Hello everyone I am Ashish Dileep Fuagare Roll No (170101023) and I am presenting a BTP under

Professor Pinaki Sir titled “Review and Implementing of Heuristic Search Techniques for solving crypto arithmetic problems”

For implantation I will focus on solving the cryptoarithamtic problem using Constrain Satisfaction Problem

Cryptarithmetic is a puzzle consisting of an arithmetic problem in which the digits have been replaced by letters of the alphabet.The goal is to descifer the number given form the contraints of puzzles with an additional constraint that each mapping to a alphabet and digit should be unquie and the leading alphabet can’t be zero.

. This type of problem was popularized during the 1930s the Sphinx, a Belgian journal of recreational mathematics.

.

There are Wide Range of Applications for Cryptarithm problems such as applications in cryptography, error detection and correction, and even AI and machine learning.

Solutions derived from this can have real-world implications in improving data security, optimization problems in logistics, and even in developing new algorithms for artificial intelligence. We are goanna try to solve cryptarithmetic using the Constraint Satisfaction Problems (CPSs), in which we try to find a solution from a domain given some problem constraints.

"In Constraint Satisfaction Problems (CSPs), the process begins by identifying and propagating constraints throughout the system to reduce the search space. If no solution is found after this propagation, the search proceeds by making an informed guess—assigning a value to a variable, such as a letter in the puzzle, and treating it as a new constraint. This systematic approach helps to eliminate large sections of the search tree, ensuring that only feasible solutions are explored. By combining constraint propagation with strategic guessing, CSP methods significantly reduce computational effort, making the process faster and more efficient."

Follwing is the description of heurstics used for optimization heuristic

* minimal residual variable which is MRV selects the variable with fewest legal values possible values for assignment. This reduces the chances of wasting time exploring invalid assignments..

The next herustic for optimization is forward checking where we proactively eliminates values from the domains of unassigned variables that conflict with the current assignment. It prevents future conflicts early, reducing the chances of exploring invalid assignments.

* **Example:** For **SEND + MORE == MONEY**, if S=9, forward checking will remove 9 from M’s domain if S and M share constraints.

"AC-3 (Arc Consistency Algorithm 3) is a powerful optimization technique used in CSP to reduce the problem size before solving. It works by enforcing arc consistency, ensuring that every value in a variable's domain satisfies the binary constraints with all connected variables. For example, in the cryptoarithmetic puzzle SEND + MORE == MONEY, if the variable 'S' must be a digit {9, 8, 7} and it cannot be 0 due to the leading zero constraint, AC-3 will prune 0 from 'S's domain. This pruning process is not limited to 'S'; it propagates across all related variables, updating their domains to reflect these constraints. By iteratively applying this process, AC-3 eliminates inconsistent values early, significantly reducing the search space and the number of potential solutions that need to be explored during the actual solving phase.

In the context of solving cryptoarithmetic problems, AC-3 proves invaluable because it preemptively resolves conflicts that could arise during backtracking. This reduces the number of backtracking steps required, leading to fewer nodes explored and faster execution times. However, the computational overhead of applying AC-3, particularly for simpler puzzles or small domains, might not always yield proportional benefits.

I am choosing following example to see the performance trends to see.

The shown are few examples of the Cryptoarthimatic problem. For a simple example with a Low Complexity I am using example of BASE + BALL == GAMES which has 7 unique letters and a medium complexity.

For a more moderate example I have taken SEND + MORE == Money which has 8 unique Alphabets and has higher complexity than previous example. Here we can see the performance for a algorithm for a slightly larger puzzle which has leading zero constraint.

For the edge case of No solution I have taken the example of TWELEVE + THREE == FIFTEEN ,this problem has 11 unique letters but we only have 10 numerical digits so by pigeon hole principle we see this has no solution

To test for handling larger search spaces and complex constraints I have taken the example og   
CROSS + ROADS == DANGER which has 9 unique letters and a higher complexity.

* Lastly to see a very high complexity for proble wwe see the example of DONALD + GERALD == ROBERT which which has 10 unquie alphabets and very high complexit.

In implanting Backtracking we systematically explores all possible digit assignments for letters in a depth-first manner. If a conflict arises that is a constraint is violated, it backtracks and tries a different assignment.

In the methods we see there is baseline for backtracking method and Vanilla CSP which is a implantation of CSP without any optimizations like forward checking or MRV.Then we see the CSP with basic optimization such as MRV ,fowrad checking and then we see for the Optimization with arc consistency called AC-3

"Next, we can examine the performance of the four methods through a comparison of execution time and the number of nodes explored. In the table, we present the results for each approach, including Backtracking, Vanilla CSP, CSP with Forward Checking, and CSP with AC-3. These metrics help us understand how efficiently each method solves the cryptoarithmetic puzzles.

Looking at the performance trends, we can observe a clear correlation: optimizations, to a significant reduction in the number of nodes explored, which directly translates to a reduction in execution time. However, it’s important to note that the overall time taken also depends on the complexity of the puzzle. As the number of unique letters increases and the puzzle complexity grows, the performance differences between the methods become more apparent.

??

For simpler puzzles, such as 'BASE + BALL == GAMES,' the impact of optimization techniques is less significant, but as the puzzles become more complex, like 'CROSS + ROADS == DANGER,' the advantage of using CSP with AC-3 becomes much more pronounced. The reduction in nodes explored and the corresponding decrease in execution time highlight how critical optimizations are when dealing with larger and more challenging cryptoarithmetic problems."

??

"To summarize, the various heuristic search methods applied to solving cryptoarithmetic problems — including backtracking, constraint satisfaction methods with forward checking, CSP without AC-3 optimization, and CSP with AC-3 — demonstrated distinct performance patterns across different problem complexities. Backtracking, which explored the entire solution space, was inefficient, particularly for larger and more complex puzzles like CROSS + ROADS == DANGER and TWELVE + THREE == FIFTEEN. It required a significantly higher number of nodes explored and more computation time due to the lack of optimizations.

Vanilla CSP, or constraint satisfaction problems without any optimization, provided some improvements over backtracking but struggled with scalability, especially when the number of variables and constraints increased, such as in the SEND + MORE == MONEY example.

The CSP approach with forward checking offered a moderate improvement in reducing the search space compared to backtracking, allowing the algorithm to handle larger puzzles more effectively. However, it was not as efficient as CSP with AC-3.

CSP with AC-3, on the other hand, demonstrated the most significant improvements across the board. It consistently reduced the number of nodes explored and decreased computation time for solving even complex puzzles like CROSS + ROADS == DANGER. The AC-3 algorithm's ability to propagate constraints early in the process allowed it to eliminate inconsistent values and prune large sections of the search tree, leading to faster and more effective solutions.

These findings highlight the clear advantages of AC-3 optimization in solving complex cryptoarithmetic puzzles. AC-3 can reduce computational overhead and improve overall performance, especially for problems with numerous variables and constraints. However, for simpler puzzles like BASE + BALL == GAMES, the performance improvements might not justify the computational cost of applying AC-3.

In conclusion, AC-3 is a powerful optimization technique for solving cryptoarithmetic puzzles. It should be considered a priority for large, complex problems to maximize performance and minimize computation time. However, the choice of optimization must be balanced with the problem's complexity, as simpler puzzles might not require such a high computational effort.

??

. Future research could explore more advanced CSP optimizations, such as constraint propagation techniques beyond AC-3, and hybrid approaches that different search technique to get a better result

.

."

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |