Review and Implementation Heuristic Search Methods for Solving Cryptarithmetic Problems

By

Ashish D. Fugare (170101023)

Under Guidance Prof. Pinaki Mitra

What are Cryptoarithmetic Problems?

- Definition: Mathematical puzzles where letters represent digits.

- Example:

```
    SEND             BASE
+  MORE            + BALL
----------------  -----------------
 MONEY.           GAMES
```

- Constraints:
  - Unique digits.
  - Valid arithmetic equation.
  - No leading zeros.

CSP

- **How It Works:**
- **Goal**: Solve the CSP using recursive backtracking without any preprocessing for vanilla
- **Steps**:
  1. **Start with an empty assignment** and recursively assign values to variables.
  2. **Check consistency**:
     - Ensure the chosen value satisfies all constraints with the current partial assignment.
  3. **Backtrack** if the current value leads to inconsistency or no solution:
     - Remove the value and try the next one in the domain.
  4. **Stop when all variables are assigned** consistently, or return failure if no values work.
- **Example:** For **SEND + MORE == MONEY**:
- Start with assignment and checking if the partial equation works.
- If a choice leads to inconsistency, backtrack and try a different value.

Key Features: Simpler but less efficient, as it does not reduce the search space beforehand. Relies heavily on exploring the full search tree which can be computationally expensive

Heuristic Used:

Forward Checking

How It Works:

- **Goal**: Prevent conflicts by immediately pruning inconsistent values after each assignment.
- **Steps**:
  1. After assigning a value to a variable, update the domains of all unassigned variables connected by constraints.
  2. Remove values from domains of neighbors that would lead to inconsistency.
  3. Stop and backtrack if any neighbor's domain becomes empty.
  4. Continue assigning values until all variables are assigned or failure occurs.
- **Example:** For **SEND + MORE == MONEY**, if S=9, forward checking will remove 9 from M’s domain if S and M share constraints.
- **Key Features:**
- Improves efficiency by pruning invalid values early.
- Prevents assigning values that will fail later in the search.

Minimum Remaining Values (MRV)

- **How It Works:**
- **Goal**: Select the most "constrained" variable first to minimize branching and increase chances of finding a solution.
- **Steps**:
  1. Identify variables not yet assigned.
  2. Choose the variable with the smallest domain size (least remaining values).
  3. If there’s a tie, use a secondary heuristic (like the degree heuristic) to decide.
- **Example:** If S,M,O,R,E,Y are unassigned and S’s domain has 3 values while others have 5 values to choose from, choose S first.
- **Key Features:**
- Reduces the size of the search tree by focusing on the hardest decisions early

AC-3 (Arc Consistency Algorithm 3)

- "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.

BENCHMARKS AND COMPLEXITY

- Cryptoarithmetic puzzles used:
  - BASE + BALL = GAMES (simple)
    - 7 Unique Letters
  - SEND + MORE = MONEY (moderate).
    - 8 Unique Letters
  - CROSS + ROADS = DANGER (complex).
    - 9 Unique Letters
  - DONALD + GERALD = ROBERT (very complex)
    - 10 Unique Letters
  - TWELVE + THREE = FIFTEEN (unsolvable)
    - 11 Unique Letters
- Categorize puzzles based on complexity and constraints.

Overview of Algorithms

- Algorithms:

  - Backtracking (Baseline).
  - CSP without optimizations (Vanilla).
  - CSP + Forward Checking and MRV.
  - CSP + AC-3

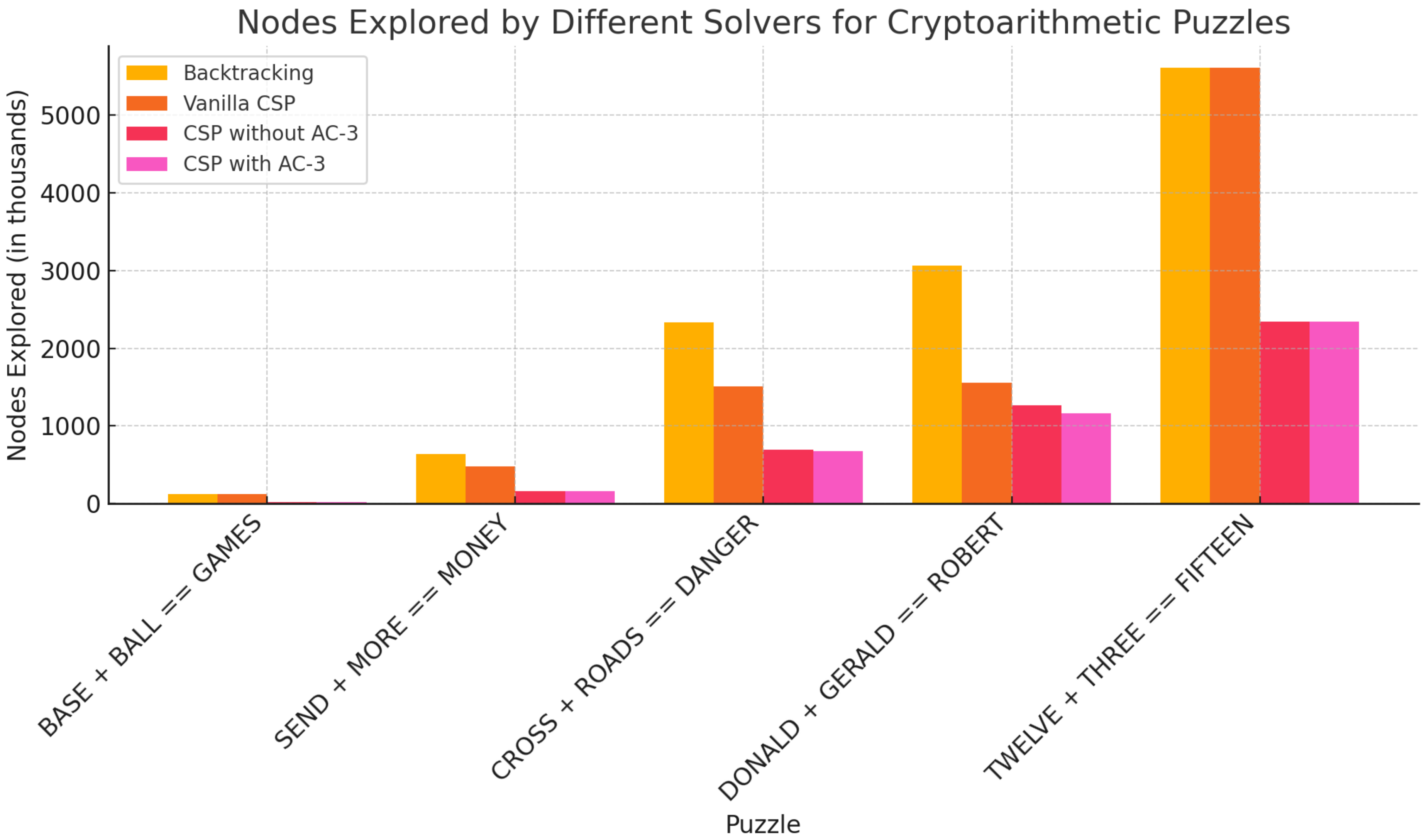
- Purpose: Compare their performance for solving cryptoarithmetic puzzles.

Results

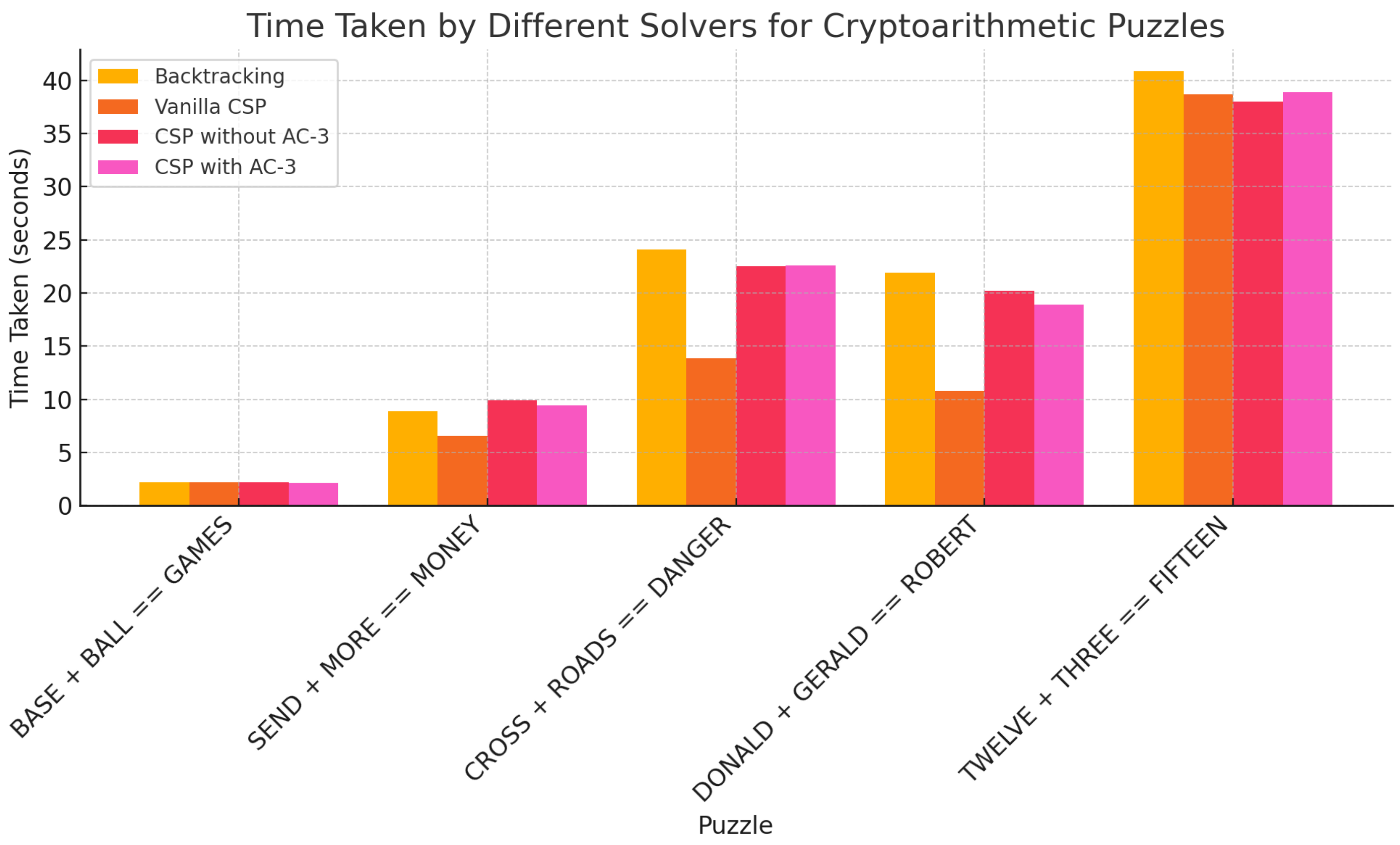
	Method	Nodes Explored	Time (sec)
BASE + BALL == GAMES	Backtrack	128,502	2.24
	Vanilla CSP	124,757	2.17
	CSP without AC3	24,708	2.19
	CSP with AC3	23,834	2.13
SEND + MORE = MONEY	Backtrack	639,364	8.89
	Vanilla CSP	475,721	6.58
	CSP without AC3	161,752	9.91
	CSP with AC3	166,229	9.45

Results

	Method	Nodes Explored	Time(sec)
CROSS + ROADS = DANGER	Backtrack	2,328,761	24.10
	Vanilla CSP	1,510,869	13.86
	CSP without AC3	697,255	22.52
	CSP with AC3	676,248	22.58
DONALD + GERALD = ROBERT	Backtrack	3,060,002	21.88
	Vanilla CSP	1,553,459	10.81
	CSP without AC3	1,270,018	20.23
	CSP with AC3	1,166,562	18.94



Nodes Explored by Different Solvers for Cryptoarithmetic Puzzles
Backtracking
Vanilla CSP
CSP without AC-3
CSP with AC-3
Nodes Explored (in thousands)
0
1000
2000
3000
4000
5000
BASE + BALL == GAMES
SEND + MORE == MONEY
CROSS + ROADS == DANGER
DONALD + GERALD == ROBERT
TWELVE + THREE == FIFTEEN
Puzzle


Time Taken by Different Solvers for Cryptoarithmetic Puzzles
Backtracking
Vanilla CSP
CSP without AC-3
CSP with AC-3
Time Taken (seconds)
0
5
10
15
20
25
30
35
40
BASE + BALL == GAMES
SEND + MORE == MONEY
CROSS + ROADS == DANGER
DONALD + GERALD == ROBERT
TWELVE + THREE == FIFTEEN
Puzzle

Difference of Performance

Nodes Explored:

- **Backtracking**: This method consistently explores the highest number of nodes across all puzzles. It is the least efficient in terms of node exploration, especially for harder puzzles like "TWELVE + THREE == FIFTEEN."
- **Vanilla CSP**: This method, which doesn't include optimizations like AC-3 or forward checking, also explores a high number of nodes, though slightly fewer than backtracking.
- **CSP without AC-3**: This method demonstrates a significant reduction in the number of nodes explored compared to the vanilla CSP, thanks to optimizations like forward checking.
- **CSP with AC-3**: This is the most optimized approach, and for most puzzles, it explores the fewest nodes. The AC-3 optimization appears to effectively prune the search space.

Time Taken:

- **Backtracking** generally takes the longest time, especially for harder puzzles like "CROSS + ROADS == DANGER" and "DONALD + GERALD == ROBERT," as it explores more nodes.
- **Vanilla CSP** reduces the time taken in comparison to Backtracking but still does not perform as efficiently as the optimized CSP methods.
- **CSP without AC-3** takes less time than Vanilla CSP due to the optimizations in variable ordering and constraint propagation.
- **CSP with AC-3** takes the least time across the board, due to the powerful constraint propagation of AC-3 that reduces the search space early in the process.

Insights:

- **AC-3 optimization** shows clear advantages in both node exploration and time taken. When solving more complex puzzles, the reduction in nodes explored and time taken is more noticeable.

- The simpler methods like **Backtracking** and **Vanilla CSP** struggle with larger and more complex puzzles, exploring many nodes and taking significantly more time.

- **CSP with AC-3** is particularly beneficial for problems with a large search space, making it an essential optimization for efficiently solving cryptoarithmetic puzzles.

Summary Result

- "Comparing the results from both methods, we can conclude:""Backtracking alone is faster for smaller problems.“
- "AC-3 optimization helps in solving complex puzzles but can introduce overhead for larger instances."

Conclusion:

- The results demonstrate that while backtracking is simple and guarantees a solution,it is computationally expensive and inefficient for larger problems.

- CSP with optimizations such as forward checking and AC-3 significantly outperforms backtracking,especially for puzzles with higher complexity.

- The AC-3 optimization, in particular, provides the best balance of efficiency in terms of both node exploration and execution time.

Future Work:

- 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.

- Thank You