  |  | 8 |  |  |  | 3 |  |  |  |  |  |
| 7 |  | 9 | 6 |  |  |  |  |  | 8 | 1 |  | 93 | 3 | 4 |  | 2 | 1 |  | 8 |  |  | 9 | 3 |  |  |  |  |  |  |  | 4 |  |  |  |
|  |  | 2 |  |  | 3 |  |  |  |  |  | 5 |  |  |  |  | 9 | 2 | 5 | 3 |  |  |  |  |  |  | 4 |  |  |  |  | 1 | 8 |  | 7 |
| Puzzle5 |  |  |  |  |  |  |  |  | Puzzle6 |  |  |  |  |  |  |  | Puzzle7 |  |  |  |  |  |  |  |  | Puzzle8 |  |  |  |  |  |  |  |  |
|  |  | 1 |  |  |  |  |  |  |  | 2 |  |  | 6 | 9 |  |  |  | 5 |  | 3 |  |  |  |  | 1 |  | 2 |  |  |  |  |  |  |  |
|  |  | 9 | 5 |  | 7 |  |  | 2 | 8 |  |  |  |  | 5 |  |  | 9 |  |  |  |  | 4 | 2 | 6 |  | 3 |  |  |  | 8 |  | 9 |  |  |
| 7 |  |  |  |  |  | 5 | 6 |  |  |  |  |  |  |  | 3 |  | 7 |  | 8 | 2 |  |  |  | 9 |  |  |  |  |  |  |  |  |  | 8 |
|  |  |  |  |  | 2 |  |  | 9 | 5 | 8 |  |  |  |  |  |  |  |  | 5 |  |  |  | 6 |  |  |  |  |  |  |  |  |  | 4 |  |
|  |  |  |  | 9 |  | 1 |  |  |  |  | 3 | 51 | 1 |  |  |  |  |  |  |  | 1 |  |  | 2 |  |  | 5 |  | 3 | 6 | 1 |  |  |  |
|  | 8 | 2 |  | 4 | 6 |  |  |  |  |  | 4 |  | 92 | 2 |  | 6 |  |  |  | 9 |  | 7 |  |  |  | 9 |  |  | 8 | 2 |  | 1 |  |  |
|  | 2 | 7 |  |  |  |  | 8 |  |  |  |  |  |  | 7 | 74 |  |  |  |  |  | 7 |  | 9 | 1 |  |  | 7 |  | 5 |  |  | 8 |  |  |
| 4 |  |  | 2 | 8 | 3 |  |  |  |  |  |  |  |  | 62 |  | 9 |  |  | 1 |  |  |  |  |  | 4 |  | 3 | 1 |  |  |  | 7 |  |  |
|  |  |  |  |  | 4 |  |  |  |  | 5 |  |  |  | 7 | 1 | 8 |  | 3 |  |  |  |  | 7 |  |  |  | 8 | 5 | 7 |  | 2 |  | 6 |  |
| Puzzle9 |  |  |  |  |  |  |  |  | Puzzle10 |  |  |  |  |  |  |  | Puzzle11 |  |  |  |  |  |  |  |  | Puzzle12 |  |  |  |  |  |  |  |  |
|  |  |  | 1 |  | 3 | 9 |  |  |  |  |  | 6 |  | 87 |  |  |  | 1 | 9 |  | 7 | 6 | 4 |  | 2 |  |  |  |  |  |  |  | 1 |  |
| 5 |  |  | 7 |  |  |  | 4 |  |  |  | 3 |  |  | 18 | 8 |  | 6 | 4 |  |  |  |  |  |  | 7 |  |  |  |  |  | 2 | 3 |  |  |
|  |  | 8 |  |  | 4 |  | 5 |  |  |  | 5 | 3 |  |  |  |  |  |  |  |  | 2 |  |  |  |  | 6 | 2 | 4 | 3 |  |  |  |  |  |
|  |  |  |  |  | 8 | 1 |  |  |  |  |  |  |  | 79 |  |  | 8 |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  | 8 | 4 |
|  | 4 |  |  |  |  |  |  | 6 |  |  |  | 4 |  | 3 | 31 |  |  |  |  |  |  | 2 | 7 |  | 8 |  | 4 | 1 | 7 |  |  |  | 3 |  |
| 2 | 9 |  |  |  | 7 |  |  |  |  |  | 8 | 9 |  |  |  |  |  |  |  |  |  |  | 1 | 3 |  |  |  | 2 | 1 |  |  |  | 5 |  |
| 3 |  | 7 | 9 |  |  |  | 2 |  | 5 | 4 |  |  |  | 1 |  | 8 |  |  |  |  | 9 | 4 |  |  |  | 2 |  |  |  |  | 8 |  |  |  |
|  | 2 |  | 6 |  |  |  |  |  | 3 |  |  |  | 2 |  |  | 4 |  | 6 |  |  |  |  |  |  |  |  |  | 8 |  |  |  |  |  | 1 |
|  |  |  |  |  |  | 7 |  |  |  |  | 7 |  |  |  | 9 |  |  | 5 | 1 | 2 | 6 | 7 |  |  | 9 |  | 3 | 6 | 2 |  | 9 |  |  | 7 |
| Puzzle13 |  |  |  |  |  |  |  |  | Puzzle14 |  |  |  |  |  |  |  | Puzzle15 |  |  |  |  |  |  |  |  | Puzzle16 |  |  |  |  |  |  |  |  |
|  | 8 |  | 3 |  |  | 4 | 2 |  |  |  |  | 4 |  | 5 | 1 |  | 1 |  |  |  | 9 | 4 |  | 6 |  |  | 3 |  | 7 |  |  |  |  | 4 |
|  |  | 7 |  | 2 |  |  |  |  |  |  |  |  | 6 | 4 | 4 |  |  |  |  |  |  |  |  |  | 4 |  | 4 |  |  |  |  |  |  |  |
|  |  | 2 |  |  | 1 |  |  |  |  |  |  |  |  | 3 | 3 | 8 |  |  |  |  |  |  |  | 7 | 2 |  |  |  |  | 9 |  |  | 7 | 8 |
| 8 |  |  |  |  | 3 |  |  |  | 2 | 5 |  | 9 | 8 | 1 | 1 |  |  |  | 7 | 3 |  | 9 |  |  |  |  |  |  | 3 | 8 |  |  |  | 6 |
| 6 |  |  |  |  | 5 |  |  | 7 |  |  |  |  |  | 6 |  |  | 4 | 3 |  |  |  |  |  | 1 |  |  |  | 2 |  |  | 9 |  |  |  |
|  |  | 3 |  | 6 | 4 | 1 |  |  |  | 7 |  |  |  | 2 |  |  |  |  | 5 |  | 1 |  |  | 4 |  |  |  | 9 |  | 5 |  |  | 2 |  |
|  |  |  | 4 | 9 |  | 2 |  | 3 |  | 4 |  |  |  |  |  |  |  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |


|  |  |  |  |  | 7 |  |  | 8 | 2 |  | 7 | 3 |  |  | 2 |  |  |  |  | 6 | 6 | 3 |  | 1 | 7 | 6 |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 7 | 7 |  |  |  |  | 45 | 5 | 9 | 7 |  |  |  |  | 2 | 6 | 4 |  |  | 5 | 5 |  |  |  | 8 |  | 2 | 4 |  | 1 |  |  |
| Puzzle17 |  |  |  |  |  |  |  | Puzzle18 |  |  |  |  |  |  | Puzzle19 |  |  |  |  |  |  |  | Puzzle20 |  |  |  |  |  |  |  |  |
| 4 |  |  |  | 3 | 5 | 6 |  |  |  |  |  | 1 | 34 | 4 |  |  |  |  | 6 | 6 |  |  |  | 8 |  | 2 |  |  |  | 7 | 1 |
|  |  |  |  | 9 | 73 | 3 |  |  |  |  |  |  |  | 6 | 6 |  |  |  |  | 8 | 8 |  |  |  |  |  |  |  |  |  |  |
|  | 1 |  |  | 5 |  |  |  | 2 | 4 |  | 8 |  | 9 |  |  |  | 2 |  | 35 | 54 | 4 |  |  | 3 |  | 8 | 1 |  |  | 4 |  |
| 6 | 6 |  | 3 |  |  |  |  |  |  | 8 |  | 9 |  | 5 |  |  |  |  |  |  |  |  |  | 6 | 9 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | 6 |  |  |  | 2 |  | 5 |  | 8 |  |  | 4 |  |  |  |  |  |  | 8 |  | 3 | 9 |
| 3 | 3 | 9 | 2 |  |  | 7 | 8 | 2 |  |  | 4 |  |  |  | 7 | 43 | 3 |  | 9 |  | 5 |  |  |  | 3 | 6 |  | 7 |  |  |  |
|  |  | 4 |  | 9 | 1 |  | 2 | 6 |  |  |  |  |  | 7 |  |  |  |  |  | 2 | 2 | 5 |  | 4 |  | 1 | 6 |  | 5 | 8 |  |
|  |  |  |  |  |  |  |  | 4 | 4 | 3 | 5 |  |  |  |  | 6 |  |  |  |  |  | 9 |  |  |  |  |  |  |  | 9 |  |
|  | 7 |  |  |  | 6 | 4 |  | 1 |  |  |  |  |  |  |  | 7 |  | 2 |  | 1 | 1 |  |  |  |  |  | 7 |  | 3 |  |  |
| Puzzle21 |  |  |  |  |  |  |  | Puzzle22 |  |  |  |  |  |  | Puzzle23 |  |  |  |  |  |  |  | Puzzle24 |  |  |  |  |  |  |  |  |
|  | 2 |  | 4 | 3 |  | 6 |  |  |  |  | 8 |  |  |  |  |  |  |  | 69 | 93 | 3 |  |  |  | 4 | 7 |  | 9 |  |  | 5 |
|  | 6 |  |  | 9 |  |  |  | 3 |  |  |  |  |  | 67 |  |  |  |  |  | 5 |  |  |  |  |  |  |  |  | 3 |  |  |
|  |  |  |  | 1 |  |  | 4 | 5 |  |  | 2 | 1 | 8 |  | 6 |  | 1 |  |  |  |  | 7 |  |  |  |  |  |  |  |  | 7 |
|  | 3 | 5 | 7 |  | 9 | 2 |  | 8 |  |  | 3 |  |  |  | 3 |  |  |  |  |  | 7 |  |  |  | 9 |  |  | 6 | 8 |  |  |
|  |  |  |  |  |  | 5 |  | 6 | 7 |  | 9 |  |  | 1 |  |  | 6 |  | 84 |  |  |  |  | 6 | 3 |  | 9 |  |  |  |  |
|  | 7 |  |  |  |  | 3 |  | 9 |  | 1 |  |  |  |  | 5 | 8 |  |  | 3 |  | 1 | 6 |  | 8 | 5 |  |  | 4 | 1 |  |  |
|  |  | 9 | 27 | 7 | 5 |  |  | 6 |  |  |  | 5 | 1 | 8 |  | 2 |  |  |  |  | 5 |  |  |  |  |  | 6 |  | 5 | 9 |  |
| 4 | 4 |  | 5 |  |  |  |  | 8 |  | 9 |  |  | 4 |  |  |  |  |  | 1 | 1 |  |  |  |  |  |  | 2 |  |  |  |  |
|  |  |  |  | 1 |  |  | 2 | 4 |  |  |  |  |  | 9 |  |  | 9 | 4 | 7 | 7 |  | 2 |  | 5 | 6 | 9 | 8 | 7 |  | 3 | 4 |
| Puzzle25 |  |  |  |  |  |  |  | Puzzle26 |  |  |  |  |  |  | Puzzle27 |  |  |  |  |  |  |  | Puzzle28 |  |  |  |  |  |  |  |  |
|  |  | 8 | 2 |  | 5 |  |  | 3 |  |  |  |  |  | 5 |  |  |  |  | 9 |  | 7 |  |  | 9 | 2 |  | 4 |  |  | 8 |  |
| 5 | 5 |  |  |  |  |  | 2 | 4 | 6 | 9 |  |  |  |  | 6 |  |  |  |  |  |  |  |  | 6 |  |  |  |  |  | 3 | 9 |
|  | 7 |  | 3 |  |  |  | 9 | 9 |  |  |  | 4 |  |  | 3 |  |  |  |  |  | 5 | 1 |  |  | 8 |  |  | 6 |  |  |  |
|  |  | 6 | 4 | 4 | 1 |  |  | 5 | 4 |  |  | 6 |  |  |  | 5 |  | 6 |  |  |  | 7 |  |  |  |  |  |  | 7 |  |  |
| 2 | 2 |  | 8 | 6 |  |  |  |  |  |  |  | 5 |  | 68 | 9 | 86 | 6 |  | 4 | 4 |  |  |  | 2 |  | 5 |  |  |  | 6 | 3 |
| 1 | 1 |  |  |  |  |  | 7 |  |  | 8 |  | 2 |  |  |  |  |  |  |  |  | 4 | 2 |  |  |  | 2 |  | 3 |  |  | 5 |
|  | 6 |  |  | 7 |  |  | 5 |  |  |  |  |  |  |  |  | 41 | 1 |  | 7 |  |  |  |  | 4 | 7 |  |  | 9 |  |  |  |
|  |  |  |  | 9 |  | 3 |  | 8 |  | 2 | 3 |  |  | 19 | 2 |  |  |  | 6 |  |  | 4 |  |  |  | 6 |  |  |  | 9 |  |
|  |  |  | 62 | 24 |  |  |  |  | 9 |  | 1 |  | 2 |  |  |  |  | 5 |  |  |  |  |  |  |  | 3 | 7 |  |  |  |  |
| Puzzle29 |  |  |  |  |  |  |  | Puzzle30 |  |  |  |  |  |  | Puzzle31 |  |  |  |  |  |  |  | Puzzle32 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | 3 |  |  |  | 2 |  | 9 |  |  | 1 |  | 3 |  |  |  |  |  | 2 |  |  | 9 |  |
|  |  |  |  | 8 | 2 | 9 | 6 | 65 |  |  |  | 1 |  |  |  | 1 |  |  |  |  |  |  |  |  |  | 5 |  | 7 | 2 | 1 |  |
| 2 | 2 |  | 5 |  |  |  | 4 |  | 2 |  | 7 |  |  | 1 |  |  |  |  | 65 | 5 |  | 8 |  | 4 |  | 9 |  |  | 7 | 6 |  |
| 5 | 5 |  |  | 9 |  |  |  | 8 |  | 1 |  |  |  | 7 | 3 |  | 5 | 1 | 4 |  |  |  |  | 9 |  |  | 7 |  |  |  |  |
|  |  |  |  | 37 |  | 8 |  |  |  | 6 | 2 |  |  |  |  |  | 9 |  |  | 5 |  | 2 |  |  |  |  |  | 8 |  |  | 6 |
|  | 4 | 7 | 2 | 6 |  |  |  | 3 |  | 8 |  |  |  | 9 | 1 |  |  |  |  |  | 7 |  |  | 3 |  |  |  | 5 | 4 |  | 9 |
|  | 2 |  | 6 |  |  |  | 7 |  |  | 9 |  |  |  | 6 | 9 |  |  | 3 |  |  |  |  |  |  | 8 |  |  | 9 |  |  |  |
| 7 | 7 | 1 |  | 2 | 4 |  |  | $1{ }^{1} 3$ |  | 7 |  |  |  | 5 |  |  |  |  | 8 | 86 | 6 | 5 |  | 1 | 3 |  |  | 4 |  |  |  |
|  |  |  |  | 3 |  |  |  | 4 |  |  |  |  |  | 98 | 2 |  | 6 |  |  |  |  |  |  | 7 |  | 2 |  |  |  |  |  |
| Puzzle33 |  |  |  |  |  |  |  | Puzzle34 |  |  |  |  |  |  | Puzzle35 |  |  |  |  |  |  |  | Puzzle36 |  |  |  |  |  |  |  |  |
|  |  |  | 43 | 3 |  |  |  | 9 |  |  |  |  |  | 8 |  |  | 2 |  |  | 9 | 9 | 8 |  |  | 3 |  |  |  | 6 |  | 2 |
|  | 85 |  |  |  | 3 |  | 1 |  |  | 6 | 9 | 7 |  | 2 |  | 5 |  |  | 4 | 4 | 6 |  |  | 9 |  |  | 5 |  |  |  | 1 |



| 9 | 8 |  |  |  |  |  |  | 6 |  |  |  | 3 |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |

Table 9: Puzzle set 2

| Puzzle1 |  |  |  |  |  |  |  |  | Puzzle2 |  |  |  |  |  |  |  |  | Puzzle3 |  |  |  |  |  |  |  |  | Puzzle4 |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 7 | 8 | 5 | 4 | 2 |  | 3 | 9 | 9 1 |  | 9 | 6 | 5 |  | 3 |  |  | 72 |  | 4 |  | 3 |  |  | 1 |  | 2 |  | 7 |  |  | 2 | 4 | 1 | 3 | 9 |
| 2 | 3 |  | 7 |  | 9 | 5 | 54 | 4 | 3 | 2 |  | 8 | 9 |  | 5 | 6 | 64 | 7 | 8 | 85 | 4 | 2 | 2 | 63 | 1 |  |  | 9 |  | 1 |  | 3 |  | 8 |  |
|  | 4 | 6 | 3 | 5 | 1 | 8 | 2 | 27 | 8 | 7 |  |  | 6 |  | 3 | 1 | 1 | 2 | 3 | 31 | 7 | 8 | 89 | 95 | 56 | 4 |  |  | 1 | 9 | 8 | 7 | 6 | 5 |  |
| 6 | 1 | 9 |  | 4 |  | 2 | 25 | 53 | 7 | 5 | 3 | 9 | 2 | 6 | 4 | 8 | 8 |  | 7 | 73 |  | 9 |  | 2 | 8 |  |  |  | 9 | 7 |  |  | 3 | 2 | 5 |
| 5 |  |  | 6 |  |  | 4 |  | 8 | 2 | 4 | 8 | 1 | 3 | 5 | 7 |  | 6 | 6 | 1 | 9 |  |  |  | 7 |  | 5 | 2 |  | 8 | 3 |  | 5 | 9 | 7 |  |
|  |  | 8 |  |  |  | 7 | 76 | 6 | 1 | 6 | 9 | 4 | 7 | 8 |  | 3 | 35 |  | 2 |  | 5 | 1 | 1 | 7 | 79 |  |  | 5 | 3 | 2 |  | 6 | 8 |  |  |
| 1 | 9 |  | 2 | 3 |  | 6 | 68 | 85 | 5 |  | 4 |  | 8 |  | 1 | 2 | 27 |  | 5 | 4 | 9 | 6 |  | 8 | 2 | 7 | 6 | 8 |  | 5 | 7 |  | 4 | 9 | 3 |
|  | 6 | 2 |  | 7 |  | 9 | 3 | 34 | 6 | 8 | 2 | 7 |  | 1 | 9 | 4 |  |  | 9 | 97 | 2 | 3 |  | 6 | 65 | 8 | 9 |  | 7 | 4 |  |  | 5 | 6 | 8 |
| 3 | 5 | 4 | 9 |  | 8 |  |  |  |  | 1 | 7 | 3 | 4 | 2 |  |  | 8 | 8 | 6 | 2 | 1 | 7 |  | 59 | 94 | 3 |  |  | 4 | 8 | 6 | 9 | 2 | 1 | 7 |
| Puzzle5 |  |  |  |  |  |  |  |  | Puzzle6 |  |  |  |  |  |  |  |  | Puzzle7 |  |  |  |  |  |  |  |  | Puzzle8 |  |  |  |  |  |  |  |  |
| 4 | 9 |  |  | 5 | 3 | 2 | 7 | 78 | 9 | 6 | 4 | 5 |  | 3 | 8 | 2 |  |  | 2 | 1 | 9 |  |  | 7 | 6 |  | 7 | 5 | 8 | 6 | 2 | 4 | 1 |  |  |
| 8 | 7 |  | 6 |  | 4 |  |  | 3 | 7 | 5 | 8 |  | 6 | 4 | 3 |  |  | 8 | 7 | 5 | 6 |  |  |  | 1 |  |  | 6 | 4 |  |  | 3 |  | 2 |  |
|  | 2 | 1 | 9 | 8 |  | 4 | 6 | 65 |  | 1 |  |  | 9 | 7 |  | 4 | 46 |  |  |  |  | 5 |  | 32 | 7 | 8 | 2 | 1 |  | 9 | 8 |  |  | 4 | 5 |
| 1 | 6 |  |  | 4 |  | 5 | 3 | 32 | 4 | 8 | 2 |  | 3 | 5 | 7 | 6 |  |  | 6 |  | 7 | 4 |  | 85 | 3 |  | 4 | 8 |  | 3 |  | 5 |  |  | 7 |
| 7 | 5 |  |  |  |  |  |  | 4 | 5 | 3 |  | 9 | 2 |  | 4 | 1 | 1 |  | 4 |  | 3 | 1 |  | 56 |  | 7 | 5 |  | 7 | 2 |  | 6 | 8 | 1 | 4 |
|  | 4 | 8 | 3 | 1 | 5 | 6 |  |  |  |  | 1 | 4 |  | 8 | 2 | 5 |  |  | 5 | 3 | 2 | 9 | 9 | 1 | 8 | 4 |  |  |  | 7 | 4 |  | 3 | 5 | 2 |
| 6 | 8 | 2 | 5 | 7 | 1 | 3 | 4 | 4 | 8 | 2 | 6 |  | 5 |  | 9 | 3 | 34 | 9 |  | 7 | 4 |  |  |  | 5 | 6 | 1 | 7 | 9 | 4 | 3 | 2 |  |  | 6 |
| 9 | 1 |  |  |  |  |  | 5 | 56 | 1 | 7 | 9 |  | 4 | 2 | 6 |  | 5 | 5 | 3 | 4 | 8 | 6 |  | 97 | 72 | 1 | 3 | 4 |  | 8 |  | 9 | 2 | 7 |  |
| 5 | 3 | 4 | 8 | 6 | 9 | 7 | 72 | 21 |  | 4 | 5 | 6 |  |  | 1 |  | 2 | 6 |  | 2 | 5 | 7 |  | 1 | 4 | 9 | 8 | 2 | 6 |  | 7 | 1 | 4 | 3 | 9 |
| Puzzle9 |  |  |  |  |  |  |  |  | Puzzle10 |  |  |  |  |  |  |  |  | Puzzle11 |  |  |  |  |  |  |  |  | Puzzle12 |  |  |  |  |  |  |  |  |
| 8 | 7 | 5 | 2 |  |  | 9 | 1 |  | 8 | 5 |  | 2 |  | 4 | 9 |  | 1 |  | 6 |  | 1 |  |  | 32 | 27 |  |  |  |  | 7 |  |  |  | 4 | 5 |
| 4 | 9 |  | 5 | 3 |  |  | 7 | 7 | 4 |  | 9 |  | 1 | 3 | 2 | 8 | 87 | 7 |  |  | 6 | 2 |  | 49 | 1 | 3 |  |  | 5 |  |  |  |  | 9 | 3 |
| 3 | 2 |  | 8 |  | 9 |  | 6 | 65 | 3 | 1 | 2 | 8 | 9 | 7 |  |  | 6 | 2 | 1 |  |  | 8 |  | 7 |  | 5 | 9 |  |  | 3 | 5 | 1 | 7 |  | 8 |
| 1 |  | 9 | 4 | 8 |  | 5 | 53 |  | 1 |  | 6 | 4 | 7 | 8 | 5 | 2 | 23 | 4 | 8 | 2 | 3 |  |  | 5 | 9 | 7 | 5 | 7 |  | 6 | 9 |  | 8 | 1 | 4 |
| 7 | 5 | 3 | 9 |  | 2 | 1 | 8 | 84 |  |  | 5 |  | 2 | 6 | 1 | 4 |  |  | 9 | 1 |  | 4 |  | 85 | 3 |  | 4 | 2 | 8 |  | 1 |  | 9 | 6 | 7 |
|  |  | 8 | 1 | 5 | 3 |  | 9 | 97 | 2 | 8 |  |  | 3 |  | 6 | 7 |  | 5 | 3 | 7 | 2 |  |  | 61 | 8 | 4 | 6 | 1 | 9 | 8 | 4 | 7 |  |  |  |
| 6 | 8 | 2 | 7 | 1 | 5 | 3 | 34 | 49 | 5 | 4 | 3 |  | 8 |  | 7 | 1 |  |  | 7 | 9 | 4 | 3 |  | 8 | 5 | 6 | 1 | , | 7 | 2 | 3 | 4 | 5 |  |  |
| 9 | 1 |  |  | 2 |  | 8 | 5 | 56 | 9 | 7 |  |  | 4 | 2 | 8 | 6 | 65 |  | 4 | 5 | 8 |  |  | 97 | 2 |  | 8 | 6 | 2 | 1 |  | 5 | 4 | 3 | 39 |
| 5 |  | 4 | 6 | 9 | 8 | 7 | 72 | 21 | 6 | 2 | 8 | 7 | 5 |  |  |  | 4 |  |  |  |  | 7 | 7 | 13 | 4 | 9 |  | 5 |  | 9 | 6 | 8 |  | 7 | 71 |
| Puzzle13 |  |  |  |  |  |  |  |  | Puzzle14 |  |  |  |  |  |  |  |  | Puzzle15 |  |  |  |  |  |  |  |  | Puzzle16 |  |  |  |  |  |  |  |  |
|  | 6 |  |  | 3 | 1 | 7 | 8 |  | 7 |  | 8 | 4 |  | 6 | 3 |  | 1 | 3 | 2 | 21 | 9 |  |  |  | 4 | 6 |  |  | 9 | 1 |  |  |  |  | 2 |
| 2 | 1 |  | 8 |  |  |  | 5 | 54 | 2 | 1 | 3 | 7 | 8 |  | 5 | 4 | 46 |  | 9 |  | 1 | 5 |  | 3 |  |  | 3 | 1 | 2 |  |  |  | 6 |  |  |
| 7 |  |  | 2 | 4 |  |  | 3 | 39 | 9 |  |  |  | 5 |  | 8 | 2 |  | 8 | 7 | 5 | 6 |  |  | 4 | 9 | 1 | 8 |  |  |  |  |  |  | 3 | 9 |
| 5 | 3 | 7 |  |  | 2 | 8 |  | 1 | 4 |  | 2 | 5 | 1 | 3 | 7 |  |  | 1 | 6 | 69 | 7 | 4 | 4 | 82 | 25 | 3 |  |  |  | 7 |  |  | 3 | 2 |  |
| 6 | 9 | 1 | 4 | 8 | 7 | 3 |  | 5 | 6 | 9 |  | 8 |  | 7 | 2 |  | 3 | 7 | 5 | 3 | 2 | 9 |  | 64 | 1 | 8 | 7 |  |  |  | 6 | 9 |  |  | 1 |
|  |  | 2 | 1 | 5 | 3 | 9 |  | 6 |  | 3 | 7 |  | 9 | 2 | 4 |  | 8 | 2 | 4 | 48 | 3 | 1 |  | 5 | 6 | 9 | 2 | 8 |  | 3 | 5 |  | 9 |  | 6 |
| 1 |  | 9 |  | 2 | 4 | 5 | 56 | 6 |  | 2 | 6 | 1 | 7 | 5 | 9 | 3 | 34 |  | 1 |  | 4 | 3 | 3 | 6 | 68 | 5 | 5 | 4 |  | 8 |  | 6 |  |  | 7 |




|  | 8 | 8 |  |  |  | 2 |  |  | 4 | 8 |  | 1 |  |  | 9 | 2 |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |

Table 10: Puzzle set 3

| Puzzle1 |  |  |  |  |  |  |  | Puzzle2 |  |  |  |  |  |  |  | Puzzle3 |  |  |  |  |  |  |  | Puzzle4 |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 9 | 9 | 8 |  | 4 |  |  | 2 | 5 | 5 1 |  |  |  |  | 7 |  | 9 | 3 | 4 | 5 |  | 6 | 8 | 2 |  | 5 |  | 9 |  | 2 |  |  |
| 2 | 2 |  | 5 | 7 |  |  | 9 |  | 6 |  |  | 5 | 58 |  |  |  |  |  | 3 |  |  | 7 | 4 |  | 3 |  |  | 47 | 8 |  | 9 |
|  |  | 3 |  |  | 2 | 7 |  |  |  |  | 81 | 1 |  | 6 |  |  | 2 |  |  |  | 7 |  | 3 |  | 9 | 2 |  | 8 | 4 |  |  |
| 3 | 3 |  | 2 |  |  |  |  | 7 | 7 | 9 |  |  |  |  | 2 |  |  | 6 | 7 |  | 4 |  |  |  |  | 1 | 3 |  |  |  | 4 |
| 4 | 4 | 7 | 9 | 1 | 3 | 6 | 8 |  | 1 | 26 | 68 | 89 | 95 | 7 |  | 3 |  |  |  |  |  |  | 6 |  |  |  |  | 2 |  |  |  |
|  |  |  |  |  | 5 |  | 3 | 8 | 8 |  |  |  | 1 |  | 9 |  |  |  | 6 |  | 1 | 9 |  | 5 |  |  |  | 4 | 6 |  |  |
|  |  | 5 | 7 |  |  | 1 |  |  | 3 |  |  | 92 | 2 |  |  | 7 |  |  | 5 |  |  |  | 9 |  |  | 7 |  | 5 | 3 | 8 |  |
|  | 2 |  |  | 5 | 4 |  | 9 | 1 | 1 |  |  |  |  | 9 |  |  | 8 | 4 |  |  | 9 |  |  | 6 |  | 3 | 8 | 7 |  | 2 | 5 |
|  | 1 |  |  | 3 |  | 2 | 7 |  |  | 4 |  |  | 2 |  | 1 |  | 5 | 9 | 2 | 7 | 3 |  | 68 |  | 2 | 5 |  | 9 |  | 4 |  |
| Puzzle5 |  |  |  |  |  |  |  | Puzzle6 |  |  |  |  |  |  |  | Puzzle7 |  |  |  |  |  |  |  | Puzzle8 |  |  |  |  |  |  |  |
|  | 9 |  |  | 6 | 1 |  | 78 |  | 9 | 5 |  |  |  | 4 |  |  |  |  | 7 | 4 |  |  | 3 | 6 |  | 9 |  | 23 |  |  | 8 |
| 1 | 1 |  | 8 |  | 2 |  | 35 | 6 | 63 | 7 |  |  | 9 |  |  | 7 |  |  |  |  |  | 4 | 5 |  |  | 4 | 5 |  |  | 7 |  |
|  |  |  |  |  |  | 1 |  | 1 | 1 |  |  |  | 97 |  | 5 |  |  | 4 |  |  | 3 | 9 |  |  |  | 5 |  | 1 | 3 |  |  |
|  | 4 | 2 | 1 |  | 6 |  | 8 | 3 | 3 | 81 | 17 | 7 | 2 |  |  |  | 9 |  |  | 8 |  | 6 | 2 | 7 |  | 2 | 3 | 8 | 5 |  | 9 |
|  | 7 |  | 3 |  | 8 |  | 9 |  | 1 |  | 3 |  | 6 | 5 |  | 3 | 1 | 2 | 5 | 6 | 9 | 7 | 48 |  | 6 |  |  |  |  | 8 |  |
|  | 5 |  | 2 |  | 9 | 6 | 1 |  |  | 9 |  | 42 | 21 |  | 7 |  | 7 | 8 |  | 2 |  |  | 9 | 5 |  | 8 |  | 47 | 6 |  | 3 |
|  |  | 9 |  |  |  |  |  | 2 | 2 |  |  |  |  |  | 1 |  |  | 1 | 6 |  |  | 2 |  |  |  | 7 |  | 9 | 2 |  |  |
|  | 42 |  | 9 |  | 3 |  | 6 |  |  | 6 |  |  | 3 | 2 | 4 | 5 |  | 9 |  |  |  |  | 6 |  | 5 |  |  | 4 | 7 |  |  |
|  | 7 7 |  | 5 | 8 |  |  | 2 |  | 7 |  |  |  | 8 | 9 |  |  | 6 |  |  | 3 | 4 |  |  | 3 |  |  | 2 | 7 | 8 |  | 5 |
| Puzzle9 |  |  |  |  |  |  |  | Puzzle10 |  |  |  |  |  |  |  | Puzzle11 |  |  |  |  |  |  |  | Puzzle12 |  |  |  |  |  |  |  |
|  | 1 |  | 3 | 7 |  | 9 |  |  | 76 | 3 |  | 95 | 54 |  | 8 |  | 4 |  | 6 |  |  |  | 87 |  | 7 |  |  | 3 |  | 4 | 6 |
|  |  | 6 | 1 | 9 |  | 5 |  | 2 | 2 | 1 |  |  |  | 6 |  | 5 |  | 9 |  |  |  |  | 64 |  |  | 6 |  |  |  | 5 | 7 |
| 7 | 74 | 9 | 2 |  |  |  | 6 | 5 | 5 |  |  | 68 | 8 |  |  |  |  |  |  |  |  |  | 5 |  |  |  | 7 | 9 |  |  | 3 |
| 3 | 3 | 2 |  |  |  | 1 |  |  |  | 2 |  | 39 | 9 | 4 |  |  | 3 | 8 | 7 |  | 6 |  |  |  |  | 9 |  | 7 |  |  | 1 |
|  |  |  | 6 |  | 1 |  |  |  |  | 9 |  |  | 3 |  |  | 2 |  |  | 5 |  | 9 |  | 8 |  |  |  | 2 | 6 |  |  |  |
|  |  | 7 |  |  |  | 4 | 5 |  | 5 |  |  | 7 | 1 |  |  |  |  |  | 2 |  | 8 | 4 | 9 | 4 |  |  | 8 |  | 6 |  |  |
|  | 8 |  |  |  | 3 | 2 | 59 |  |  |  |  | 5 |  |  | 4 | 4 |  |  |  |  |  |  |  | 9 |  |  | 5 | 4 |  |  |  |
|  |  | 3 |  | 8 | 2 | 6 |  |  | 2 |  |  |  | 8 |  | 3 | 7 | 2 |  |  |  |  | 8 | 6 |  | 6 |  |  |  | 5 |  |  |
|  |  | 1 |  | 6 | 5 |  | 7 | 9 | 9 | 54 | 48 | 8 | 2 | 7 | 1 | 3 | 9 | 6 |  |  | 4 |  | 2 |  | 8 |  |  | 2 |  | 1 | 9 |
| Puzzle13 |  |  |  |  |  |  |  | Puzzle14 |  |  |  |  |  |  |  | Puzzle15 |  |  |  |  |  |  |  | Puzzle16 |  |  |  |  |  |  |  |
|  | 2 | 1 |  |  | 5 | 3 | 9 |  | 2 |  | 1 |  |  | 4 | 3 |  |  |  |  |  |  |  | 38 |  |  | 2 |  | 37 |  | 8 | 5 |
|  | 96 |  |  |  | 2 |  | 7 |  | 7 | 1 |  | 5 |  |  |  |  |  |  |  |  |  | 7 | 9 | 3 |  |  | 2 |  |  |  |  |
|  |  | 7 |  |  |  |  | 2 |  | 6 |  |  |  | 9 | 7 |  |  |  |  | 9 |  | 3 | 6 | 54 | 5 |  |  | 4 | 9 |  |  | 2 |
|  | 4 |  | 9 |  | 1 |  | 2 |  |  | 2 |  |  |  | 8 | 4 |  | 7 |  |  | 1 | 9 |  | 5 |  | 2 | 9 |  |  |  |  |  |
|  |  | 5 |  |  |  | 4 |  |  | 8 |  |  | 42 | 2 |  | 7 | 4 |  |  | 8 |  | 5 |  | 7 |  | 6 |  |  |  |  | 2 | 3 |
|  | 2 |  | 4 |  | 6 |  | 3 |  | 7 3 |  |  |  | 2 |  |  | 9 |  |  | 3 | 2 |  |  | 4 |  |  |  |  |  | 9 | 4 | 8 |
|  | 8 |  |  |  |  | 2 |  |  | 5 |  | 6 |  |  |  | 2 | 5 | 6 | 7 | 2 |  | 4 |  |  |  | 6 |  | 5 | 8 |  |  | 9 |




