Magic Hexagon

Group 17

Christoph Bartmann

The Problem

Consecutive Integers

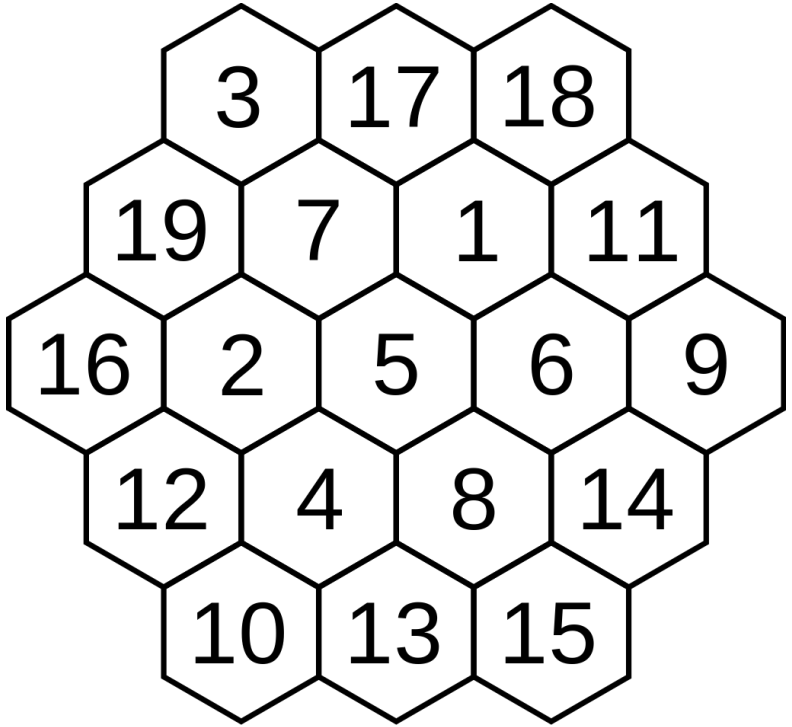
Arranged in a Hexagon

Each row and diagonals add up to the same constant

Given Program

Major functions:

- labeling: explores the search tree
- solve: checks constraints and updates bounds



n=3, M=38

Constraint Check Restart

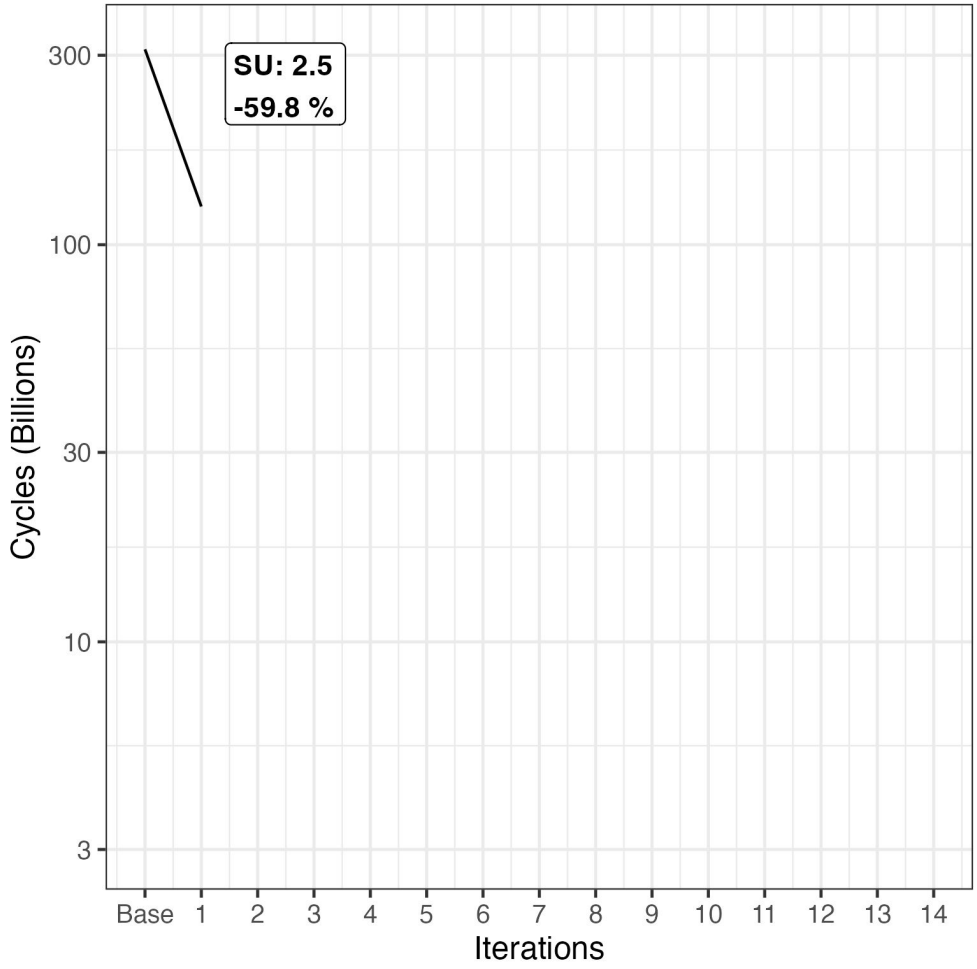
Check all constraints in solve() before restart

if change_counter > 0 and solution still

feasible restart

Don’t restart solutions that are already

impossible in not checked constraints



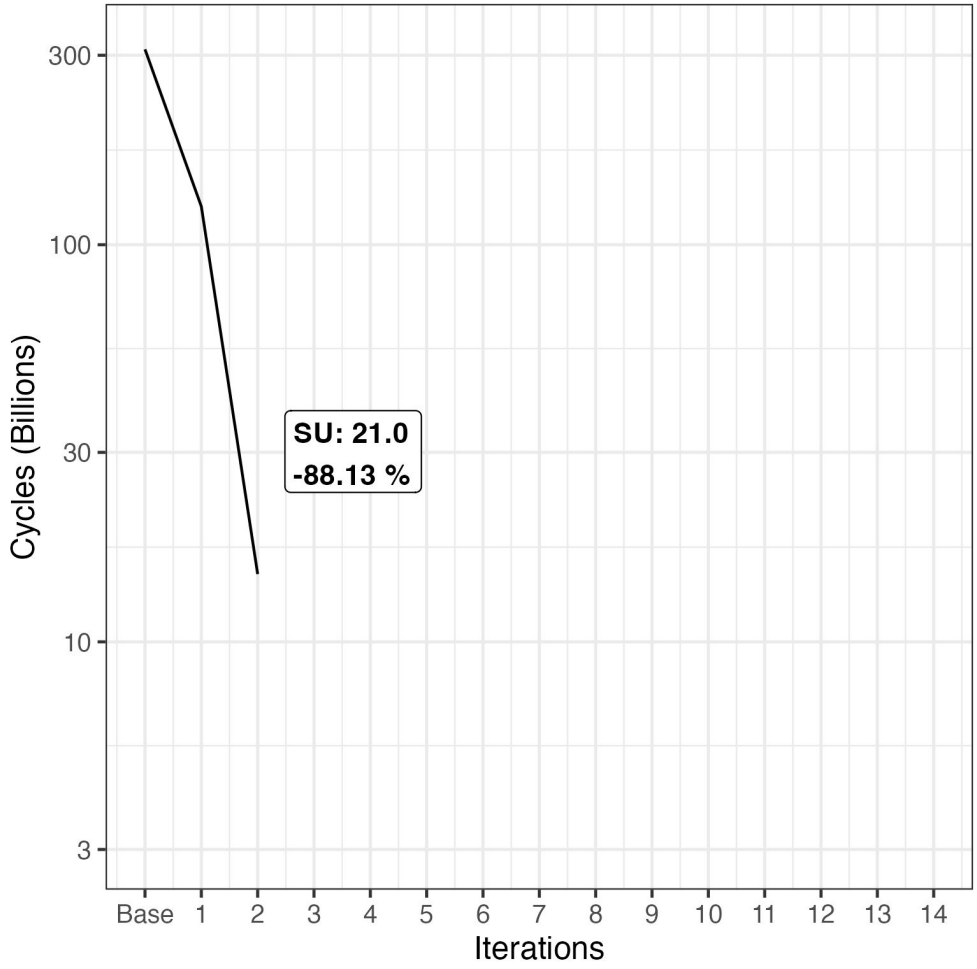
Spirale Labeling

Static index array as a spirale path in labeling()

Outside to inside performed best

Corners first, outer edges first, Center after outer edges all worse

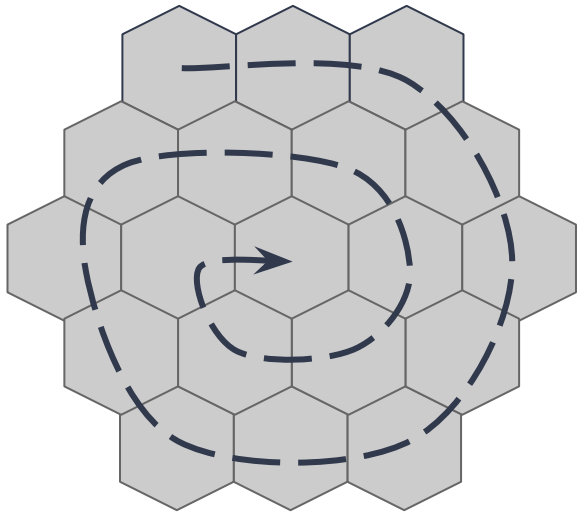
Inside to outside spirale very bad



Spirale Labeling

Description	CPU Time [s]	Cycles (billions)
Corners First	49.1	230.1
Corners First, Outer Edges	40.6	190.5
Spiral outer Edges, Middle, Spiral Inner Edges	8.3	38.9
Spiral	6.1	28.4
Reverse Spiral	timeout	

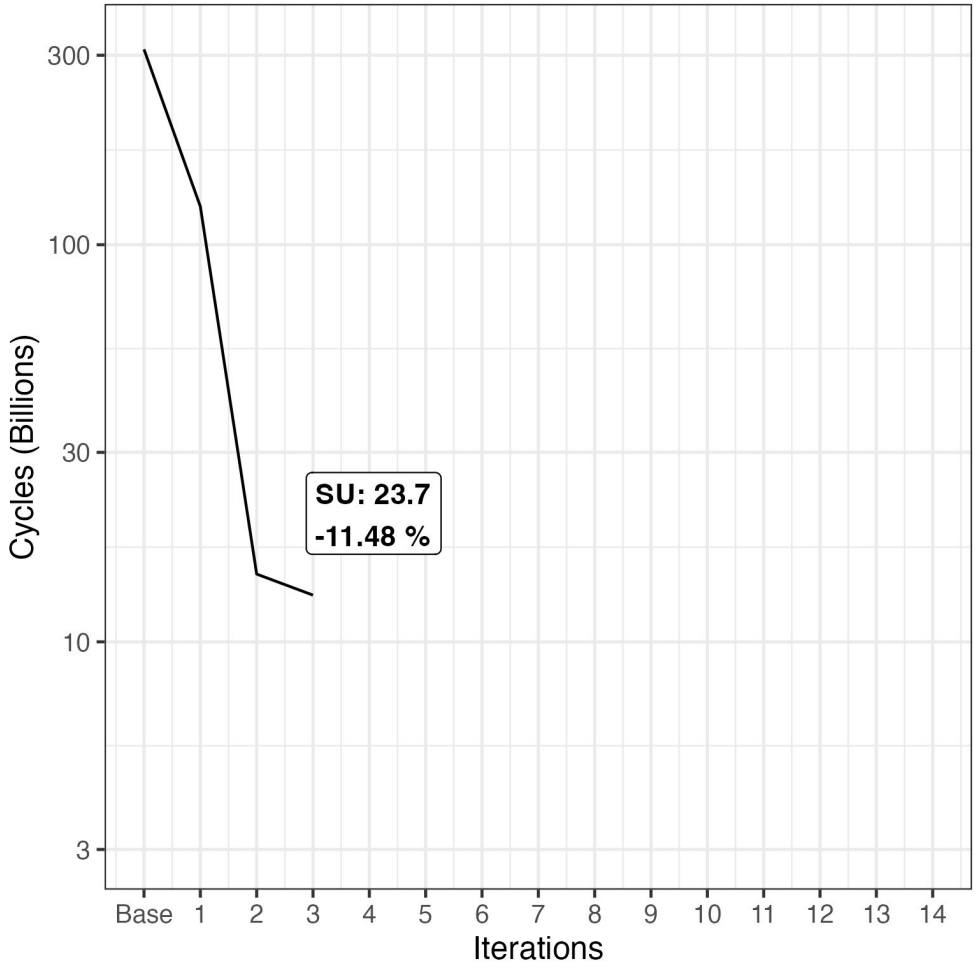
* Metrics without any other optimization



Ignoring Assertions, Occupation Pointers

Commented out assertions, in total 16

Pointers instead of indices for accessing occupation array in solve()

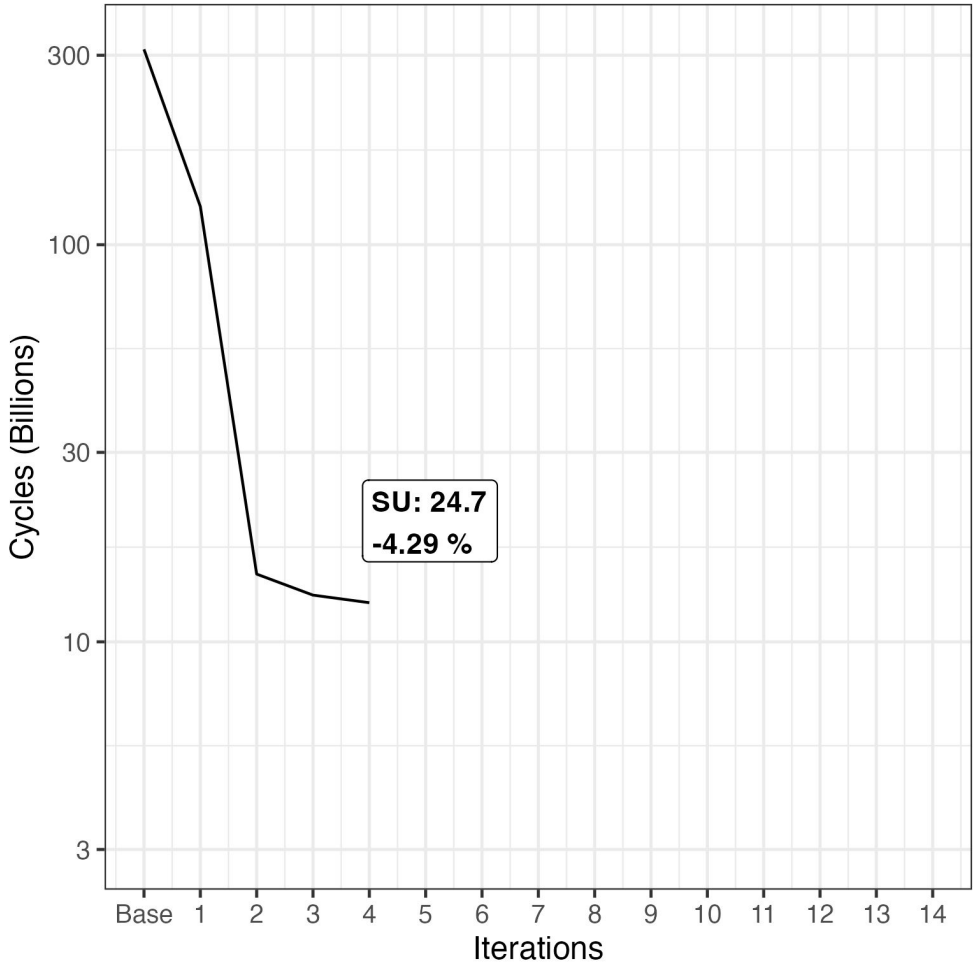


(1/2) Order in Sum Constraint Evaluation

Index array for evaluation of sum constraint

Sum checks now begin from middle and move alternatively to the outside

Earlier failure as middle has more hexagon pieces and more constraints



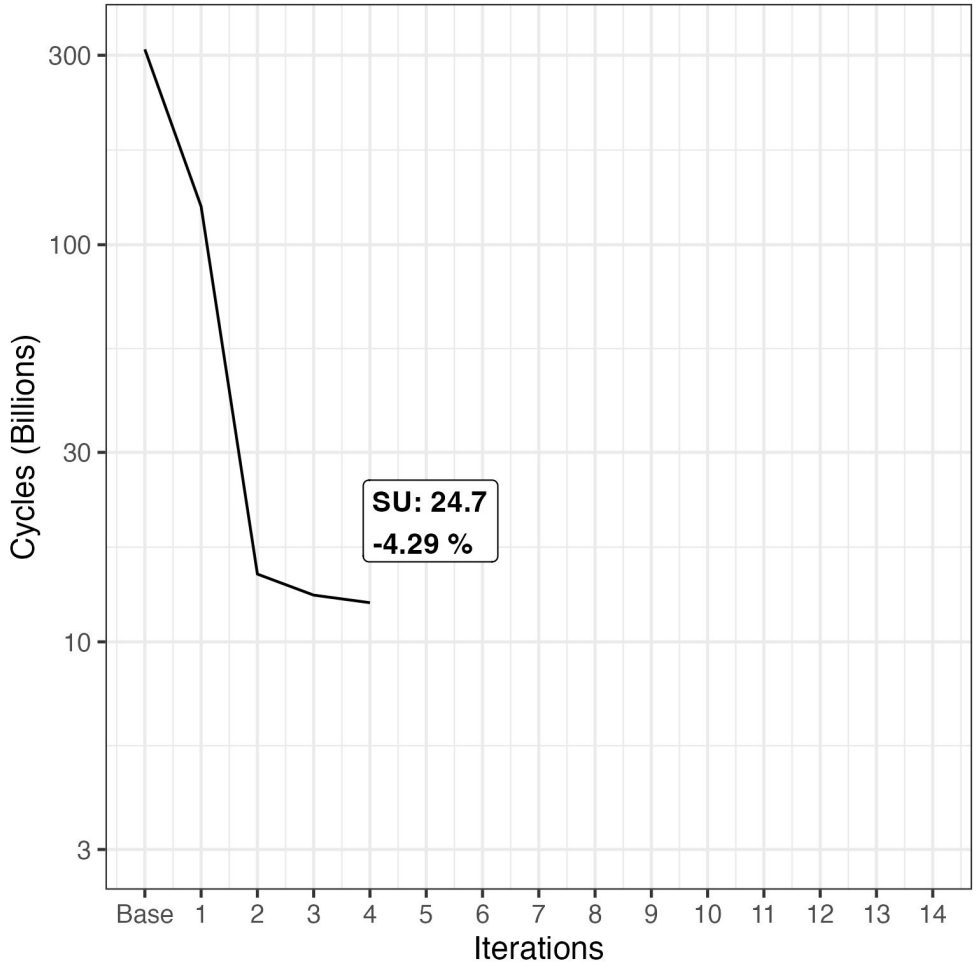
(2/2) Optimized sethi/setlo

Change logic evaluation: Check if x >= v->hi first

Most common case, nothing changes

Skips rest of function

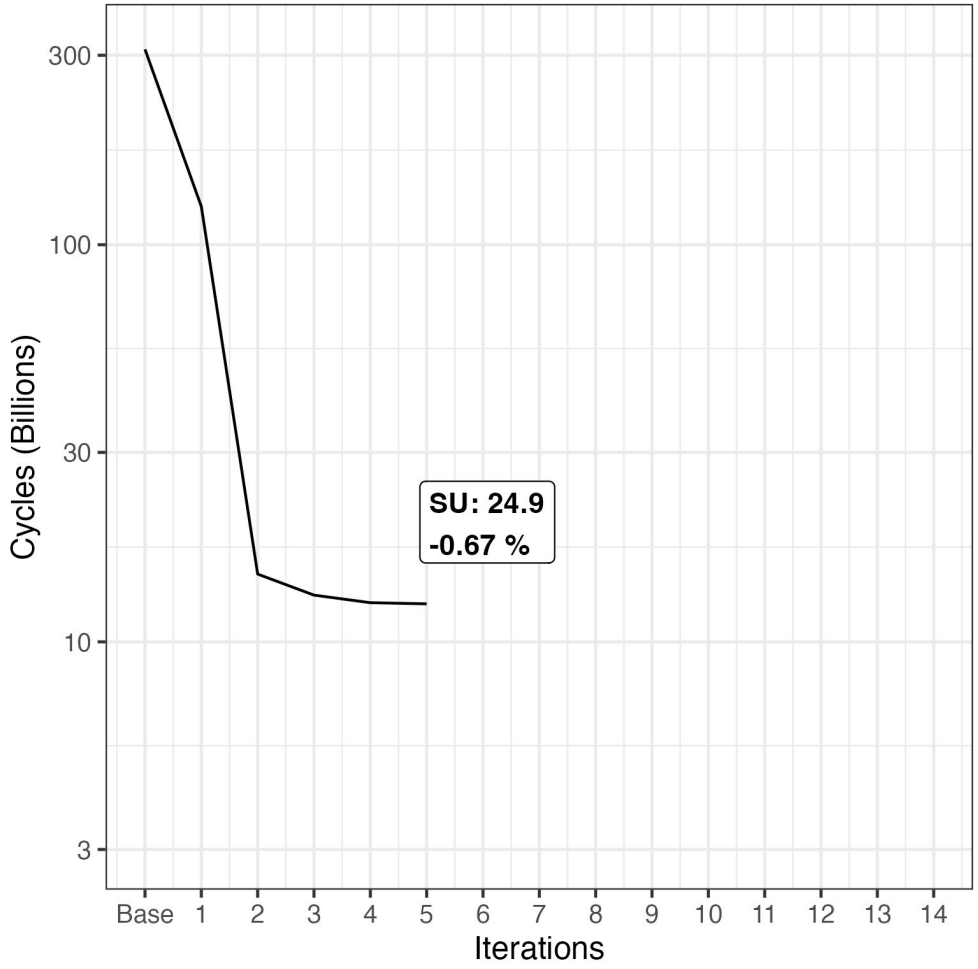
Inlining additional improvement



Global Parameter Initialization

Defined global parameters r,H,M,o and corners

Calculated once, not each time solve-function is called

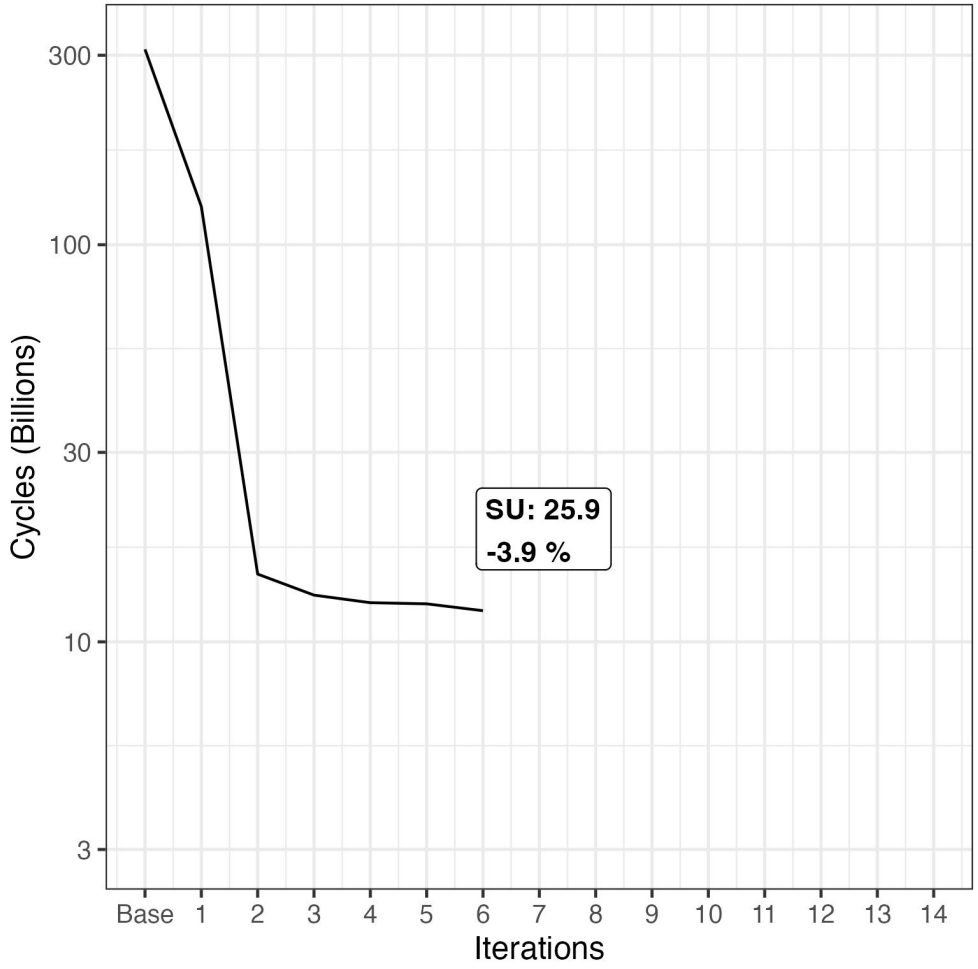


Combined for-loops

Combined the loops that

- propagate the alldifferent constraint
- propagate the alldifferent results to bounds

They iterate the same scope (redundancy)

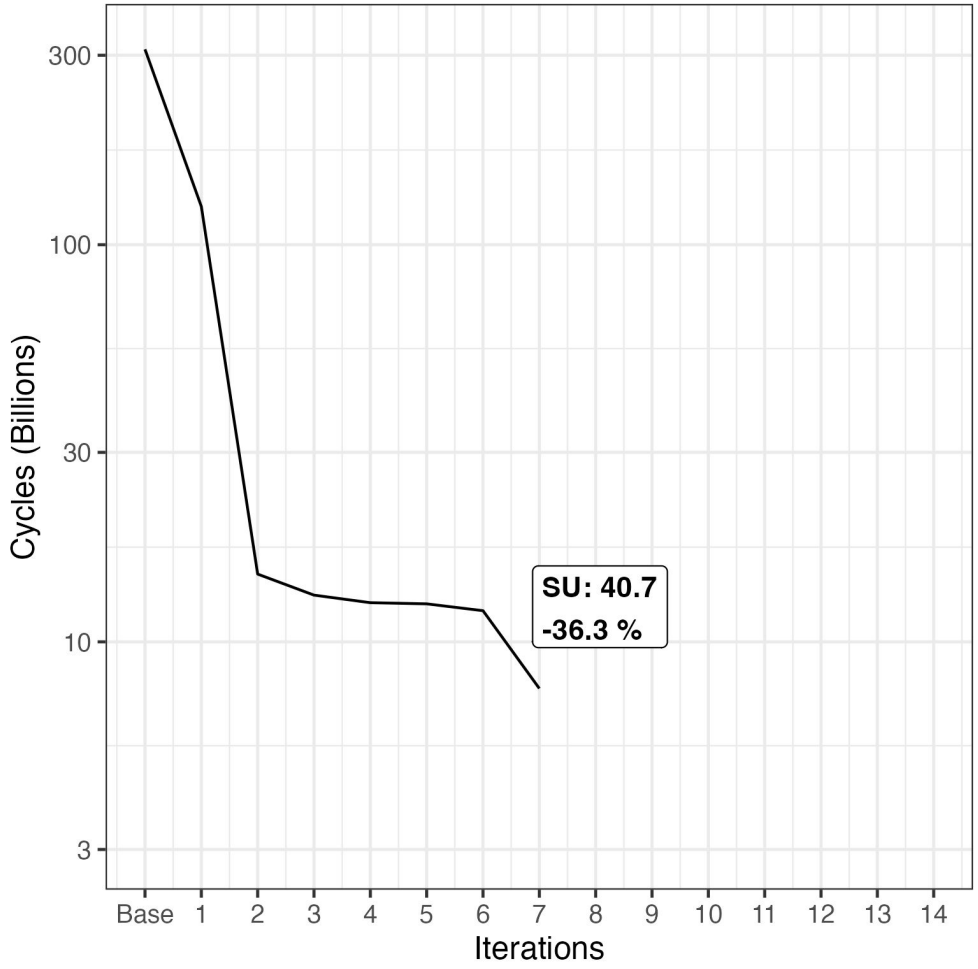


Changed Occupation Array Filling

Include while loop in occupation array filling

As long as free value for low/high, we increase low/high

Removed Combined for-loops from previous slide

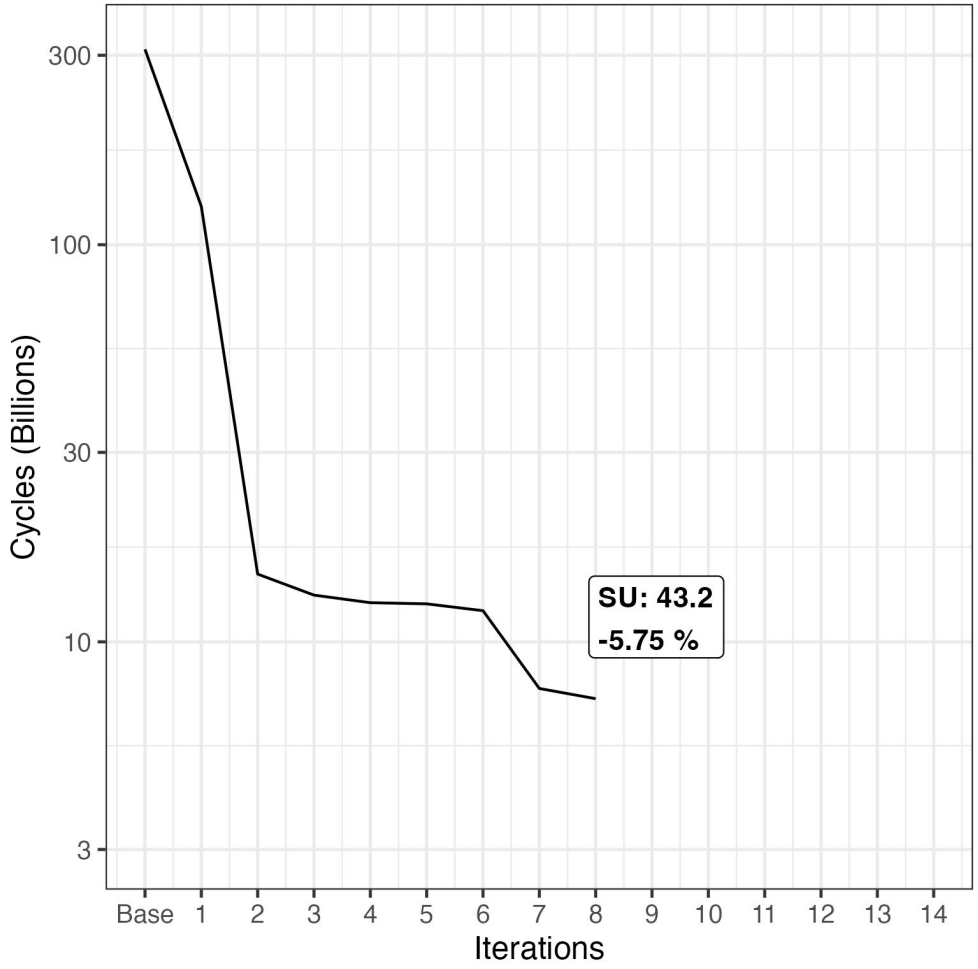


Sum Calculation Changed Flag

Introducing flag in Sum calculation:

- Continue sum calculation and not restart once a change is made
- Keep track in flag if a change happened
- If it did return 1

Introduced equality check of hi and lo, to move to next index

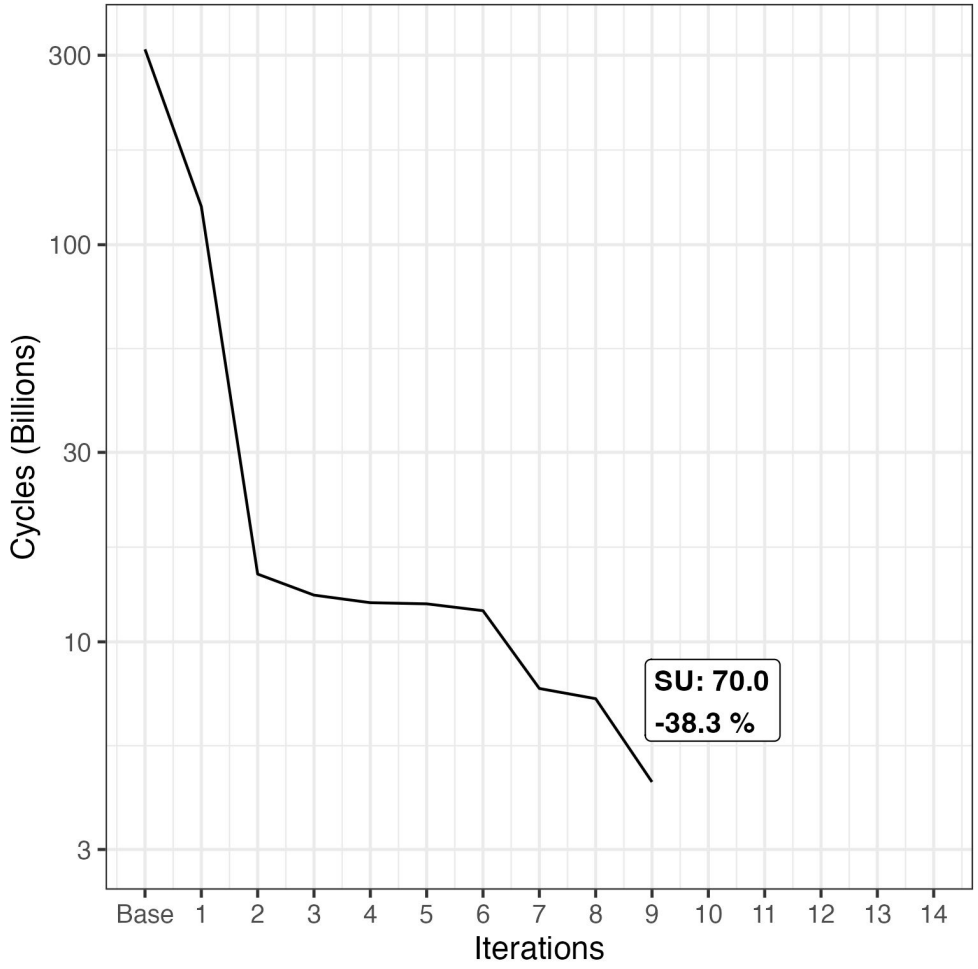


Bisection of Range

Not simply test all values between the lower and higher bound

Bisect range and make two separate calls to solve

Ideally fail early, find that one of these subranges does not contain a solution without exploring it in more depth.

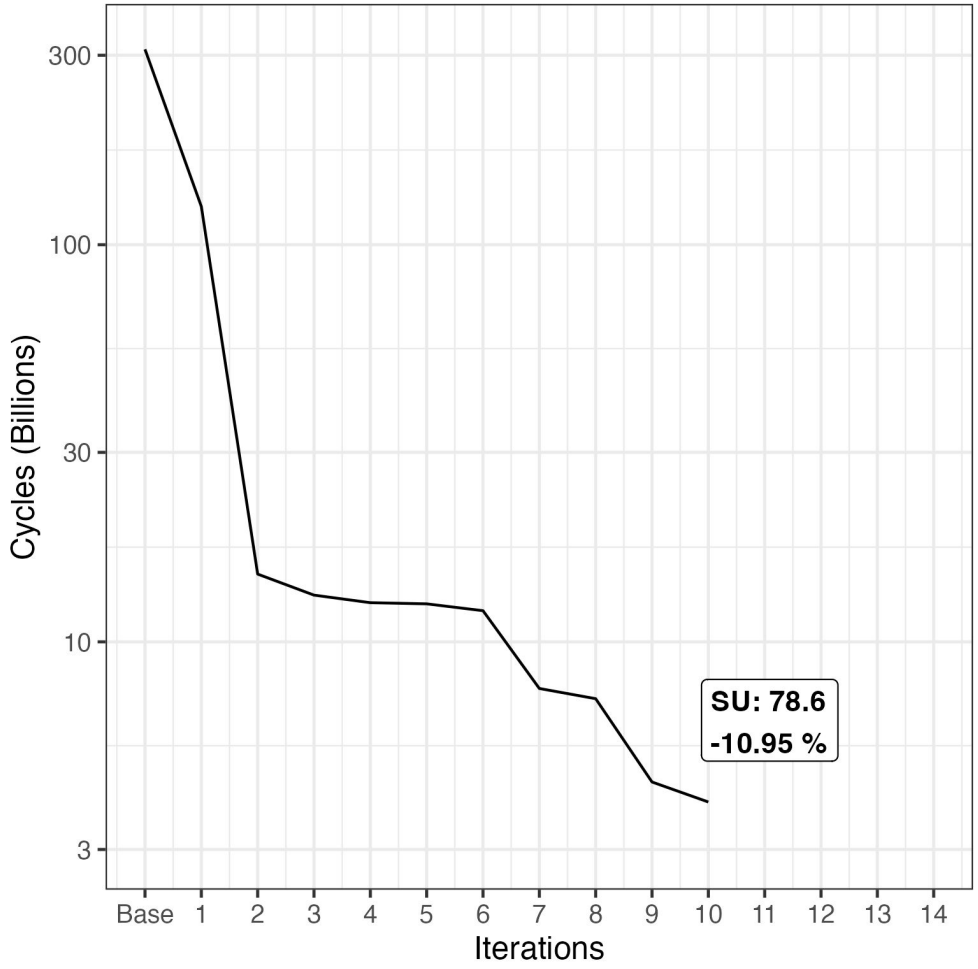


Highest Lower Bound Heuristic

At each labeling step, go through all indices

Select the variable with highest lower bound next

Indices are still at first explored by the spiral but after the first index heuristic takes over

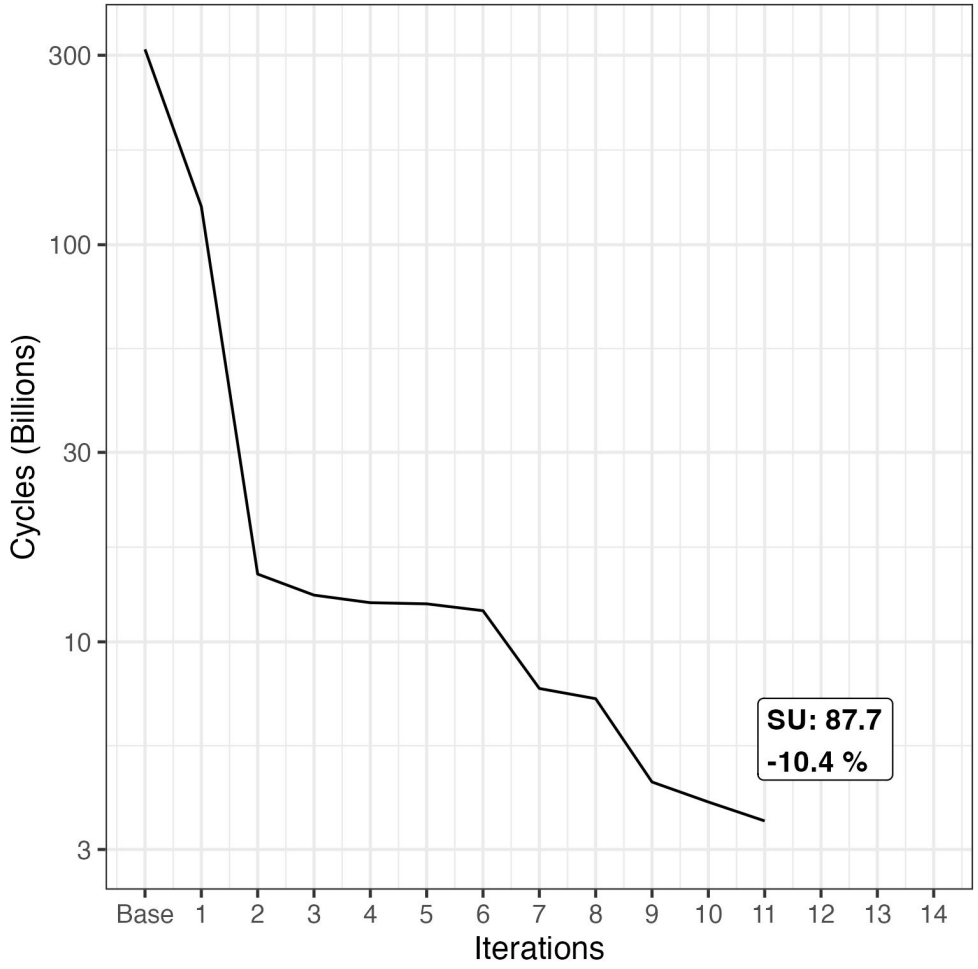


Corners First

Put the corners in the beginning of the static indexing

Remaining static indices are still spiral

Corners are involved in the most constraints



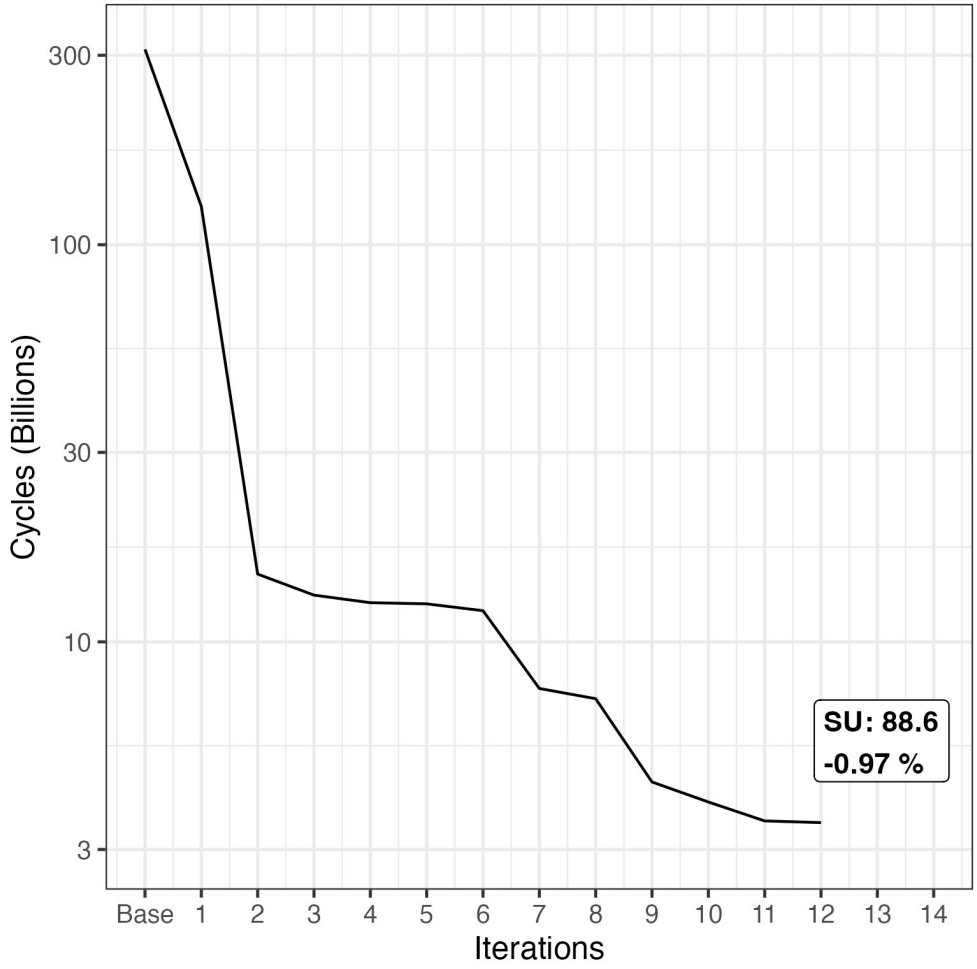
Spiral Path Removal

Remove spiral ordering, redundant at this point

Overhead of spiral computation is removed

Simply have static list with corners in first positions

Index selection is done by heuristic anyway

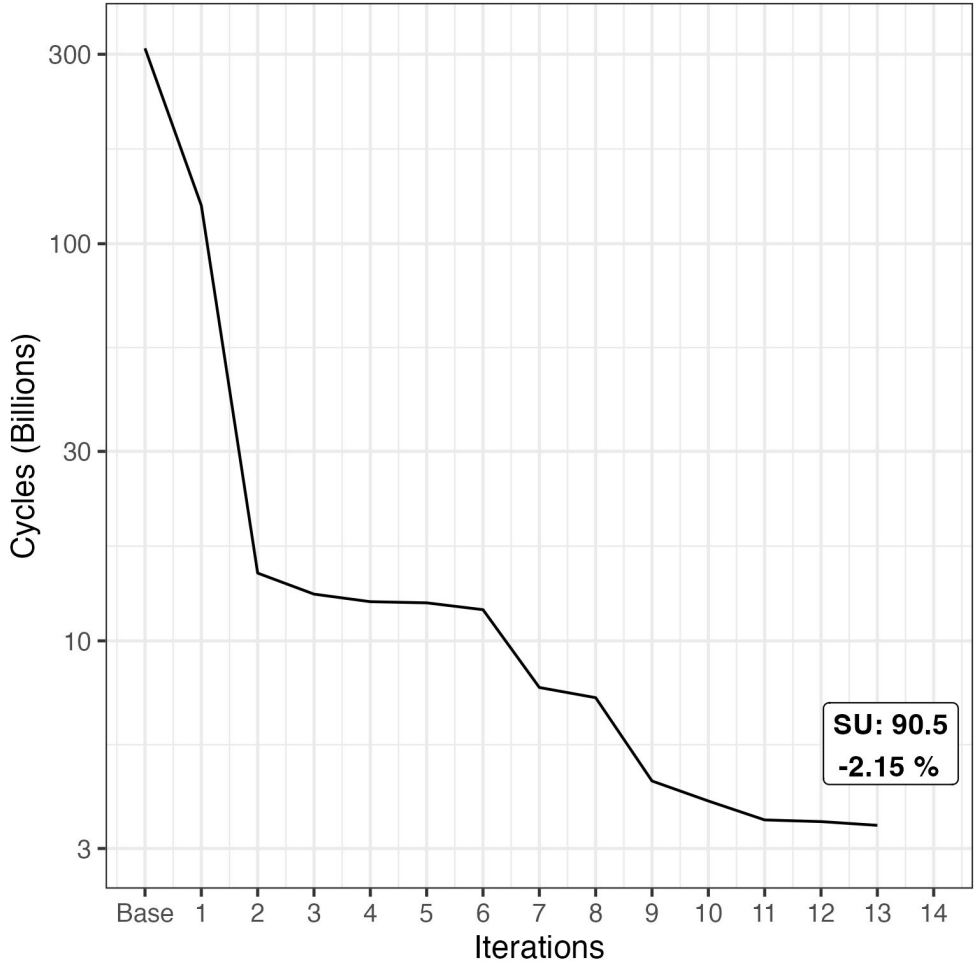


Early Stopping in Sum-Function

Introducing check in sum loop

Assert that hi and lo are still feasible after worst case bound calculation

If not return 0 immediately



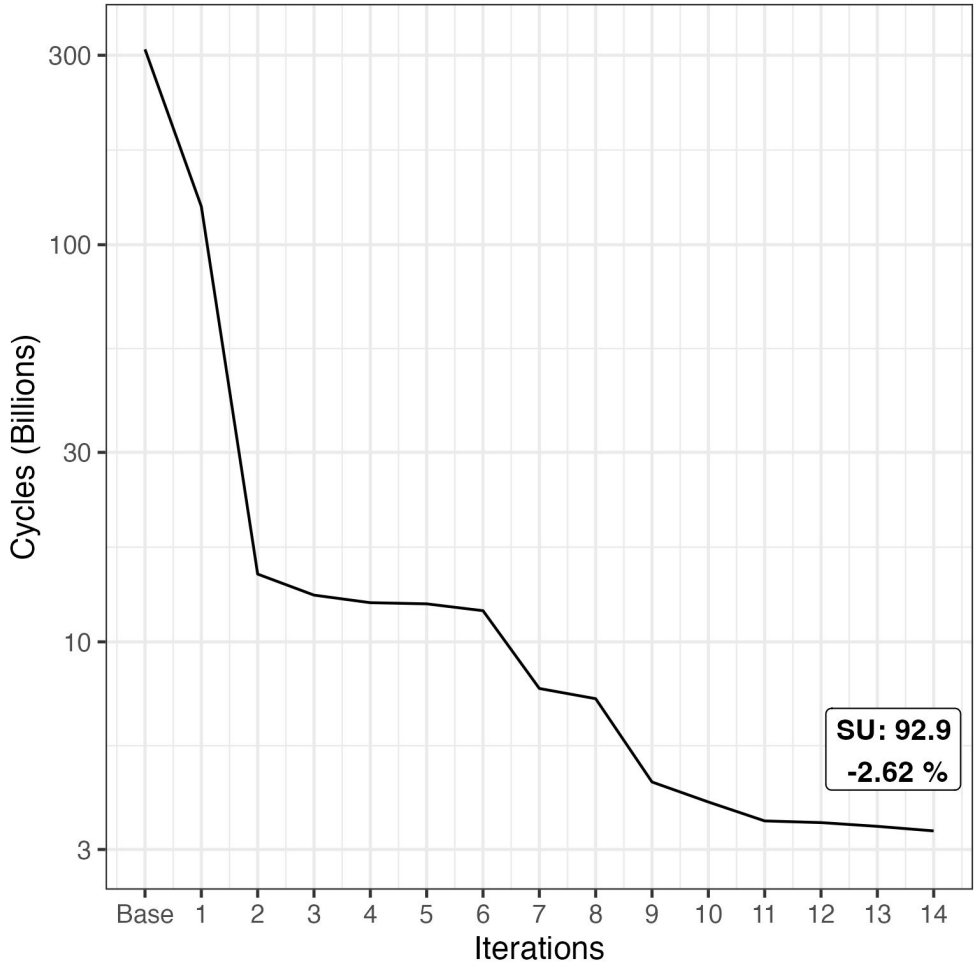
Limited Highest Lower Bound

Add back in spiral static indexing, with corners first

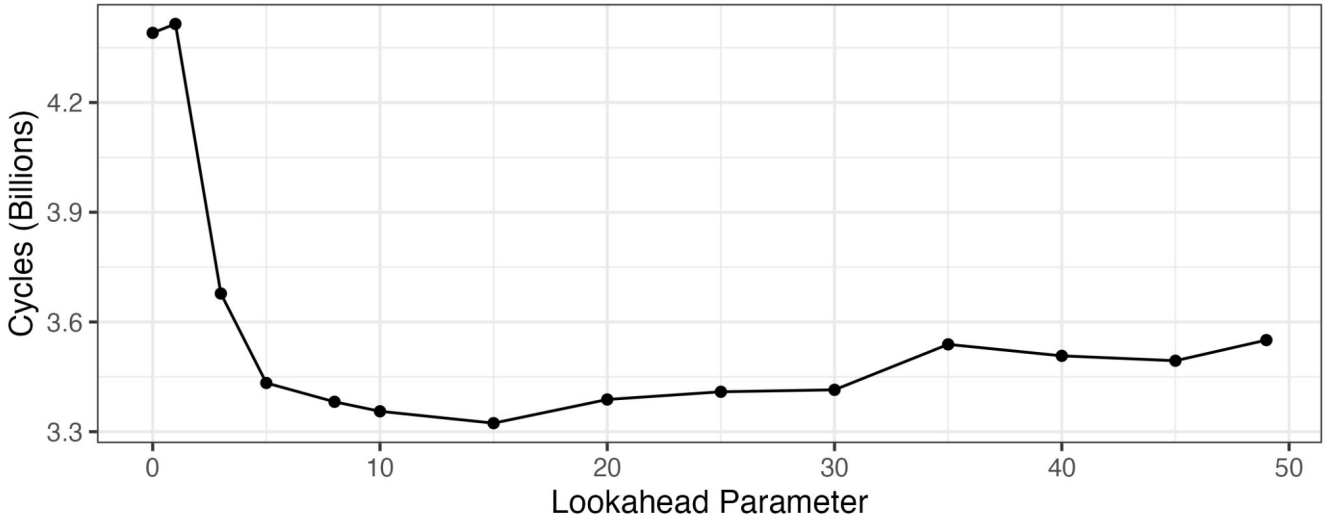
Bias heuristic via a lookahead parameter

Choose index with the best heuristic within the lookahead of the static indices

Best parameter for benchmark: 15

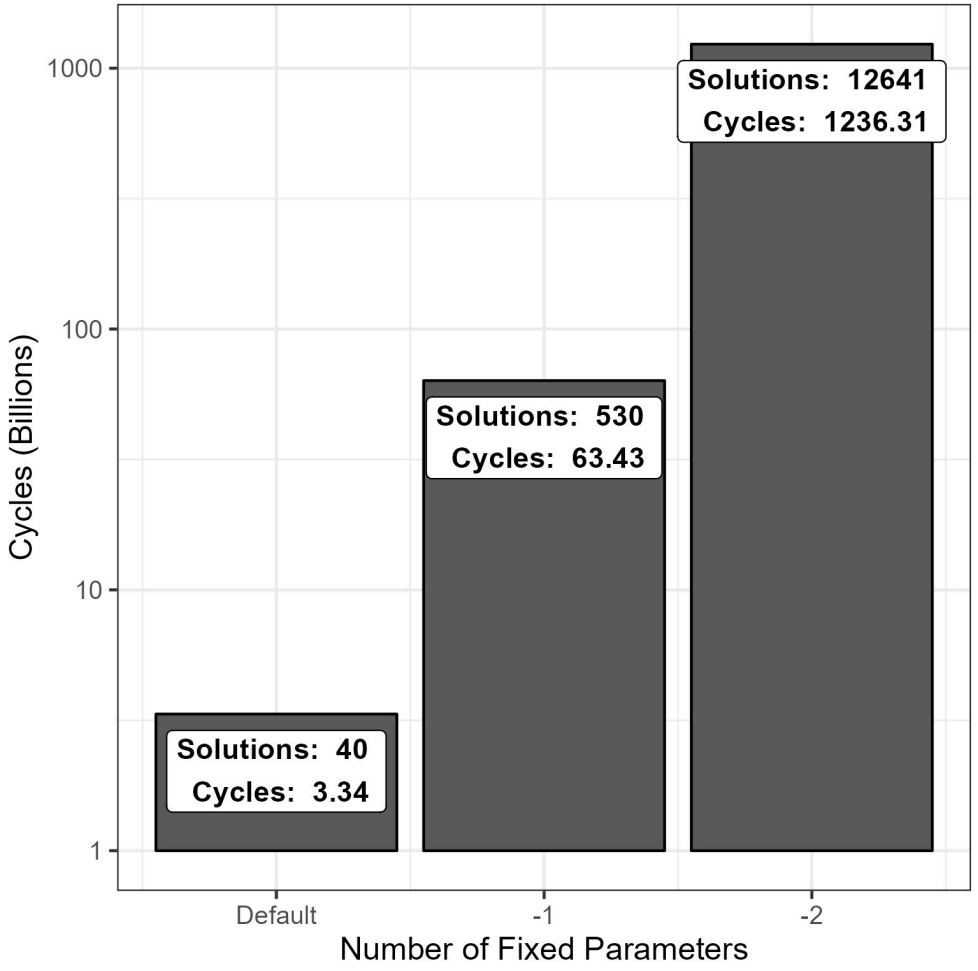


Limited Highest Lower Bound



Result overview

N. of Fixed Param.	CPU Time [s]	Elapsed Time [s]
Default	0.715	0.715
-1	13.525	13.526
-2	264.431	266.544



Subpar Approaches

- Minimal value range heuristic
- Trail stack
- Loop unrolling
- Occupation filling with -1
- Remove sizeof calculation
- Different order of constraint check
- Flags

Thank you for your Attention !

Appendix

Version	Billion Cycles
Base	310,51
1	124,82
2	14,81
3	13,11
4	12,55
5	12,46
6	11,98
7	7,63
8	7,19
9	4,44
10	3,95
11	3,54
12	3,51
13	3,43
14	3,34