Testing Hyper-Heuristics: Sudoku Applications

Hugh Simmons
Supervised by Matthew Davison and Rebecca Hamm
Lancaster University Intern

An Introduction to the Sudoku Problem

- The **aim of the project** is to code a programme that will find a valid solution to an empty Sudoku of order n as quickly as possible.
- "Sudoku" is an abbreviation of the Japanese phrase "Suuji wa dokushin ni kogiru", meaning "the numbers must remain single". Contrary to the Japanese name, the 1st modern Sudoku was published in the US in 1979 by Howard Garns; a retired architect and puzzlemaker.

3			8		1			2
2		1		3		6		4
			2		4			
8		9				1		6
	6						5	
7		2				4		9
			5		9			
9		4		8		7		5
6			1		7			3

© Encyclopædia Britannica, Inc.

Fig. 1: An example of a standard Sudoku

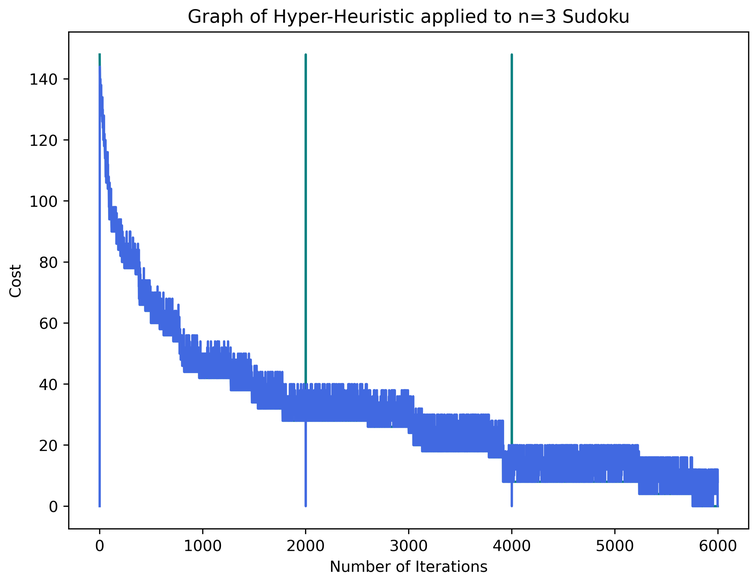
- The standard Sudoku puzzle is a 9 by 9 grid of cells, each to be filled with a number from 1 to 9. This grid can be further split into 9 boxes of 3 cells by 3 cells and, importantly, each number must only appear once in a given column, row or box.
- In the literature, this is referred to as an order 3 Sudoku. **An order n Sudoku** consists of a grid of n^2 by n^2 cells, into which the numbers 1 to n must be placed, with the same constraints as before.

The Backtracking Algorithm

- The backtracking algorithm is a **systematic method** for solving standard Sudoku puzzles, we modified it to solve order n puzzles before applying it to empty grids and recording its run time.
- The algorithm works by **stepping through the grid** one cell at a time and discarding the combinations that aren't valid until it eventually finds a solution.
- An **important quality** of the backtracking algorithm is that given enough time it will find a solution, this is not the case for the heuristic method due to its random nature.
- However, though this method works very well for low values of n, it **scales poorly** for higher n. So there is demand for a method with more desirable scaling such as a heuristic method.

Hyper-Heurisitcs

- A Hyper-Heuristic is a **heuristic search method** that seeks to automate the selection of heuristics to solve a search problem as efficiently as possible.
- This is done by defining a **cost function** that tells us how close to a valid solution a certain grid is and then reducing it through random swaps within rows, columns or boxes.
- In our case the random swaps within rows, columns, boxes and the whole Sudoku are our **heuristics**.



- For every iteration of the heuristic algorithm, 2 decisions must be made:
  - i **Which heuristic** to use to generate the next potential solution.
  - ii Whether to **accept** this new solution.
- The acceptance method used for the above solutions was Simulated Annealing, where the probability of acceptance is given as follows [2]:

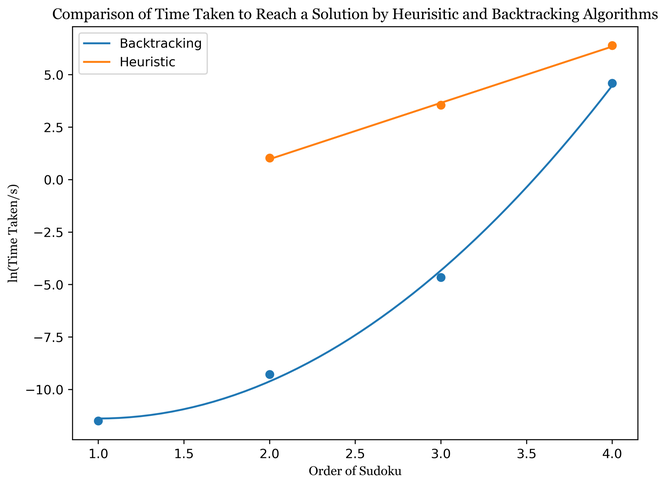
$$p = \left\{ \begin{array}{ll} 1, & \text{if } ProposedCost < CurrentCost \\ e^{\delta/t}, & \text{if } ProposedCost \geq CurrentCost \end{array} \right\}, \ \delta = CurrentCost - ProposedCost \quad (1)$$

- The **temperature parameter**, t, was stepped down and up at certain iteration numbers; tuned by the order 4 case. This makes it less and more likely, respectively, that a worse solution is accepted.
- Heuristics were **selected randomly**, which didn't perform significantly worse than biasing towards the heuristics which worked best.

Comparison

	8		14	10		12			13				2	3	
		3	4			14	16			11		15	12		8
		9						3			8		13		
	7				4	8	5			1	2		14	6	
			8	16		10	1		15		3	2		9	
11	4			9	14		12	7	5						13
9		2	10	4					1	13	16	5			
6	1			8	7	13			2	9		14		11	
	2		12		16	7			6	15	5			10	3
			13	15	10	6					11	9	8		2
1						3	8	13		16	14			12	5
	3		11	5		1		2	8		4	13			
	15	1		12	3			16	11	10				13	
		13		2			6						9		
14		12	7		1			15	4			8	3		
	11	4				16			9		13	12		2	

Fig. 3: An example of an order 4 Sudoku



- Both algorithms were run for a **range** of order Sudoku and we can see that, as hoped, the heuristic algorithm looks to **scale** better than the backtracking algorithm.
- Furthermore, if the plotted fits are extrapolated to order 5 then the heuristic is predicted to **outperform** the backtracking by a multiple of over 2000.

References

[1] Broderick Crawford et al. "Using Constraint Programming to solve Sudoku Puzzles". In: *2008 Third International Conference on Convergence and Hybrid Information Technology*. Vol. 2. 2008, pp. 926–931. DOI: 10.1109/ICCIT.2008.154.

[2] Rhydian Lewis. "Metaheuristics can solve Sudoku puzzles". In: *J. Heuristics* 13 (July 2007), pp. 387–401. DOI: 10.1007/s10732-007-9012-8.

STOR-i | Lancaster University