

Review and Implementation Heuristic Search Methods for Solving Cryptoarithmatic Problems

By
Ashish D. Fugare (170101023)

Under Guidance Prof. Pinaki Mitra



Certificate

This is to certify that the work contained in this thesis entitled "**Review and Implementation Heuristic Search Methods for Solving Cryptoarithmic Problems**" is a bonafide work of Ashish D. Fugare (Roll No. 170101023), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati ,under my supervision and that it has not been submitted elsewhere for a degree.

Supervisor: Prof. Pinaki Mitra
Associate Professor
Department of CSE
IITG

Abstract

This thesis explores heuristic search methods for solving Cryptarithmic problems, focusing on the implementation and evaluation of Constraint Satisfaction Problem (CSP) techniques. Cryptarithmic problems are mathematical puzzles where digits are represented by letters of the alphabet, posing a significant challenge for computational algorithms. Traditional brute-force methods quickly become unmanageable as the problem size increases, leading to inefficiencies in time and space complexity. Our study leverages the inherent constraints within these puzzles to reduce the search space and enhance computational efficiency. We review existing literature on heuristic search methods and implement various algorithms to analyze their performance. Our experiments demonstrate that the heuristic CSP method significantly outperforms the brute-force approach, yielding faster and more effective solutions. This research aims to contribute to the field by providing a more efficient approach to solving Cryptarithmic problems, especially as the size and complexity of the problems increase.

Acknowledge

We acknowledge the kind guidance provided to us by Prof. Pinaki Mitra.

1.Introduction

Cryptoarithmic problems are a type of mathematical puzzle where digits are represented by letters of the alphabets. These problems have intrigued both amateurs of all ages and professionals in the field such as mathematicians. These puzzles are an engaging mental exercise but also represent a significant challenge for solving these by computational algorithms,. The aim of this thesis is to explore heuristic search methods for solving cryptoarithmic problems, with a focus on constraint propagation techniques. By leveraging the inherent constraints within these puzzles, we can significantly reduce the search space, leading to a more efficient solution with better time and space .We will also implemnt this technqiues. This study will review existing literature on heuristic search methods and implement various algorithms and analyze their performance from a baseline. This research aims to contribute to the field by providing in solving these intricate puzzles through heuristic approaches.

1.1.Cryptarithmic Puzzle

Cryptoarithmatics is a genre of mathematical puzzle in which the digits are replaced by symbols or by letters of the alphabet or etc. The aim of solving these puzzles is to guess the unique numbers representing aplahabets in the puzzle conting simple arithmetic operators such as +,*,-/.

The puzzle shown in figure 1.1 is a Classic example in this field. Cryptarithmic has already become a standard AI problem because it characterizes numbers of important problems in the computer science field.

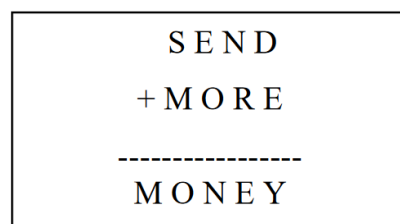

$$\begin{array}{r} \text{S E N D} \\ + \text{M O R E} \\ \hline \text{M O N E Y} \end{array}$$

Figure 1.1

1.2.Constraint Satisfaction

Constraint Satisfaction propagation is a very powerful technique that is often used in solving constraint satisfaction problems. One example of which is cryptarithmic puzzles..CSP is a search procedure that operates in a space of constraint sets.The initial state contains the constraints that originally given in the problem description.A goal state refers to a state that has undergone sufficient constraint, where enough conditions have been established to define the solution for each specific problem.Like in example of Cryptarithmic puzzle each letter must be uniquely assigned.Constraint satisfaction is a 2 step process .First constraints are discovered and are propagated as far as possible throughout the system. Search begins if there is still not a solution. Afterwards to progress a guess about a letter and added as a new constraint. By systematically applying constraints across the search space, the method can eliminate large sections of the search tree, reducing the overall search effort and leading to faster and more effective solutions.

This process can significantly reduce the complexity of the problem by eliminating inconsistent values early on.

3.Review of Prior Works

Researchers have increasingly used Constraint Satisfaction Problem (CSP) techniques to solve cryptarithmic puzzles, which are mathematical problems where each letter in an equation represents a unique digit. Earlier approaches often relied on brute-force methods, where every possible digit combination was tested, but this quickly became unmanageable as the number of letters grew leading to increases in time and space complexity. Unlike brute-force approaches that try every possible digit assignment which becomes computationally intensive, heuristic search methods incorporate intelligent strategies to prioritize certain paths over others, aiming to reduce the number of guesses needed. For cryptarithmic puzzles, researchers have also experimented for cryptarithmic problems to optimize with heuristics like **Minimum Remaining Values (MRV)**, which selects variables with the fewest possible digit options first, and **Least Constraining Value (LCV)**, which chooses values that impose the fewest restrictions on other variables. Some papers such as [SolFeng] which, have instead used a Logical Programming approach to solve these types of problems. Others are using Heuristic algorithms like Genetic Algorithm [SolvPara] to solve this problem, which also gives very efficient results

4. Proposed Solution

Constraint Satisfaction Propagation

In this part we are looking at the Psudocode of CSP that we will be implementing we are also going to its step using th example of Puzzle from Figure 1.1

Algorithm for CSP

Alorithm for CSP

Psudocode for CSP Algo

Algorithm Constraint_Satisfaction

Initialize the set of all objects that must have values assigned in a complete solution

OPEN = {set of all objects}

Loop until an inconsistency is detected or OPEN is empty

while OPEN is not empty:

Select an object OB from OPEN

 OB = select an object from OPEN

Strengthen the set of constraints that apply to OB as much as possible

strengthen_constraints(OB)

Check if the current set of constraints is different from the last time OB was examined

if constraints_set(OB) has changed since last examination or this is the first time OB is examined:

Add to OPEN all objects that share any constraints with OB

for each object X that shares constraints with OB:

 add X to OPEN

Remove OB from OPEN

remove OB from OPEN

To apply in general solution, the algorithm is stated in general possible term. To apply it in the problem domain requires 2 kinds of rules: rules that define the constraints may validly be propagated and rules that suggest guesses when necessary. In general more powerful the rules for propagating constraints, the less need there is for guessing,

We can explain this using the example from figure 1.1 Assuming C1 ,C2,C2,C4 be, respectively first,second, third and fourth carry bit. Intially rules for propagating constraints generate the following additional constraints:

- $M=1$, since 2 single-digit no. plus a carry cannot exceed more than 19
- $S=8$ or 0 since $S+M+C3 > 9$ and $M=1, S+1+C3 > 9$ or $S+C > 8$ and $C3$ is at most 1
- $O=0$ since $S+M(1) + C3 \leq 1$ must be atleast 10 to generate a carry and it can at most be 11. But M is already 1 that implies O must be 0.
- $N=E$ or $E+1$ depending on value of $C2$. But N can't be having same value as E so $N=E+1$ and $C2 = 1$
- In order for $C2$ be 1 sum of $N+R+C1$ must be > 9 , so $N+R$ must be > 8 .
- $N+R$ can't be greater than 18 even with carry so E can't exceed 9

At this point let us assume no more constraints can be generated. Then to make progress here we must guess. Suppose E is guessed as value 2. Now the next cycle begins

Now the constraint propagator are observed as

- $N=3$ since $N = E+1$
- $R=8$ or 9 and $R+N(3) + C1 (1 \text{ or } 0) = 2$ or 12 . But from previous step N is already 3, the sum of these non-negative numbers can't be less than 3. thus $R+3+ (0 \text{ or } 1) = 12$ and $R=8$ or 9
- This Means $2+ D = Y$ or $2+ D = 10+Y$

Now we need another guess suppose $C1$. If we try the value 1 then we eventually reach a dead end, when this happens, we backtrack and try $C1=0$

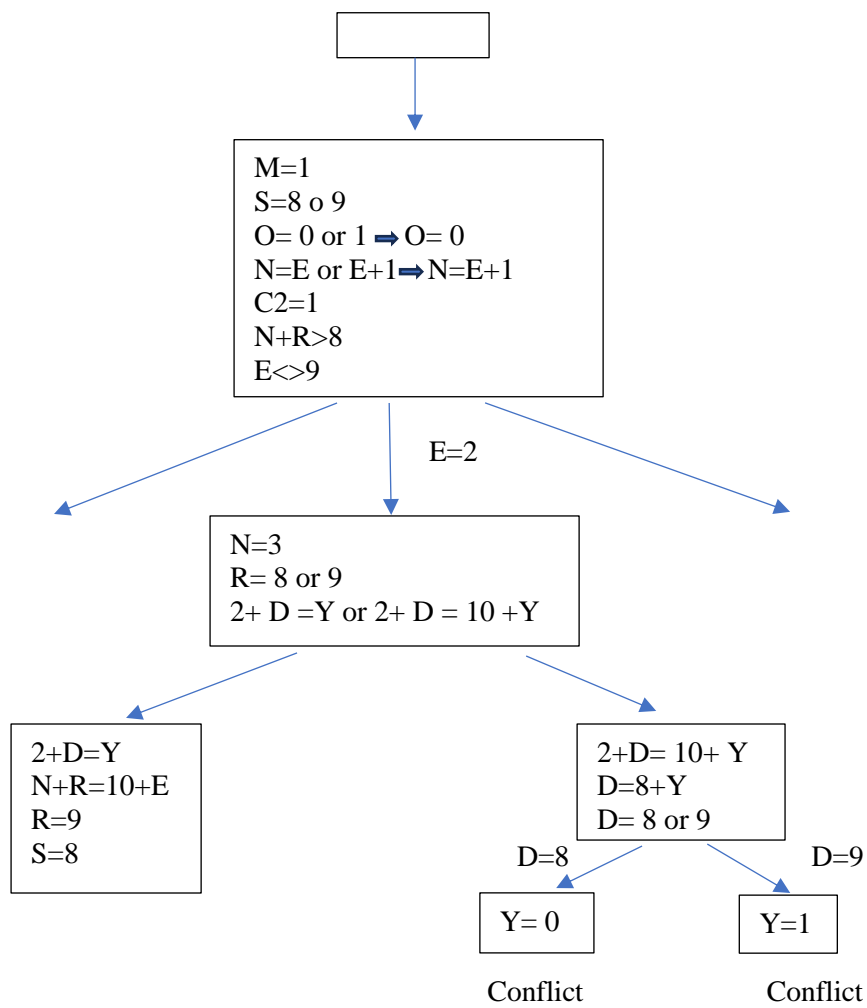


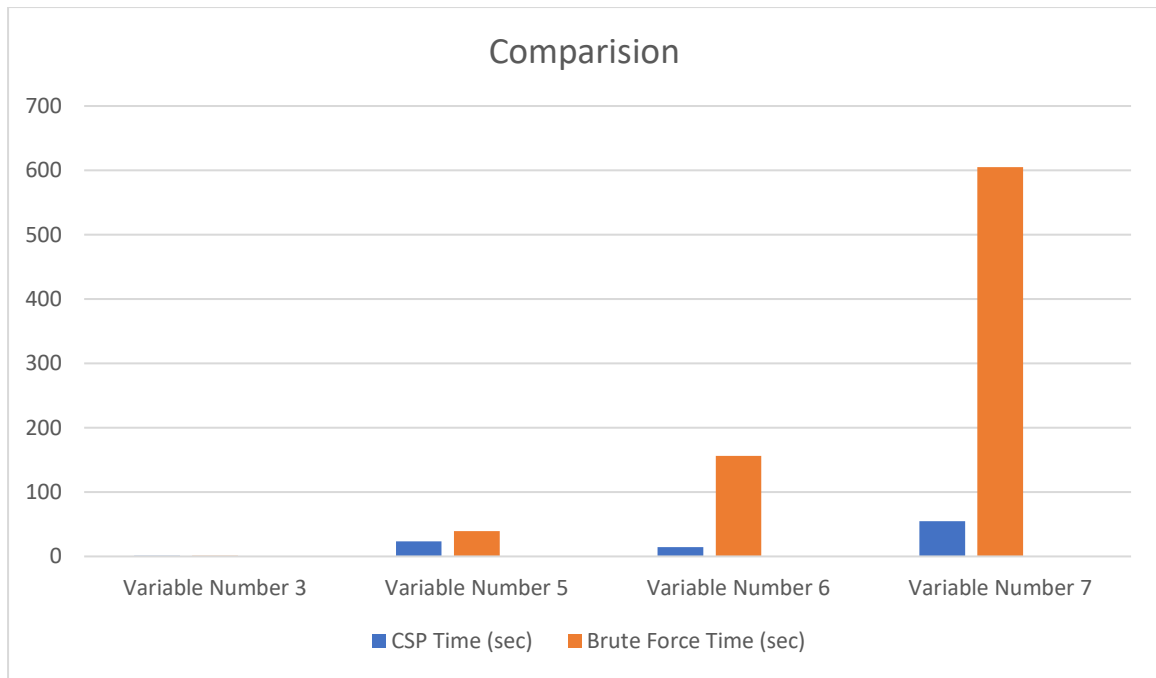
Figure 1.2 Solving a Cryptarithmic Problem

5.Experiments

In this section, we evaluate the performance of our implemented code using different numbers of alphabets, utilizing both the Constraint Satisfaction Problem (CSP) and brute force methods for comparison. The algorithm was implemented in Python 3 and applied to various Cryptarithmic problems with differing numbers of symbols and numbers. The performance was tested on a machine with a 13th Gen Intel(R) Core(TM) i5-13500H 2.60 GHz processor, 16 GB RAM, and Windows 11 OS. Our results indicate that the heuristic method of CSP yields significantly faster results compared to the brute force method, due to its ability to efficiently prune the search space and reduce the number of potential solutions that need to be evaluated. The comparison of performance metrics shows that the CSP method consistently outperforms the brute force approach, especially as the size of the problem increases. This efficiency makes CSP the preferred method for solving larger and more complex Cryptarithmic problems.

Variable Number	3	5	6	7
CSP Time	1.26	23.6553	14.4586	54.556
Brute Force Time	1.27	39.1570	156.842	604.8

Table 1.1



Graph 1.1 By the comparison shown above we can see we have a efficient method with CSP rather than Brute force method.

So we see that Constraints Satisfaction Propagation is indeed efficient to the Brute Force method of using permutation for solution

Number of Variables	Time For Execution(sec)
3	1.25844
5	23.6353
6	14.4586
7	54.5567

Table 1.2

The above table also shows the average executing time for solving the problem with different number of variables in the puzzle.

6.Conclusion

Our experiments demonstrate that the heuristic method of Constraint Satisfaction Problem (CSP) significantly outperforms the brute force method for solving Cryptarithmic problems. The efficiency of the CSP approach is evident in its ability to quickly and effectively prune the search space, thereby reducing the computational time and resources required. This advantage becomes increasingly pronounced as the problem size grows, making CSP the preferred method for larger and more complex scenarios. Overall, the heuristic method of CSP proves to be a more efficient and scalable solution compared to the traditional brute force approach.

7.Future Work

In future work, the application of Constraint Satisfaction Problems (CSP) for solving cryptarithmic puzzles can be expanded in several ways to improve performance and broaden the scope of problems that can be addressed.

1. **Enhanced Heuristics and Optimization Techniques:** Developing more advanced heuristics, such as dynamic variable ordering and adaptive constraint propagation, could reduce the search space even further and lead to faster solution times. Techniques like forward checking or constraint propagation based on arc consistency can also be refined for specific characteristics of cryptarithmic puzzles.
2. **Parallel and Distributed Computing Approaches:** Implementing parallel or distributed versions of CSP-based solvers could allow for handling puzzles with a greater number of variables or more complex constraints. This would involve distributing constraint propagation or search tasks across multiple processors, potentially speeding up computation significantly.
3. We can also experiment with mix of other Heuristic Techniques such as Greedy Method, Genetic Algorithm ,etc

Refrence

- [elaineAI] Artificial Intelligence Book by Elaine Rich and Kevin Knight Chapter 3
- [SolFeng] Solving Cryptarithmic Puzzles by Logic Programming Feng-Jen Yang
- [SolvPara] Solving Cryptarithmic Problems Using Parallel Genetic Algorithm
Reza Abbasian
- [BonO] Bonnie Averbach and Orin Chein, Problem Solving Through Recreational Mathematics,
Courier Dover Publications, 2000, pp. 156.
- [Duen68] H. E. Dudeney, in Strand Magazine vol. 68 (July 1924), pp. 97 and 214.
- [DavGen] David Goldberg, "Genetic Algorithms in Search, Optimization and "Machine
Learning," Addison-Wesley, Reading, MA 1989.
- [5PoliPa] Mariusz Nowostawski, Riccardo Poli, "Parallel Genetic Algorithm Taxonomy",
KES'99
- [Konf18] Konfrst, Z, "Parallel Genetic Algorithms: Advances, Computing
Trends, Applications and Perspectives," Parallel and Distributed
Processing Symposium, 2004. Proceedings. 18th International
26-30 April 2004 Page(s):162
- [Bartko4] I. Bratko, "Prolog programming for artificial intelligence," 4th Ed.,
Addison Wesley, 2012.
- [Yang18] F. Yang, "Solving the Water Jug Puzzle in CLIPS," Lecture Notes in
Engineering and Computer Science: Proceedings of The World Congress
on Engineering and Computer Science 2018, San Francisco, CA, pp.
564-567, 2018.
- [Yang20] F. Yang, "A Logic Programming Solution to the Water Jug Puzzle,"
Transactions on Engineering Technologies: World Congress on Engineering and
Computer Science 2018, pp. 66-73, 2020.

