Explore Fitness Functions in GA for Solving Sudoku Puzzles

Group Members: Cao Yang, Chen Chen, Tang Yunhe, Wang Jiacheng

Content

- Introduction
- Analysis
- Design
- Evaluation

Introduction

- Goal: Explore influence that fitness function have on performance of genetic algorithm in solving Sudoku problem.
- Key word: Fitness function
- Experiment:
  - Propose several fitness functions
  - Apply them to solve Sudoku problem
  - Evaluate the results

Motivation

- Why GA?
  - GA has proposed for many years, not preferred for practice.
  - Reasons: 1. not suitable for certain problem

    2. computational complexity

    3. converge local optima
- Why Fitness Function?
  - Key to GA: To a large extend fitness function determine the converge rate and evolve direction.
- Why Sudoku?
  - Easy to implement and model, existed work on similar topic for comparison

Procedure for Solving Sudoku Puzzle

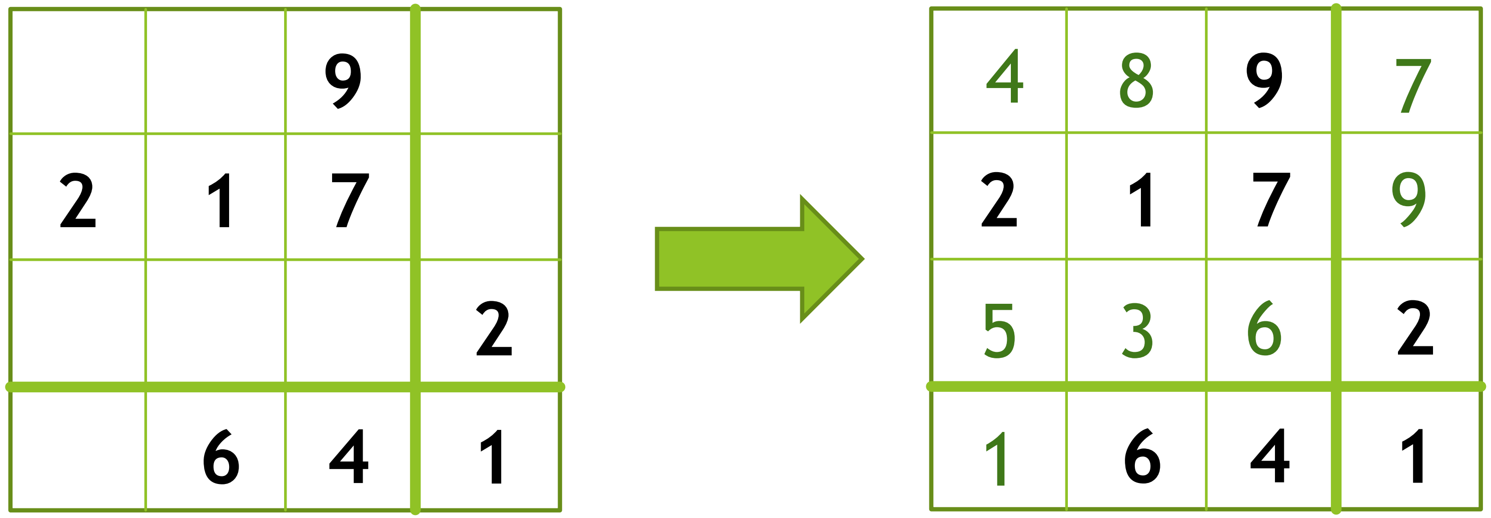
- **Initial settings**
- **While no one meets the criterion:**
  - Select 2 parent samples
  - Crossover
  - Mutation
  - Add 2 child samples into population
  - Eliminate 2 samples from population

Procedure for Solving Sudoku Puzzle

- **Some Experimental Parameters:**
  - Population size : 150
  - Number of child Candidates/Parents : 2
  - Crossover Rate : 0.3
  - Mutation Rate : 0.3
  - Tournament Size : 3

Initial Setting

- Satisfy the rule in sub-block : no more than one of any numeral

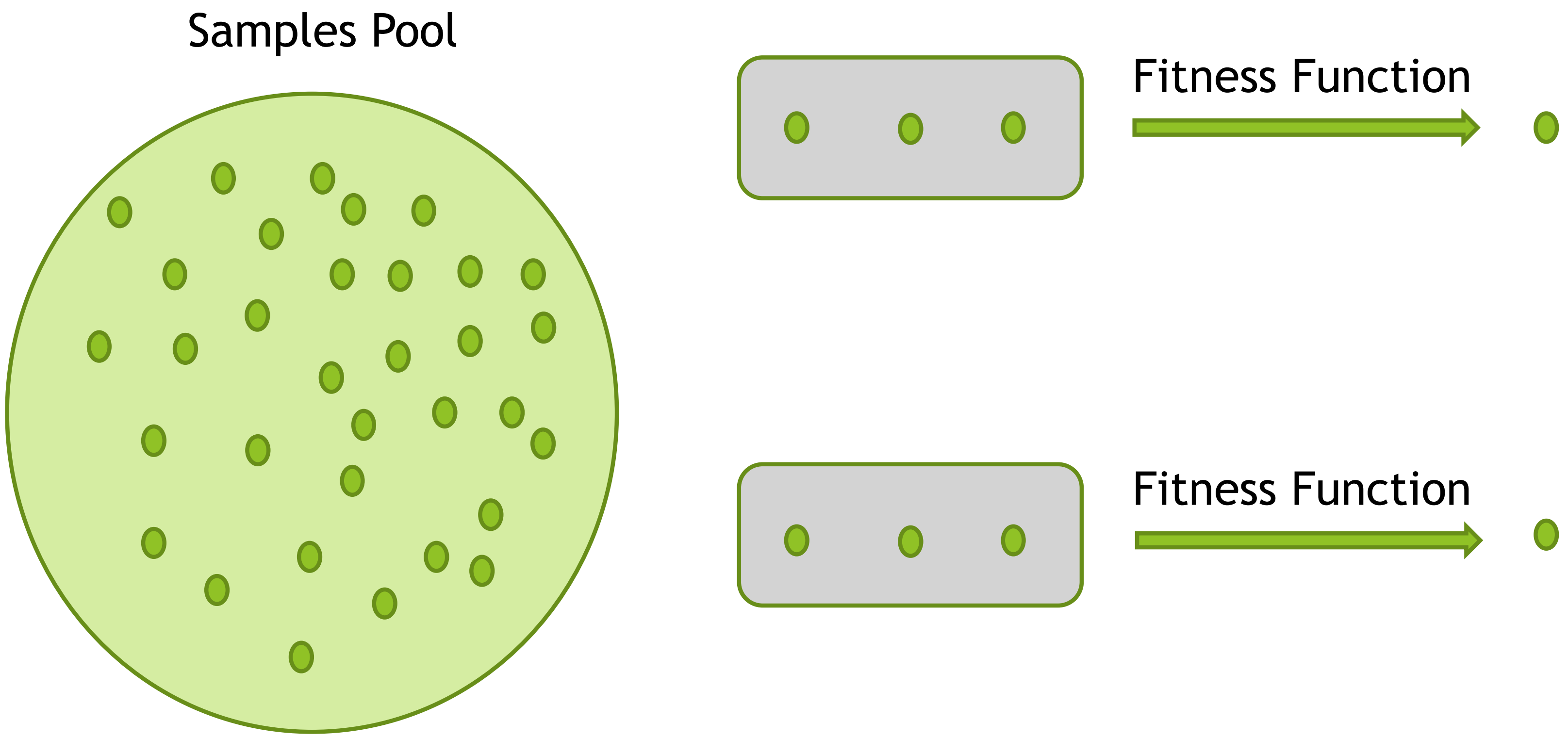


		9	
2	1	7	
			2
	6	4	1

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

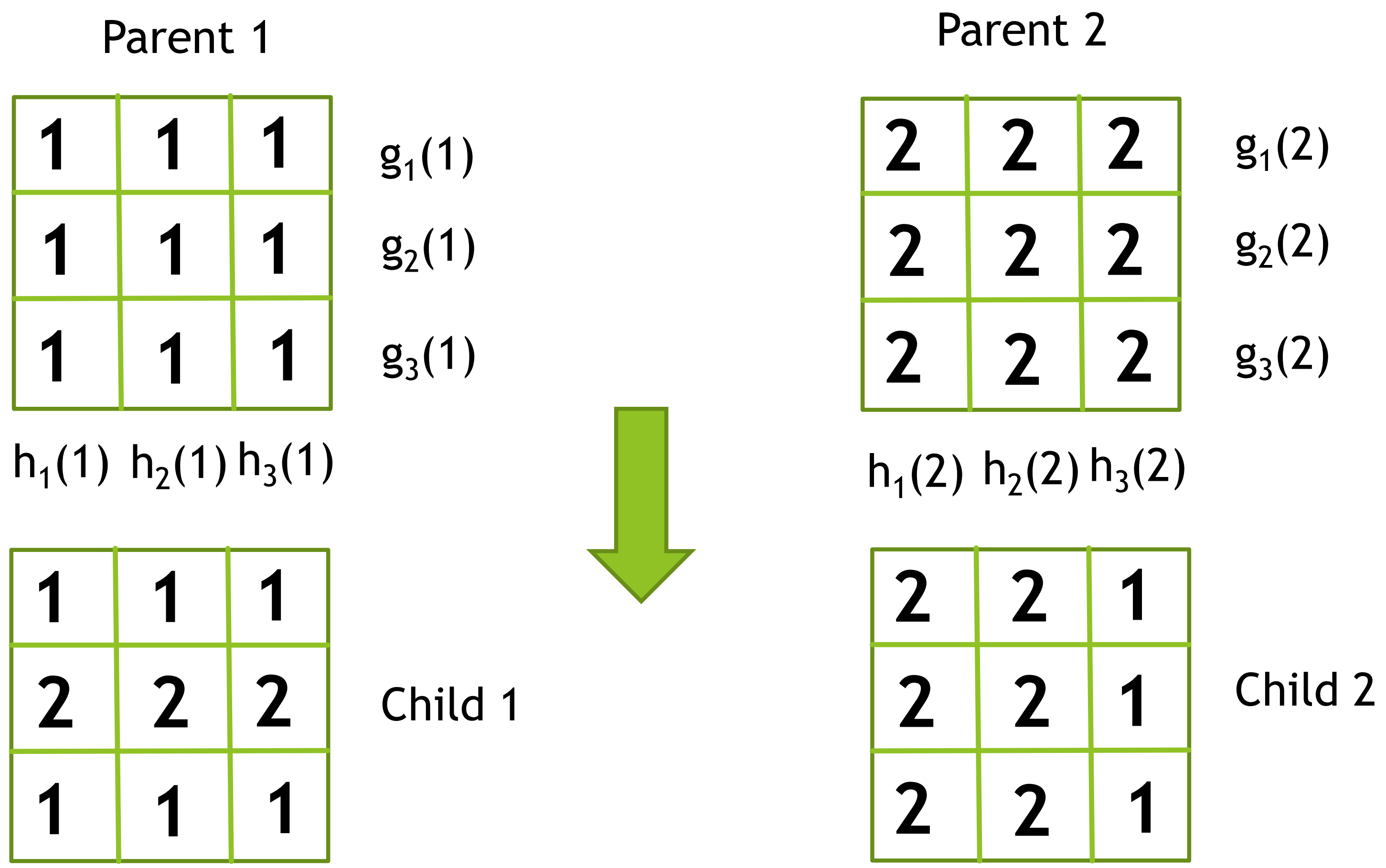
- 150 samples are generated

Parent Samples Selection



- Tournament size = 3; Parents = 2.

Crossover



Parent 1

1	1	1	$g_1(1)$
1	1	1	$g_2(1)$
1	1	1	$g_3(1)$
$h_1(1)$	$h_2(1)$	$h_3(1)$	

Parent 2

2	2	2	$g_1(2)$
2	2	2	$g_2(2)$
2	2	2	$g_3(2)$
$h_1(2)$	$h_2(2)$	$h_3(2)$	

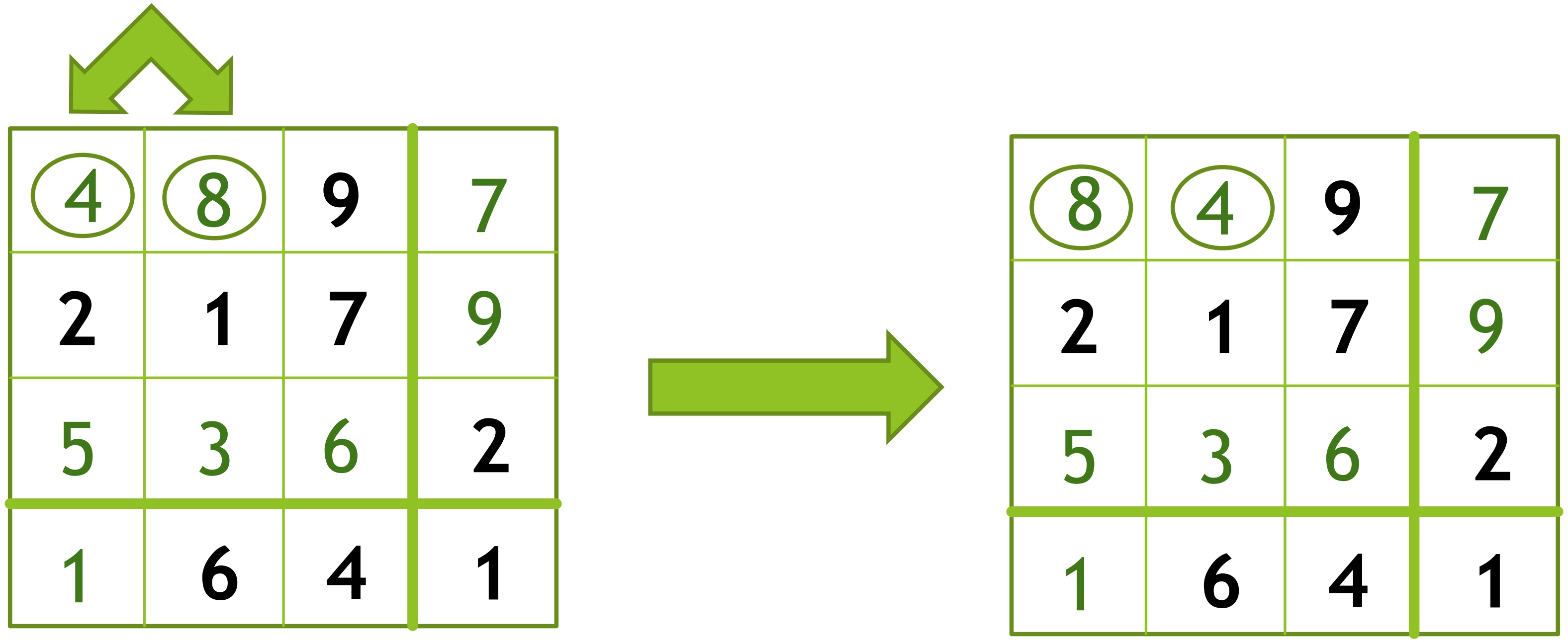
1	1	1	
2	2	2	Child 1
1	1	1	

2	2	1	
2	2	1	Child 2
2	2	1	

Crossover rate = 0.3

Suppose larger the fitness function value → better sample.
In this case:
$g_1(1) > g_1(2)$
$g_2(1) < g_2(2)$
…
$h_1(1) < h_1(2)$
$h_2(1) < h_2(2)$
…

Mutation

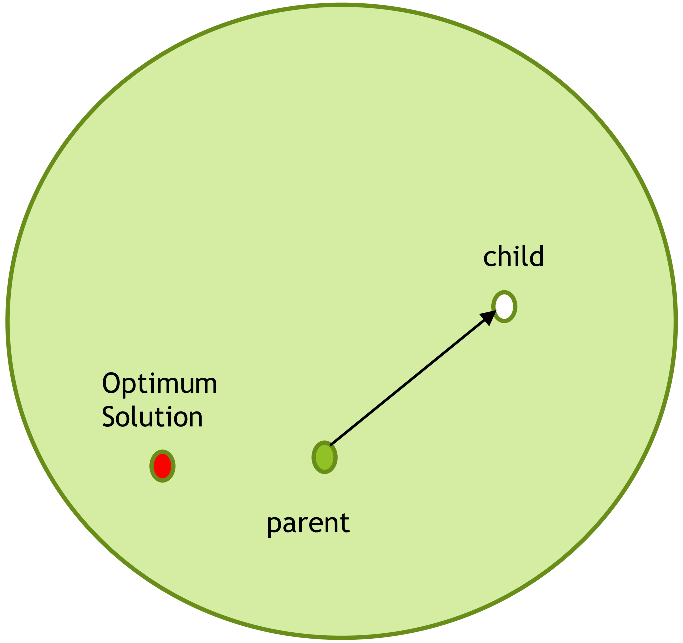


4	8	9	7
2	1	7	9
5	3	6	2
1	6	4	1

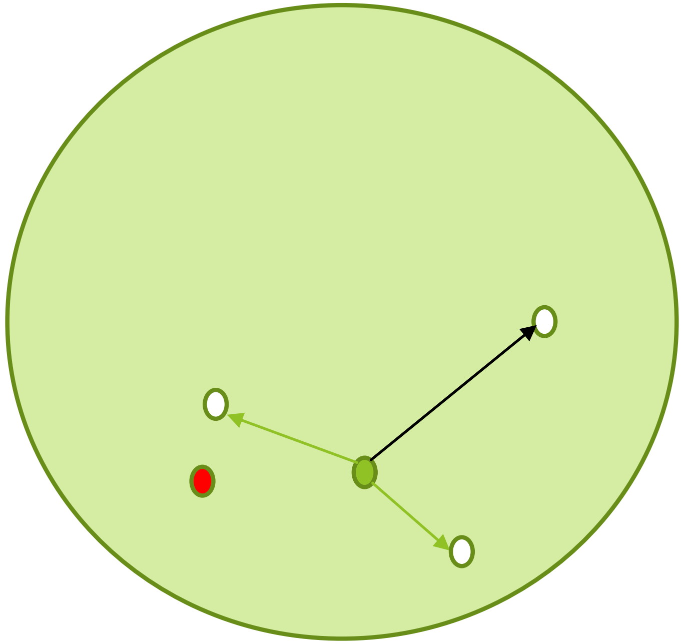
8	4	9	7
2	1	7	9
5	3	6	2
1	6	4	1

- Mutation rate = 0.3

More About Mutation



Parent : Child = 1 : 1



Parent : Child = 1 : 3
Then select best child

- Number of child candidates = 2

- Why we propose this fitness function?
- In order to compare the influence on the performance of GA.

Fitness Function #1

- Base on : The sum of each row or column is 45.
  - $f = \sum_{i=1}^{9} g_i(x) + \sum_{j=1}^{9} h_j(x),\ g_i(x) = \left|45 - \sum_{j=1}^{9} x_{ij}\right|,\ h_j(x) = \left|45 - \sum_{i=1}^{9} x_{ij}\right|$
- Weakness :

2	1	3	4
3	4	1	2
3	4	2	1
2	1	4	3

- Modification : assistance function h : number of conflicts in whole matrix
- F1 = a * f + b * h, where a + b = 1. If F1 == 0, solution is found.

Fitness Function #2

- Motivation : Instead of checking conflicts in whole matrix, we use another assistance function.
  - Base on : The factorial of each row or column is 9!.
  - $y = \sum_{i=1}^{9} g_i(x) + \sum_{j=1}^{9} h_j(x), g_i(x) = \left|9! - \prod_{j=1}^{9} x_{ij}\right|, h_j(x) = \left|9! - \prod_{i=1}^{9} x_{ij}\right|$
- F2 = a * f + b * y, where a + b = 1. If F2 == 0, solution is found.

Fitness Function #3

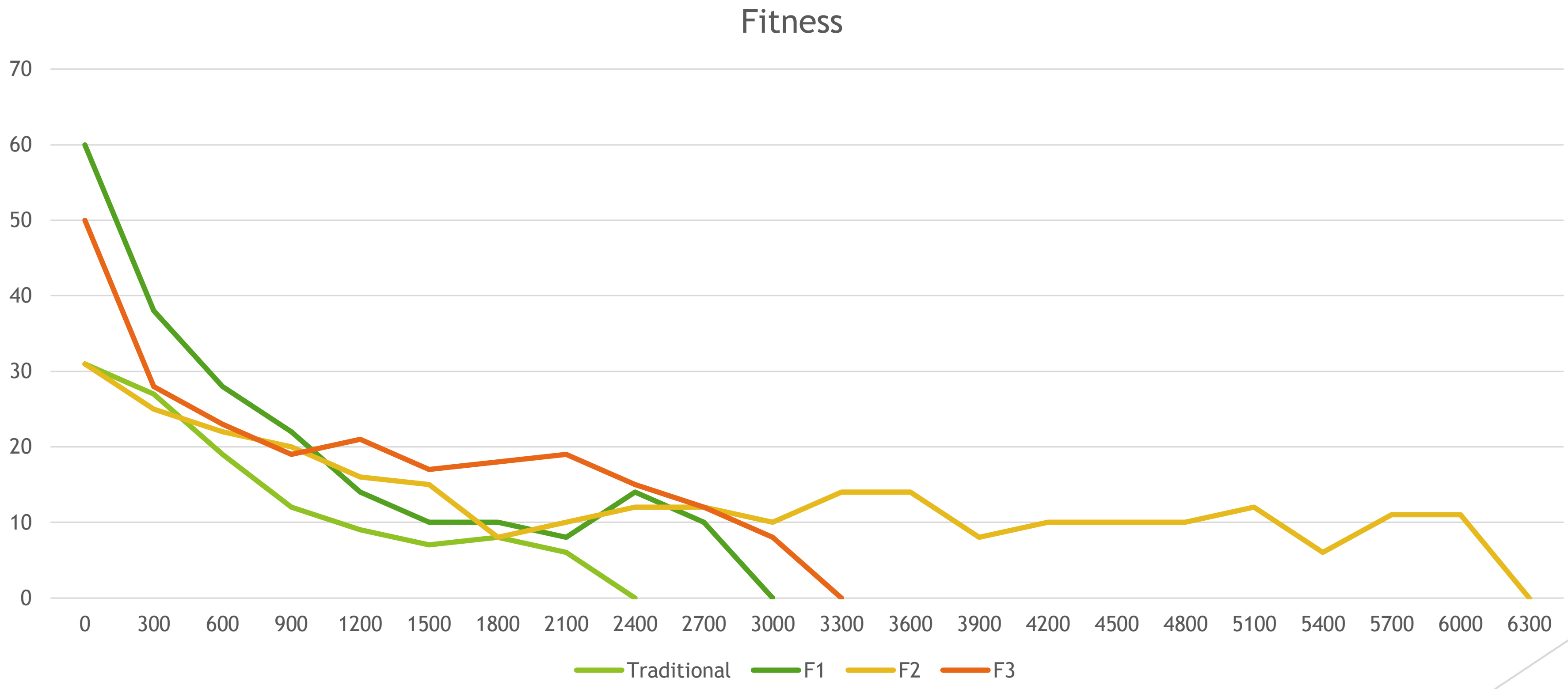
- Motivation : Consider each row or column as a vector.
- For optimization solution, if we sort each row, it’s a vector like:
  - <1, 2, 3, 4, 5, 6, 7, 8, 9>
- Calculate Euclidean distance by the Pythagorean formula:

Improvement

- **Problem A: Inefficient calculation of fitness function.**
  - cache the score, avoid repeated calculation.
- **Problem B: Stuck into local optima.**
  - Random elimination
- **Problem C: Early converge.**
  - Increase mutation to in increase diversity

Result

Fitness function	Count	Average	Time cost
Traditional function	20/20	2356	2.78
F1(SUM)	5/20	2842	3.17
F2(SUM&PRODUCT)	17/20	6362	8.37
F3(VECTOR)	18/20	3054	4.29


Fitness
70
60
50
40
30
20
10
0
0
300
600
900
1200
1500
1800
2100
2400
2700
3000
3300
3600
3900
4200
4500
4800
5100
5400
5700
6000
6300
Traditional
F1
F2
F3