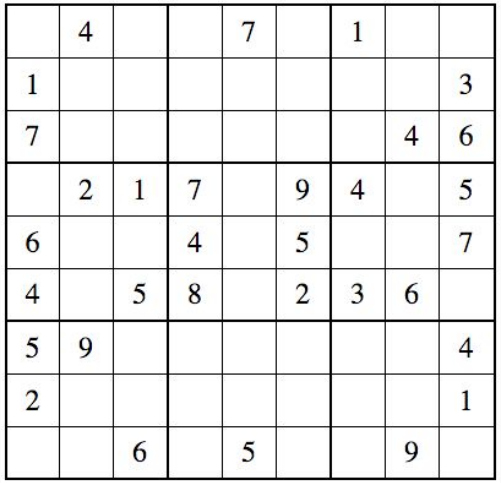
Parallel Sudoku Solver on Graphical Processing Units

Angela Tobin, Chenshan Shari Yuan and Muhammad Osama

What is a Sudoku Puzzle?

- Logic based number puzzle
- Filling 9x9 grid with digits from 1-9
  a. each column,
  b. row, and
  c. 3x3 block has all the numbers once and only once.



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

http://www.websudoku.com/?level=1&set_id=8789350568

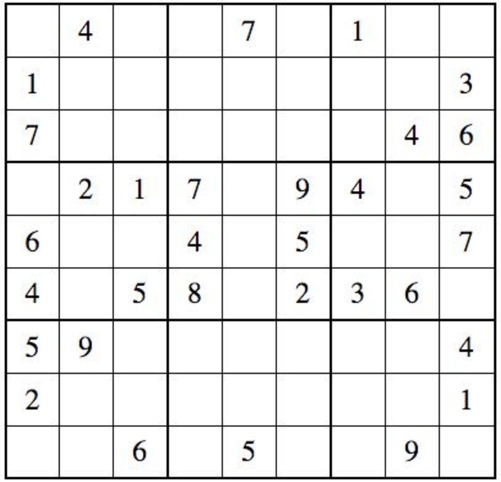
Why Sudoku on a GPU?

- 6.67 * 10^21 possible valid solutions for 9x9 Sudoku grid
  - Memory bandwidth bound
  - Compute intensive
- Mimics a genetic algorithm model
  - Chromosome represents a possible solution
  - Simulate genetic mutations
  - Compute quality of each solution to define new parents
  - Eventually converge to an absolute truth

Logic (Human-like) Approach

How to Teach Logic to a GPU

- To humans, solving a Sudoku puzzle is very logical
  - View entire puzzle at once - all accessible
  - Easy to jump around and use new information as it comes in
- GPU's ability to solve is only as good as its code
  - No eyes - How to view and store the data?
  - Clarity - How to pick out important data?
  - Efficiency - If multiple methods exist, how can we optimize our resources?



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

How to Teach Logic to a GPU

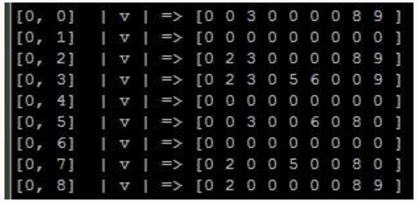
- Data Management: Square structure

```
struct Square {
        int value;              //Actual value of the Square
        int isLocked;           //Indicates the Square needs no more work
        int possValues[PUZZLE_SIZE]; //Contains all possible values the Square could be
};
```

- Important Data for each Square
  - Decision - One thread per Square
  - Each Square cares about the other Squares in the row, column, and block that it exists in

Logic Kernels

- Populate.cu
  - Removes values from the possValues array,
    if found as locked values in same row, column, or block
- Human.cu
  - Uses human-like logic to set values
  - Based on possValues arrays of Squares in the same row, column, and block
- Dependence
  - Both kernels need each other, and must cycle to gather new information to proceed



```
[0, 0]   | v | => [0 0 3 0 0 0 0 8 9 ]
[0, 1]   | v | => [0 0 0 0 0 0 0 0 0 ]
[0, 2]   | v | => [0 2 3 0 0 0 0 8 9 ]
[0, 3]   | v | => [0 2 3 0 5 6 0 0 9 ]
[0, 4]   | v | => [0 0 0 0 0 0 0 0 0 ]
[0, 5]   | v | => [0 0 3 0 0 6 0 8 0 ]
[0, 6]   | v | => [0 0 0 0 0 0 0 0 0 ]
[0, 7]   | v | => [0 2 0 0 5 0 0 8 0 ]
[0, 8]   | v | => [0 2 0 0 0 0 0 8 9 ]
```

Performance

- One pass through both populate.cu and human.cu takes approximately 0.87 ms
- After several iterations, stuck in local minima
    - No more numbers can be locked after a number of passes
    - Need more complex logic to eliminate numbers from possValues
        - (twins, triplets)

Brute Force Approach

Brute Force

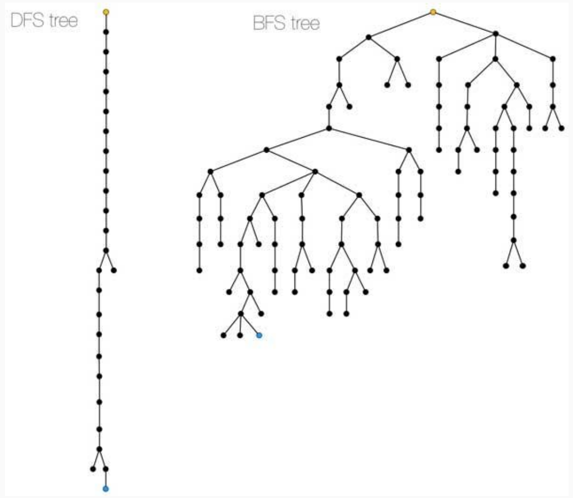
- BFS Search
    - Worst case: 9^N solutions to search
    - Memory bound search

Logic + BFS

- Logic + BFS
  - Logic effectively decrease number of solutions to search
  - Ease requirement on Memory

BFS + DFS

- BFS: Generate SOME possible valid Sudoku combo
- DFS: Try in the left empty spots in the generated Sudoku combo



Performance

- Naive Brute Force
    - Run out of memory at ~15-18 iterations. 300ms
- Logic + Brute Force
    - Run out of memory at ~20-25 iterations. 300-500ms
- Bfs + Dfs
    - Incorrect solution found ~500 ms

Artificial Bee Colony (ABC) Approach

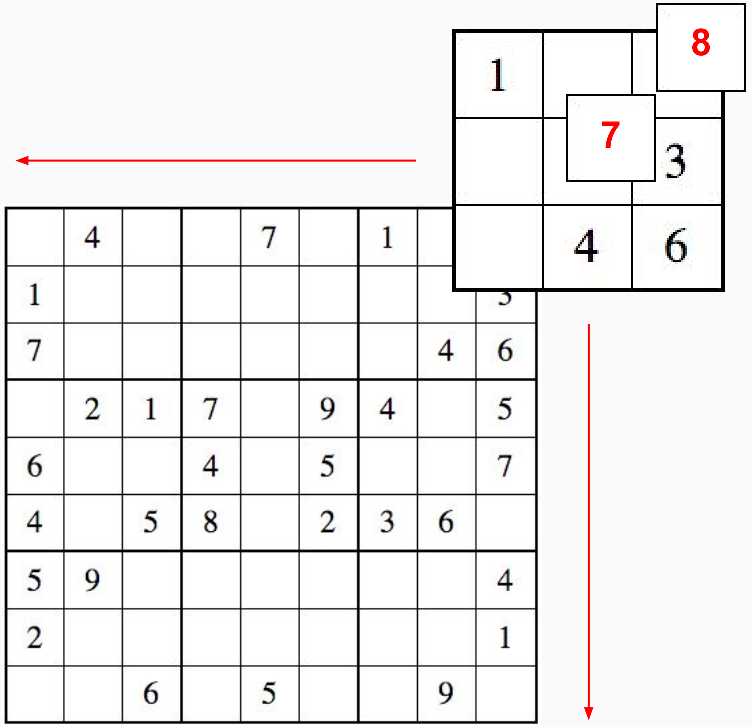
Overview: ABC Algorithm

Dervis Karaboga, An Idea Based On Honey Bee Swarm for Numerical Optimization.

- Initial food sources are produced for all **employed bees**
- REPEAT
    - Each employed bee goes to a food source in her memory and determines a neighbour source, then evaluates its nectar amount and dances in the hive
    - Each **onlooker** watches the dance of employed bees and chooses one of their sources depending on the dances, and then goes to that source.
    - After choosing a neighbour around that, she evaluates its nectar amount.
    - Abandoned food sources are determined and are replaced with the new food sources discovered by **scouts**.
    - The best food source found so far is registered.
- UNTIL (requirements are met)

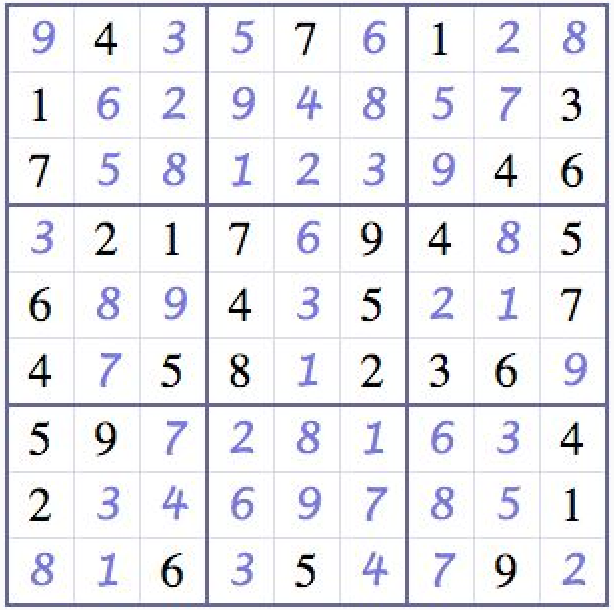
Employed Bees (threads/block)

1. Selects a random 3x3 grid section of sudoku puzzle.
2. Generates and assigns two random values (from 1-9) to two squares in the grid.
3. Computes the uniqueness of that element in the specific block, row and column.



Onlooker Bees (blocks/kernel)

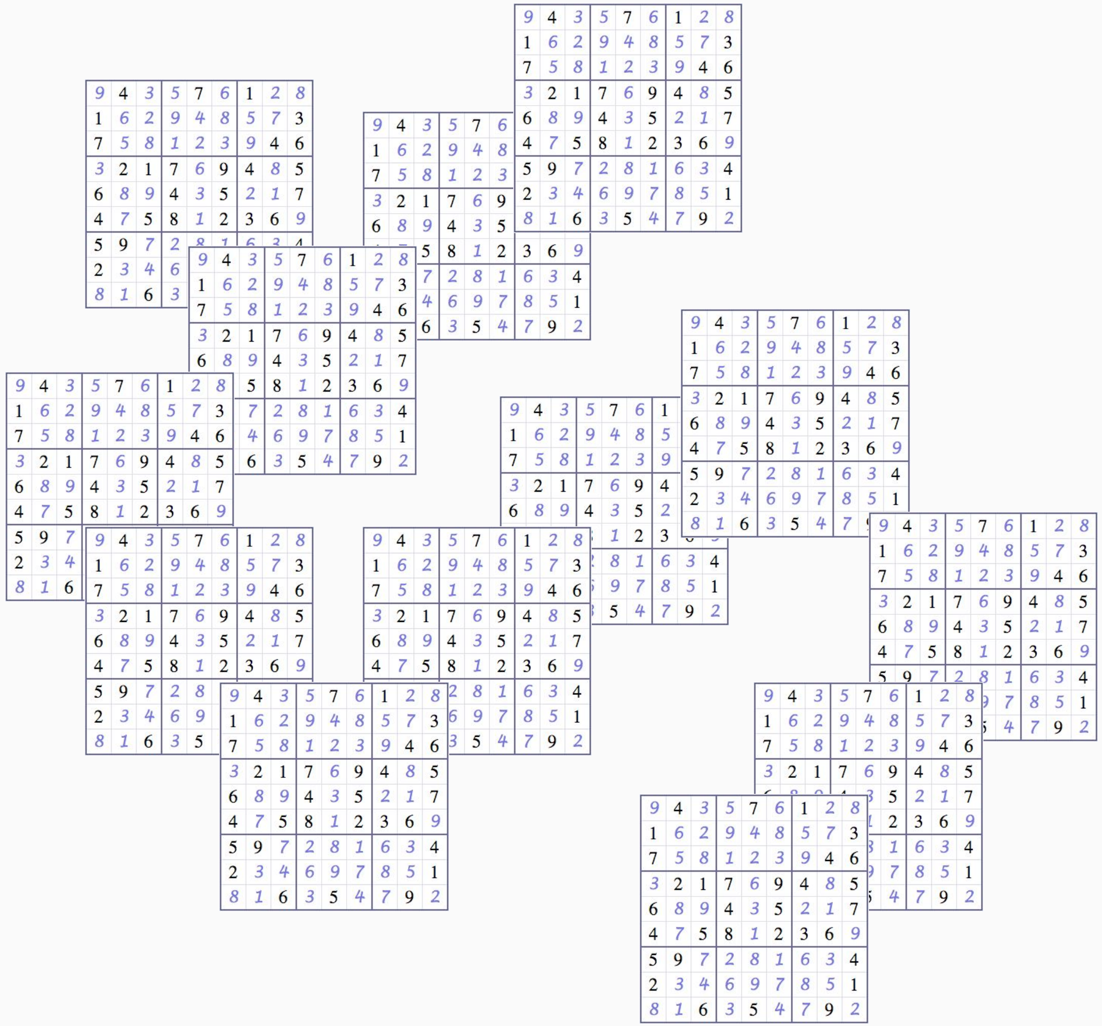
1. Computes the quality of entire food source (9x9 grid) by adding the uniqueness and quality of each element.
2. When entire solution is prepared, its quality is then compared with all other possible solutions to determine the “parent” solution.



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

Scout Bees (kernels/iteration)

1. Replaces the abandoned food with more randomly generated solutions.
2. Essentially, resets the values in each squares if a solution didn't pass a certain threshold of quality.
3. Scout bees are also triggered if tolerance becomes negative and the algorithm is stuck in a local minimum.



ABC: Performance Evaluation

	Easy	**Medium**	**Evil**	**Hardest**
ABC	3.387s	1.206s	19.85s	unsolved

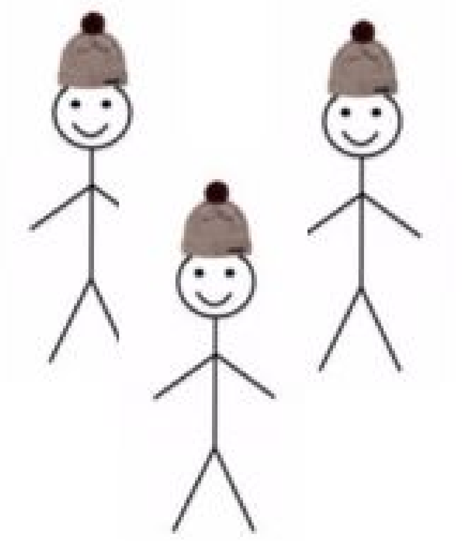
Muhammad likes to solve puzzles.

Angela likes to solve puzzles.

Shari likes to solve puzzles.

Muhammad, Angela, and Shari are smart.

Be like Muhammad, Shari, and Angela.



Sudoku Solver:
https://github.com/neoblizz/Sudoku