

# On the Combination of Constraint Programming and Stochastic Search: The Sudoku Case

**Rhydian Lewis**

Cardiff Business School

Pryfysgol Caerdydd/ Cardiff University

`lewisR9@cf.ac.uk`

# Talk Plan

- Introduction: What is Sudoku?
- Making and solving Sudoku puzzles
- A simple stochastic algorithm for Sudoku
  - General characteristics of the algorithm
- Improving performance using constraint-based techniques
- Conclusions and further work

# Introduction

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

*Example of an order-3 logic-solvable grid*

- ... the 21<sup>st</sup> Century's *Rubik's Cube*
- Given a partially filled  $n^2 \times n^2$  grid, fill each of the blank cells so that each row, column, and "box" contains 1 through to  $n^2$  exactly once.
- Order-3 grids are the most popular, though others are possible

# Introduction

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

*Example of an order-3 logic-solvable grid*

- ... the 21<sup>st</sup> Century's *Rubik's Cube*
- Given a partially filled  $n^2 \times n^2$  grid, fill each of the blank cells so that each row, column, and "box" contains 1 through to  $n^2$  exactly once.
- Order-3 grids are the most popular, though others are possible

# Introduction

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

*Example of an order-3 logic-solvable grid*

- ... the 21<sup>st</sup> Century's *Rubik's Cube*
- Given a partially filled  $n^2 \times n^2$  grid, fill each of the blank cells so that each row, column, and "box" contains 1 through to  $n^2$  exactly once.
- Order-3 grids are the most popular, though others are possible

# Making and Solving Puzzles

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

*Example of an order-3  
logic-solvable grid:*

- There is a unique solution*
- No guessing required*

- The puzzle master should supply a **logic-solvable** puzzle
  - Unique solution
  - No guessing required (deductive techniques only)
- Computers can easily solve such puzzles
  - [www.sudokusolver.com](http://www.sudokusolver.com)
  - [www.scanraid.com](http://www.scanraid.com)
- Thus, only certain grid formations will constitute a “good” puzzle (from a human and/or solver’s perspective)

# Making and Solving Puzzles

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

*Here all 3's have been removed:*

- More than one solution*
- Guessing may be required*

- However, not all puzzles will be logic-solvable
- E.g. removing some numbers from the left grid may mean:
  - Deductive rules are not enough;
  - Searching/ guessing is required
  - More than one solution
- Logical solvers thus rely on being given “good” puzzles.

Sudoku is an **NP-complete** puzzle. It also has an exponential search space. B&B is one possibility. But what about other types of search?

# A Stochastic Search Method

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

1) Take a valid Sudoku puzzle

# A Stochastic Search Method

1	2	4	1	5	7	6	9	2	2
6	5	9	3	4	2	3	8	7	1
8	7	3	6	8	9	4	1	5	1
4	3	1	6	8	5	4	7	9	1
5	2	6	7	1	9	1	3	2	2
7	9	8	4	3	2	8	6	5	1
2	7	9	7	1	8	8	2	1	4
8	4	6	2	9	3	9	3	4	3
3	1	5	5	6	4	7	5	6	3
1	2	2	2	2	2	2	1	2	34

Row Scores

Column Scores

Cost

- 1) Take a valid Sudoku puzzle
- 2) Randomly add the missing values to each box and evaluate

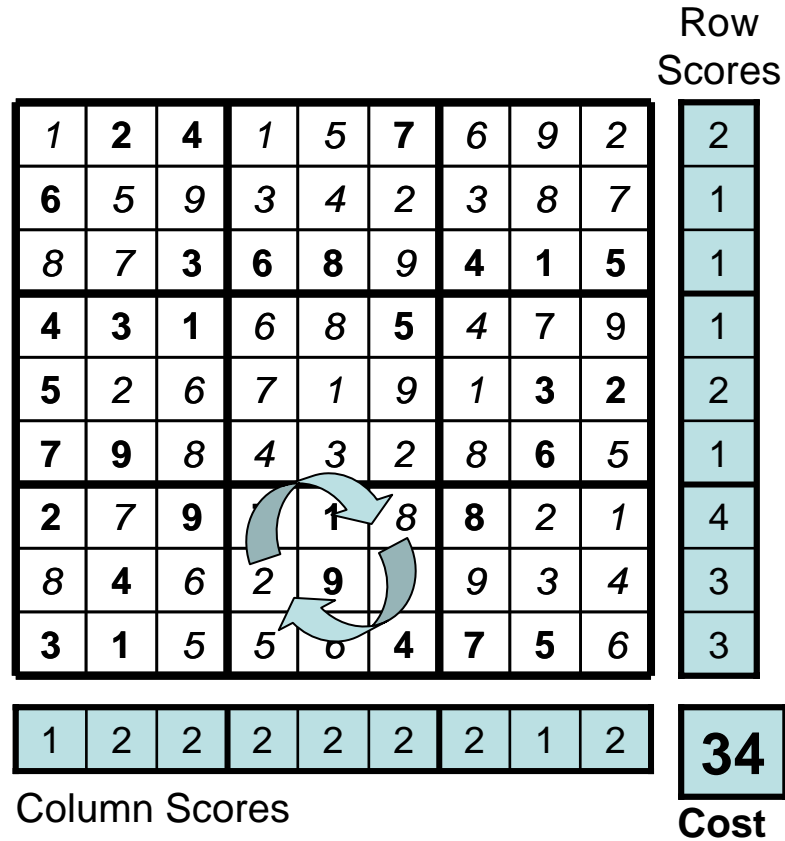
# A Stochastic Search Method

1	2	4	1	5	7	6	9	2	2
6	5	9	3	4	2	3	8	7	1
8	7	3	6	8	9	4	1	5	1
4	3	1	6	8	5	4	7	9	1
5	2	6	7	1	9	1	3	2	2
7	9	8	4	3	2	8	6	5	1
2	7	9	1	8	8	2	1	4	4
8	4	6	2	9	9	3	4	3	3
3	1	5	5	0	4	7	5	6	3
1	2	2	2	2	2	2	1	2	34

Row Scores

Column Scores

Cost



- 1) Take a valid Sudoku puzzle
- 2) Randomly add the missing values to each box and evaluate
- 3) Iteratively apply the neighbourhood operator and re-evaluate

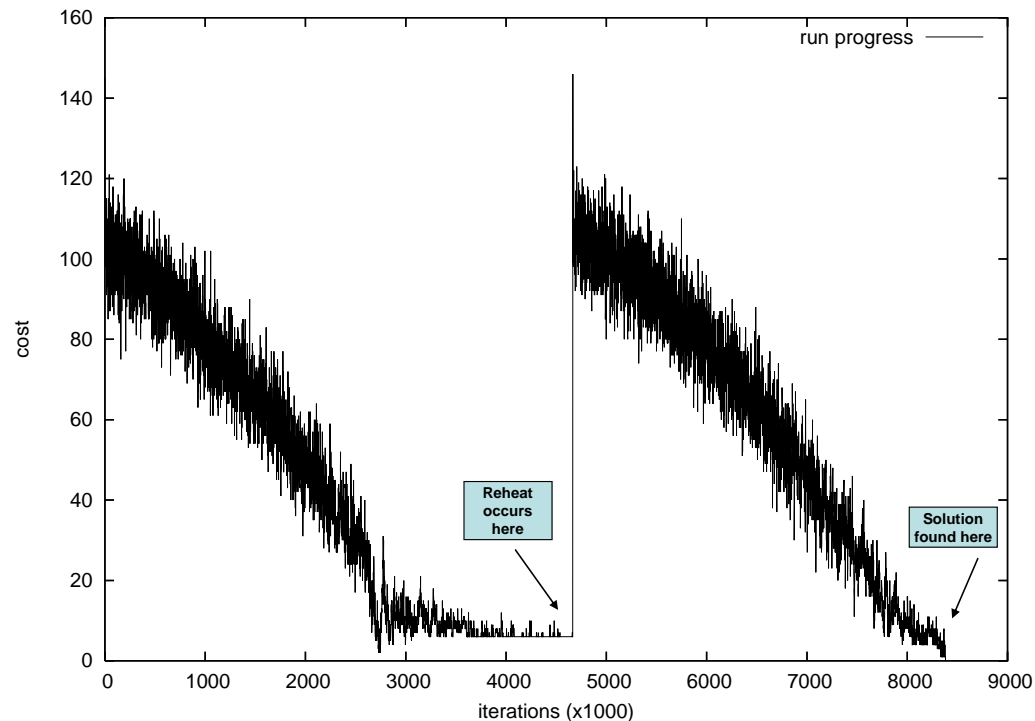
# A Stochastic Search Method

1	2	4	9	5	7	3	8	6	0
6	8	5	3	4	1	2	9	7	0
7	9	3	6	8	2	4	1	5	0
4	3	1	2	6	5	9	7	8	0
5	6	8	4	7	9	1	3	2	0
7	9	2	1	3	8	5	6	4	0
2	5	9	7	1	6	8	4	3	0
8	4	7	5	9	3	6	2	1	0
3	1	6	8	2	4	7	5	9	0
0	0	0	0	0	0	0	0	0	0

- 1) Take a valid Sudoku puzzle
- 2) Randomly add the missing values to each box and evaluate
- 3) Iteratively apply the neighbourhood operator and re-evaluate
- 4) **Stop when an optimal grid is found**

- Problem instances do not have to be logic solvable for this algorithm to be able to find a solution
- Our particular algorithm made use of **Simulated Annealing**

# Solving Published Puzzles



Example Run with an Order-4 (16 x 16)  
instance taken from *The Wales on Sunday*

Order 3 logic-solvable  
puzzles from a variety of  
newspapers quickly solved  
(less than 2 seconds)

No relationship with  
perceived “difficulty rating”

Order-4 puzzles typically  
solved in 5-15 seconds  
(some reheating needed  
occasionally)

- Only one solution in each case.
- In many cases, logic based algorithms are faster

# Solving “Random” Puzzles

- Because the SA algorithm does not need a puzzle to be logic solvable, what *does* make an instance difficult to solve?
- A new method of puzzle generation was implemented:

1) Take a complete and valid Sudoku puzzle

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

# Solving “Random” Puzzles

- Because the SA algorithm does not need a puzzle to be logic solvable, what does make an instance difficult to solve?
- A new method of puzzle generation was implemented:

- 1) Take a complete and valid Sudoku puzzle
- 2) **Alter the puzzle whilst keeping validity**

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

# Solving “Random” Puzzles

- Because the SA algorithm does not need a puzzle to be logic solvable, what does make an instance difficult to solve?
- A new method of puzzle generation was implemented:

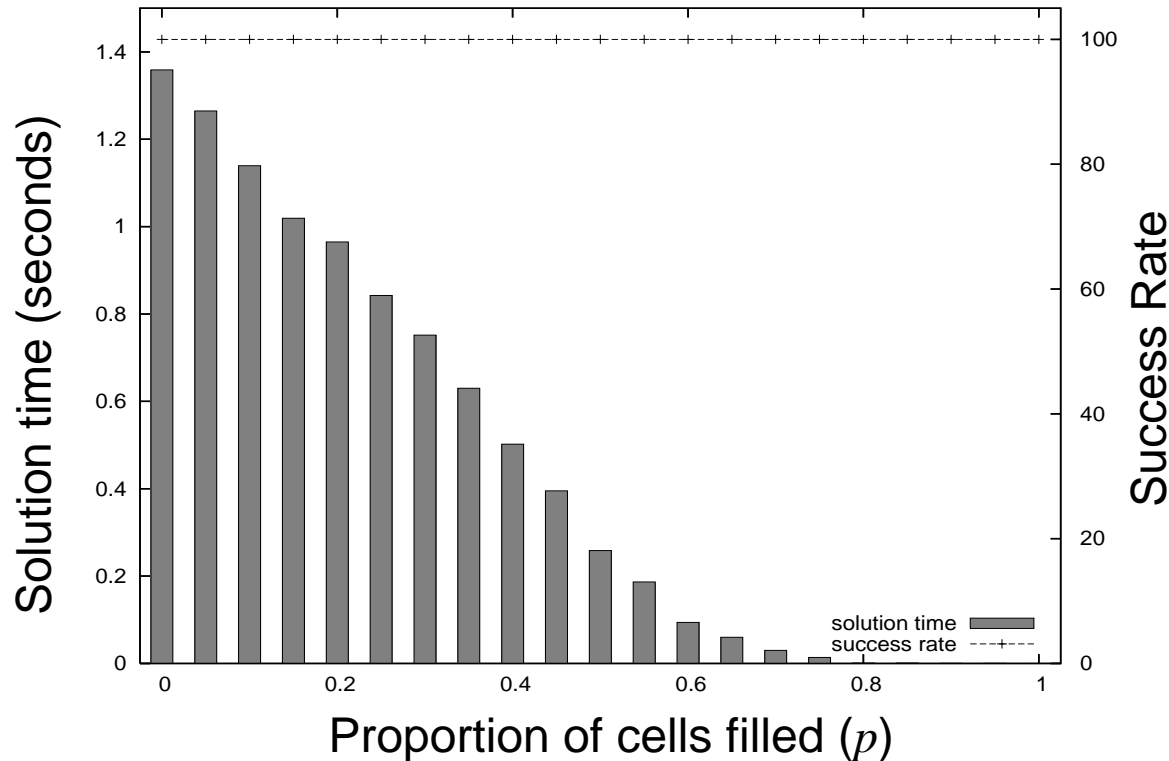
- 1) Take a complete and valid Sudoku puzzle
- 2) Alter the puzzle whilst keeping validity
- 3) **Go through each cell and remove each entry with a probability  $1 - p$ , (where  $p = [0, 1]$ ).**

For low  $p$ 's,

- Puzzles may have more than one solution
- Not necessarily logic solvable

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

# Algorithm Performance for $n = 3$ : i.e. (9 x 9) grids



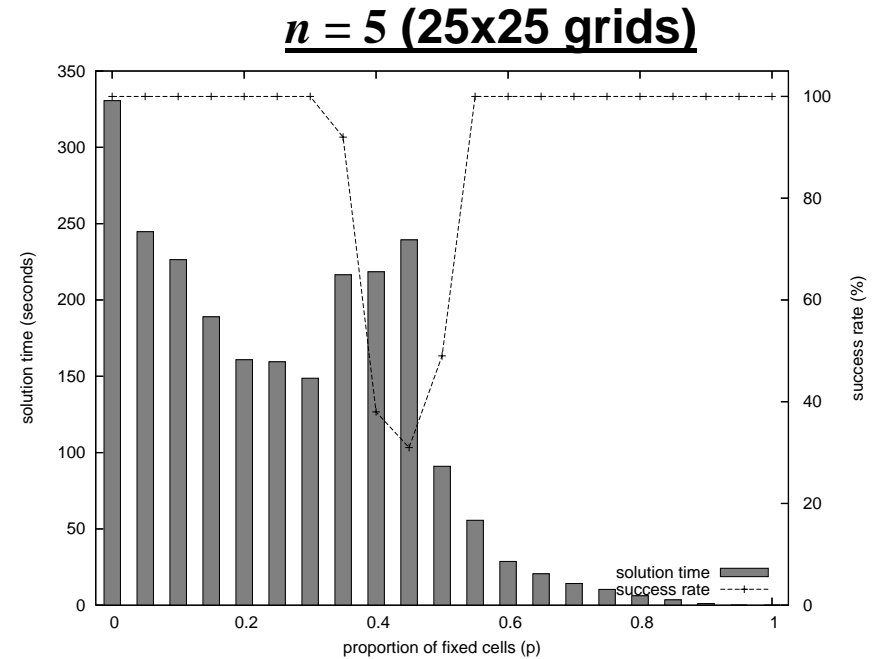
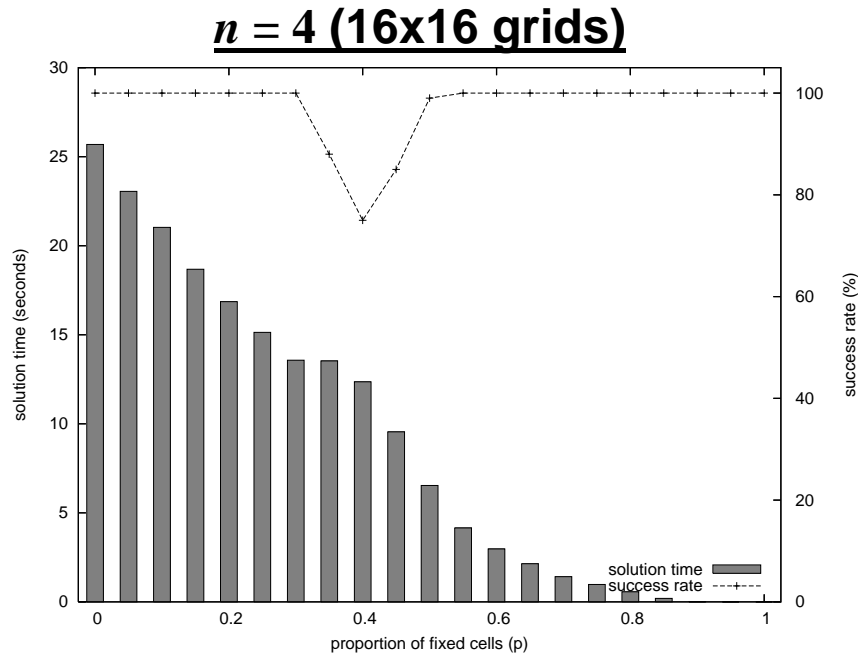
- For low  $p$ 's there are many different optimal solutions (6,670,903,752,021,072,936,960 for  $p = 0$ )

- For high  $p$ 's the search space will be small and will have a strong basin of attraction

Empty grids  
(large search space)

Full grids  
(small search space)

# Algorithm Performance for $n = 4$ and $n = 5$

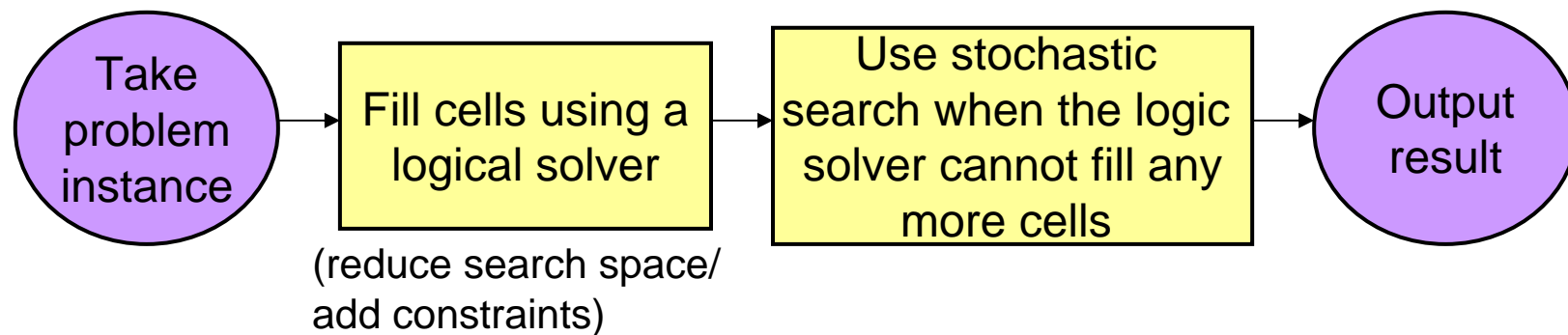


- Similar patterns to  $n = 3$ , though a phase transition region is also visible
- The phase transition causes a fluctuation in both the success rate and solution time

# Improving Algorithm Performance

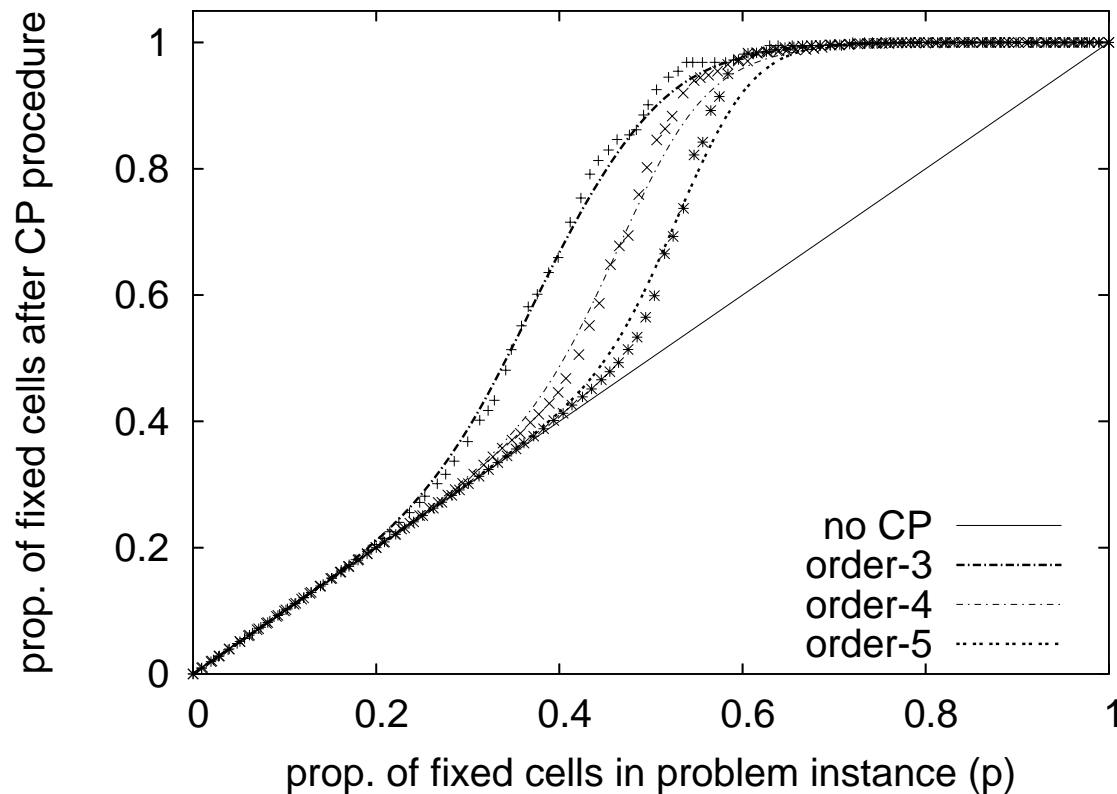
- Phase transition seems to be caused by two factors:
  - Relatively large search space, but very small number of solutions
  - The moderate number of constraints will cause a more inhospitable cost landscape
- **Q:** Can we improve performance by adding a logic-based solver (constraint programming)?

## Proposed *Hybrid* Method



# Solving Random Instances

## Performance of logic-based procedure On random instances of various $p$ 's

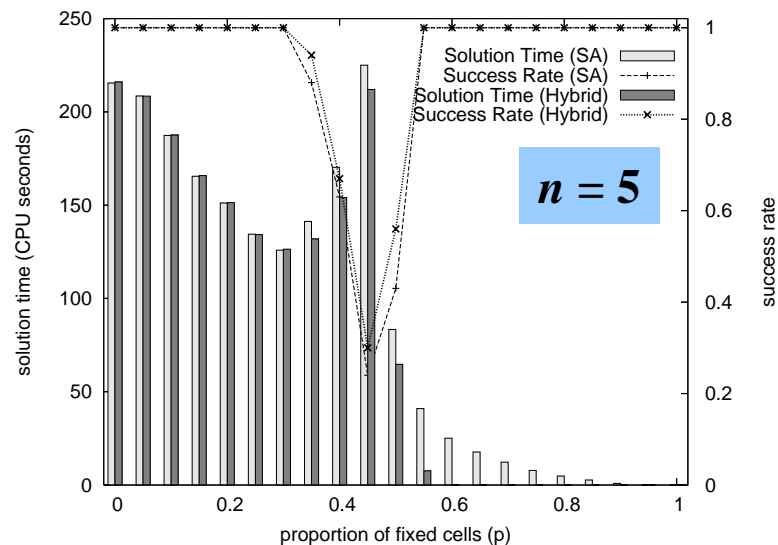
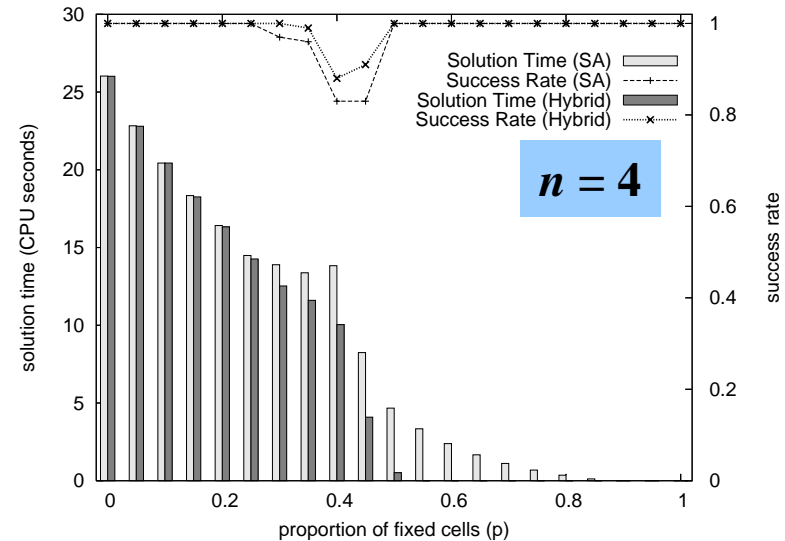
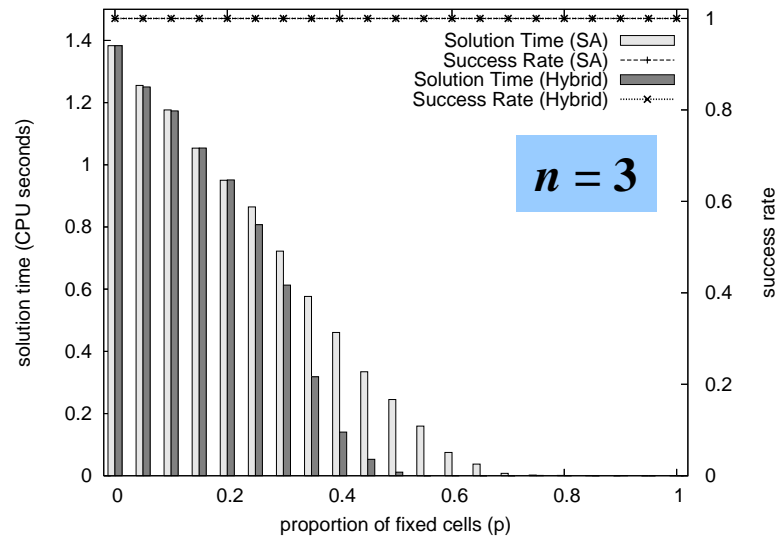


Empty grids

Full grids

- For  $p < 0.2$  there are insufficient clues for any cells to be filled
- For  $p > 0.7$ , the procedure completes the puzzles.
- Between these values, *some but not all* cells are filled

# Solving Random Instances: Hybrid Vs SA algorithm



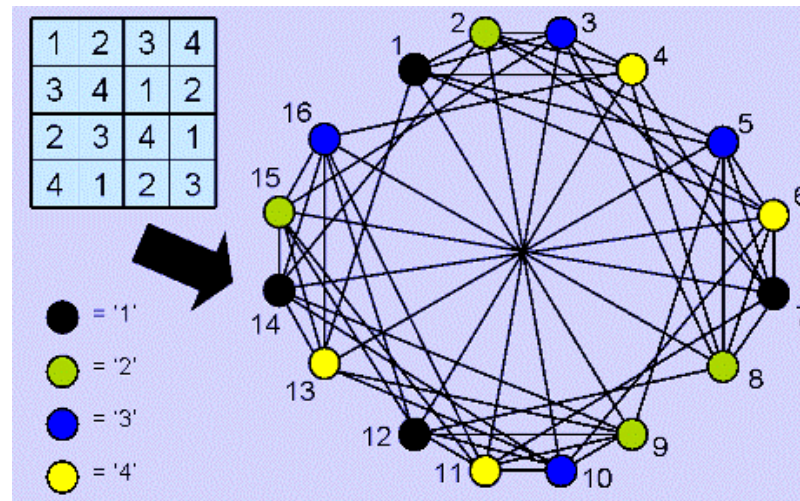
- For  $p < 0.2$  the algorithms are equivalent
- For  $p > 0.5$  the hybrid algorithm has shorter run times
- Success rate of the hybrid algorithm is greater throughout the phase transition regions

# Conclusions and Further Work

- Experiments have shown that the hybrid algorithm also outperforms the SA algorithm on many published, logic-solvable instances.
- The CP and stochastic algorithms seem to complement one another: improvements in one aspect should lead to enhanced overall performance
- CP procedure has the potential to move instances out of the phase transition region. Can it move other instances *into* the region?

# Further Work

- Though Sudoku is a silly little problem, it does reflect aspects of real world problems: e.g. timetabling and scheduling.
- Sudoku can also be modelled as a graph colouring problem



- each of the  $n^4$  cells is a node; each entry  $\{1, \dots, n^2\}$  is a colour
- add edges between nodes to represent the constraints

- Can known graph colouring heuristics be employed to help solve Sudoku problems (and indeed, vice-versa)?

# On the Combination of Constraint Programming and Stochastic Search: The Sudoku Case

**Rhydian Lewis**

Cardiff Business School

Pryfysgol Caerdydd/ Cardiff University

`lewisR9@cf.ac.uk`