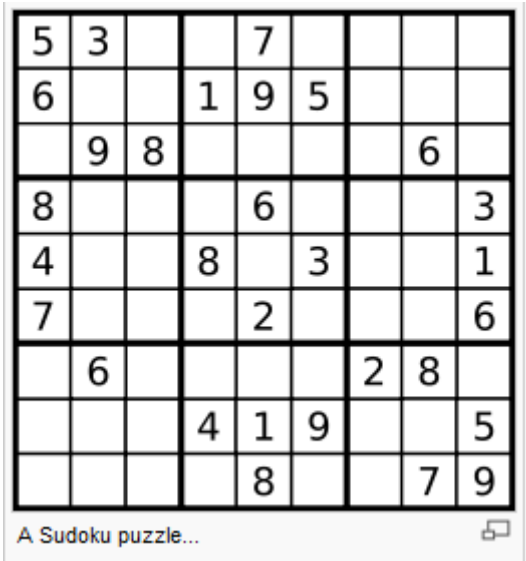
Solving Sudoku Puzzles with Particle Swarm Optimisation

ITEC808 - Workshop Slides

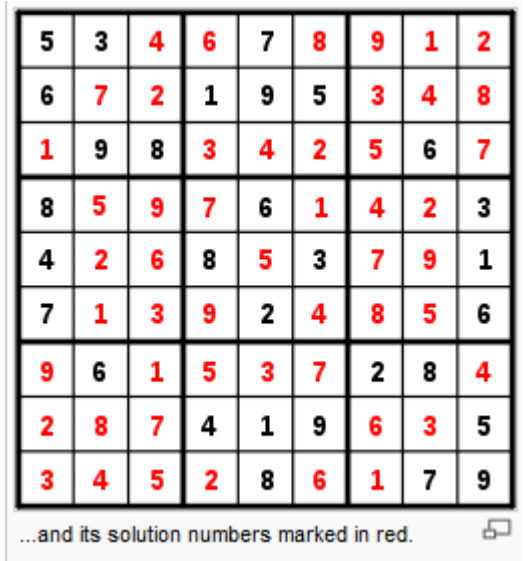
Sean McGerty – 89214617

Supervisor: Mehmet Orgun



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

A Sudoku puzzle...



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

...and its solution numbers marked in red.

http://en.wikipedia.org/wiki/Sudoku (2009.03.15)

Solving Sudoku Puzzles with Particle Swarm Optimisation

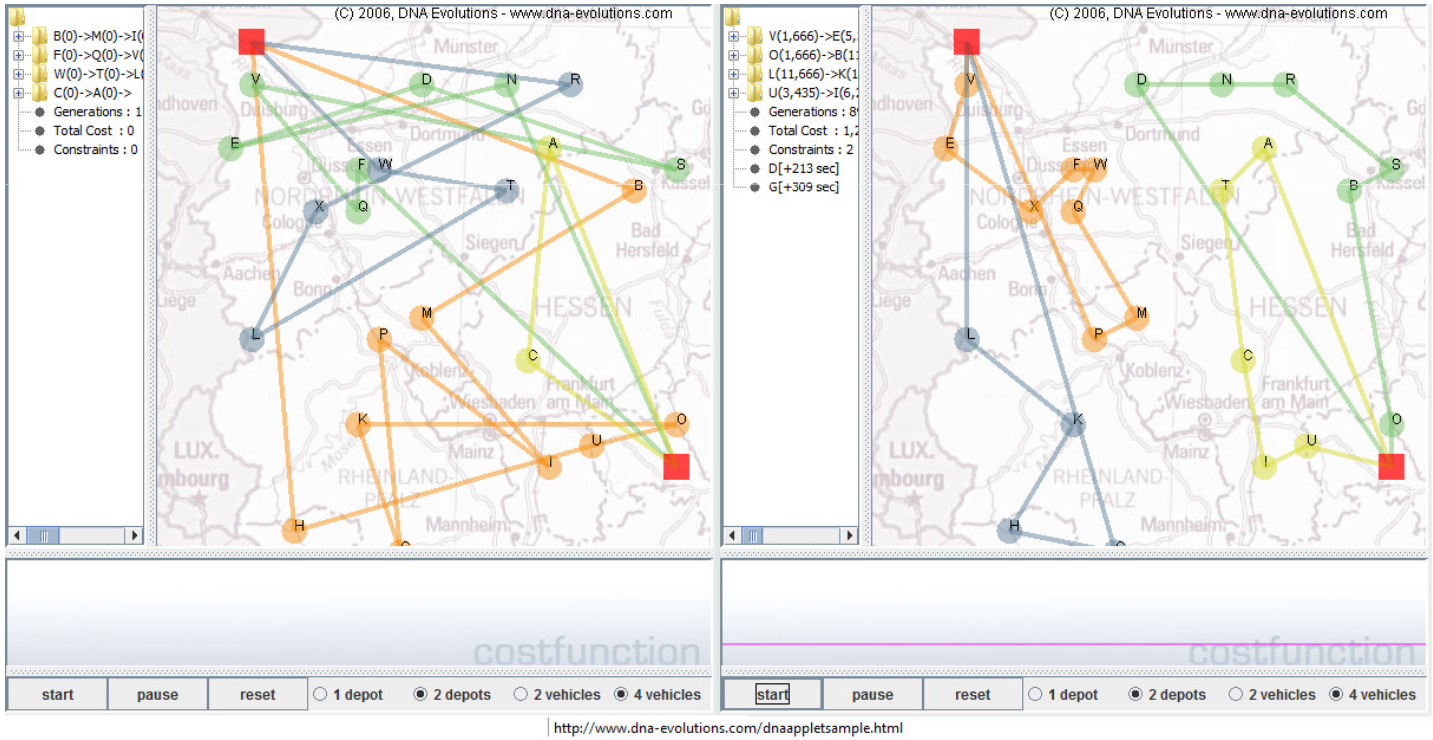
- Sudoku puzzles are complex problem spaces
  - $6{,}670 \times 10^{18}$ unique boards possible!
  - NPComplete – worst case exponential time order.

- Heuristics spread out and look for good solutions,
  - They learn by experience.
  - Can they find the best solution?

- Two main types:
  - Evolutionary(eg. Genetic) or
  - Swarming(our focus).

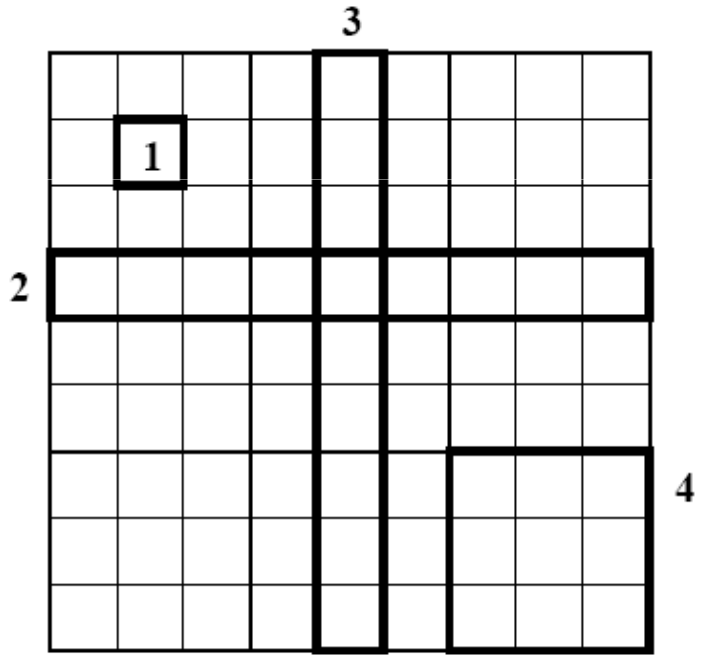
Solving Sudoku Puzzles with Particle Swarm Optimisation

- Heuristics do not have a preconceived idea of the puzzle. They randomly change and learn from each other, measuring success with a fitness function.



Solving Sudoku Puzzles with Particle Swarm Optimisation

- The Fitness function is the Heuristic's guide to comparing solutions. These are very solution specific. I suggest they are related to constraints.



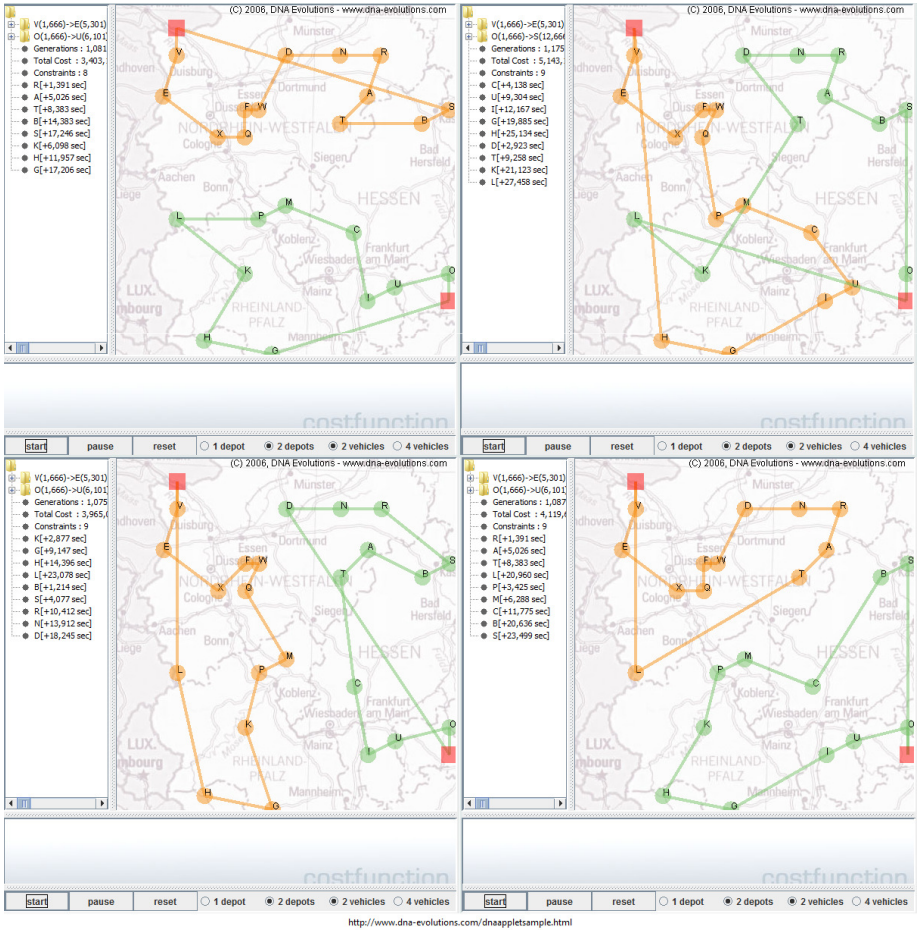
5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

...and its solution numbers marked in red.

Figure 3: Constraints on the Sudoku problem.

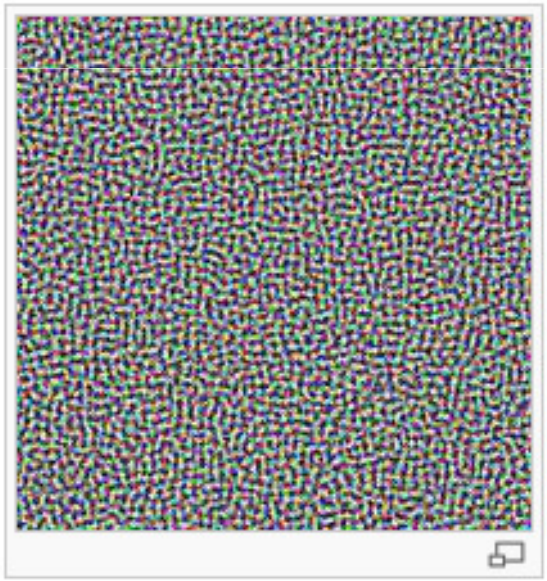
Solving Sudoku Puzzles with Particle Swarm Optimisation

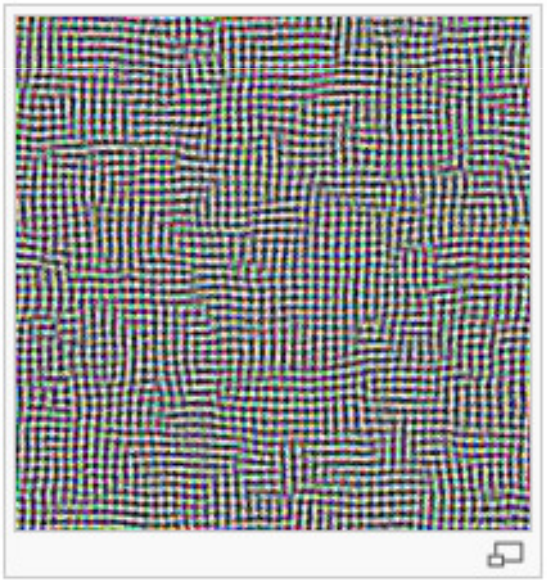
- The main issue we are trying to resolve with Heuristics is local maxima.
- Because Heuristics don't check all possible solutions they settle on a 'best so far' state. These are called local maxima.
- All our Heuristics will supply solutions, but our aim is to optimise them such that we get the complete Sudoku board solved.



Solving Sudoku Puzzles with Particle Swarm Optimisation

- Random factors spread the Heuristics to avoid clustering too early.
- For example the 'Temperature' of a Simulated Annealing Heuristic is cooled and random interactions are reduced.
- The Left example here was allowed to cool too fast.





http://en.wikipedia.org/wiki/Simulated_annealing

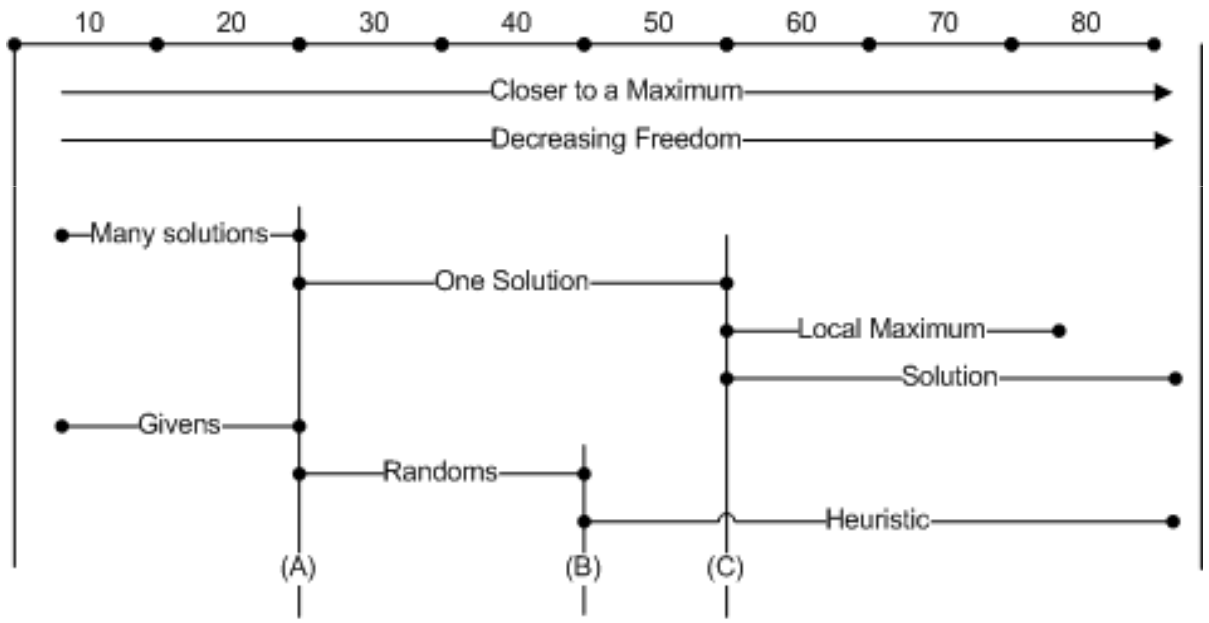
Solving Sudoku Puzzles with Particle Swarm Optimisation

- Optimised Swarming Heuristics show the best results
  - particles that flock
  - Trust in:
    - self,
    - neighbours,
    - randomisation.

- Local Maxima are a danger. Particles are spread by:
  - Spread them wide.
  - Stop them collecting.
    - Good results from Geometric Crossovers.

Solving Sudoku Puzzles with Particle Swarm Optimisation

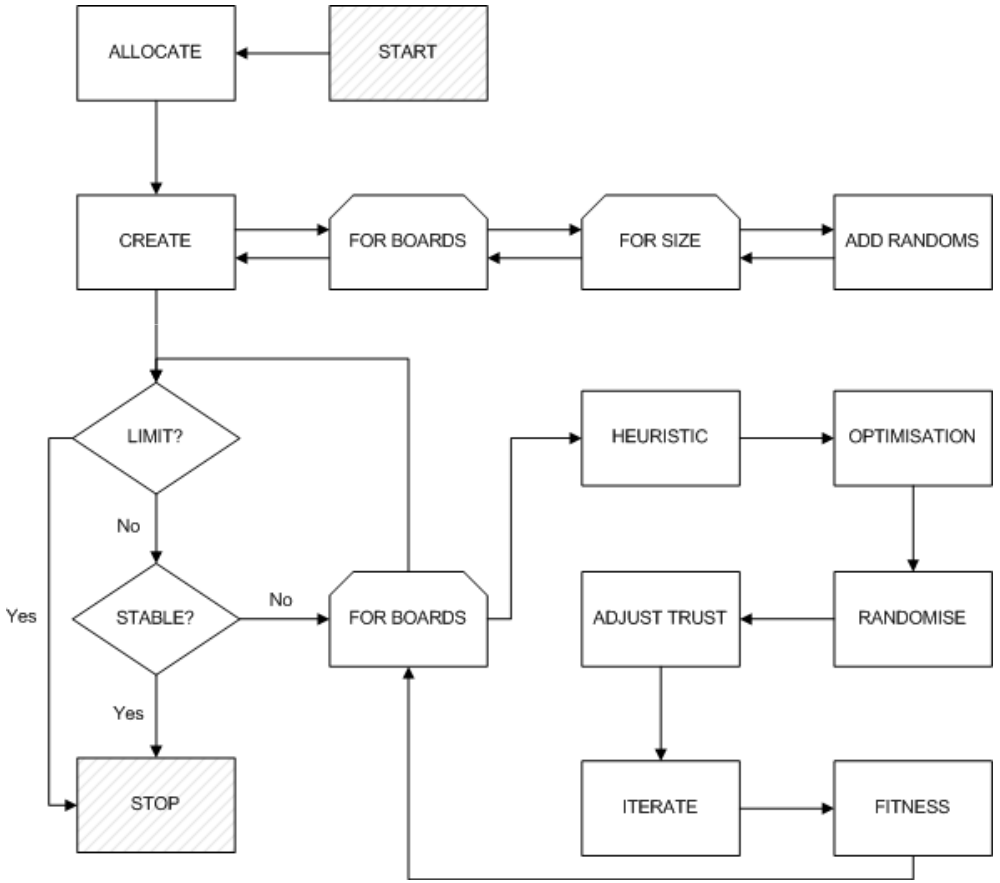
- In our Particle Swarm Example we spread our population of particles as far as possible at the start and then allow the Swarming behaviour to take over.



(A) – Puzzle masters attempt to restrict board to one solution.
(B) – Need to add Random numbers to distribute population – estimated.
(C.) – Need to start Heuristic processing before local maximum can't be avoided.

Solving Sudoku Puzzles with Particle Swarm Optimisation

- A Component Framework would work well
- Separates reusable trust factors
  - Lifecycle
  - Initialisation
- Better management and reporting
- Seperates Sudoku specific considerations from Heuristics.
- Allows much stronger comparison.



Solving Sudoku Puzzles with Particle Swarm Optimisation

- Our GPSO implementation is simple
- We initialise thousands of particles by randomly spreading them around.
- We join them together in social groups.
- Then until we stabilise we continue to try and improve the population with random change and reference to each other. This is the 'swarming' behaviour'.
- We also swap cells between solutions randomly. This is the Geometric Crossover which helps continue to spread the solutions around.

Algorithm 2 Geometric PSO algorithm

1: **for all** particle i **do**
2: initialise position x_i at random in the search space
3: **end for**
4: **while** stop criteria not met **do**
5: **for all** particle i **do**
6: set personal best $\hat{x}_i$ as best position found so far by the particle
7: set global best $\hat{g}$ as best position found so far by the whole swarm
8: **end for**
9: **for all** particle i **do**
10: update position using a randomized convex combination

$$x_i = CX((x_i, \omega), (\hat{g}, \phi_1), (\hat{x}_i, \phi_2)) \quad (3)$$

11: mutate x_i
12: **end for**
13: **end while**

Solving Sudoku Puzzles with Particle Swarm Optimisation

Questions?

Solving Sudoku Puzzles with Particle Swarm Optimisation

Thanks!