Genetic Algorithm: Scheduling Problem- Sports League Fixtures

Gokul Anantha Narayanan, Krupashankar Sundararajan, Raghavan Renganathan

Final Draft: April 15, 2018

Problem

Implemented Genetic Algorithm (GA) to create a schedule of game fixtures in the league stage. Given, the number of teams and their names and locations, the algorithm creates a detailed schedule for the tournament. Each team plays every other team twice, once on their home ground and once on the opponent's home ground. Thus, if there are N teams, the number of matches will be N*(N-1). Genetic algorithm is used to provide maximum optimization to reduce the number of conflicts arising while creating the schedule. To make the algorithm more practical, some real-world constraints have been applied.

- Scheduling a match on a location with bad weather on that day must be avoided
- No team will play games on consecutive dates
- Two games cannot happen on the same day
- Each location will host N-1 games
- Each Team should play exactly 2 games with all the other teams.
- Each team should have played 2*(N-1) games.

Implementation Design

Genetic code: Each gene corresponds to one match fixture. The fixture contains information on the date, location, home team and away team.

Gene expression: A league having N teams will have N*(N-1) fixtures. Each schedule containing a list of fixtures is considered as an individual. The algorithm creates the most optimal schedule that satisfies the constraints with minimal conflicts.

Fitness function: The fitness of a schedule is dependent on the number of conflicts that arise during its creation. Conflicts are updated every time if any of the constraints are not satisfied. Constraint for weather is computed based on the weather index (Probability that the weather will be bad on that day at that location). Weather index greater than 80, will increase the conflict by 1. Since a high value of conflicts implies low fitness, the fitness is inversely proportional to the conflicts.

$$Fitness = \frac{1}{1 + conflicts}$$

Crossing Over: The crossover takes traits from both parents, in this case two schedules. The crossover is done by choosing the best fitting individuals to contribute with high probability. This is done by killing a part of the population which are not fit based on the **culling ratio (0.5)**. Then choosing individuals from the rest of the population to contribute to the cross-over.

This is done by choosing 2 individuals randomly from the survived population, for example,

Schedule A	A_1, A_2, A_3, A_4, A_5
Schedule B	B_1, B_2, B_3, B_4, B_5

Then picking a random cross-over point (Say 3),

Schedule A	A_1, A_2, A_3, A_4, A_5
Schedule B	$B_1, B_2, B_3, \uparrow B_4, B_5$

Cross-over point

Then interchanging all the genes before this point to form the offspring;

Schedule Offspring-1	B_1, B_2, B_3, A_4, A_5
Schedule Offspring-2	A_1, A_2, A_3, B_4, B_5

Mutation: A temporary schedule is created with randomly created fixtures and the fixtures of the schedule to be mutated are replaced randomly by the corresponding fixtures of this temporary schedule. The number of replacements can be increased/ decreased by altering the value of mutation rate.

Schedule A	A_1, A_2, A_3, A_4, A_5
Temporary Schedule B	B_1, B_2, B_3, B_4, B_5

Traverse schedule A and replace with elements from B at random. For example, replace A_2 with B_2 and A_4 with B_4. (Swaps chosen randomly)

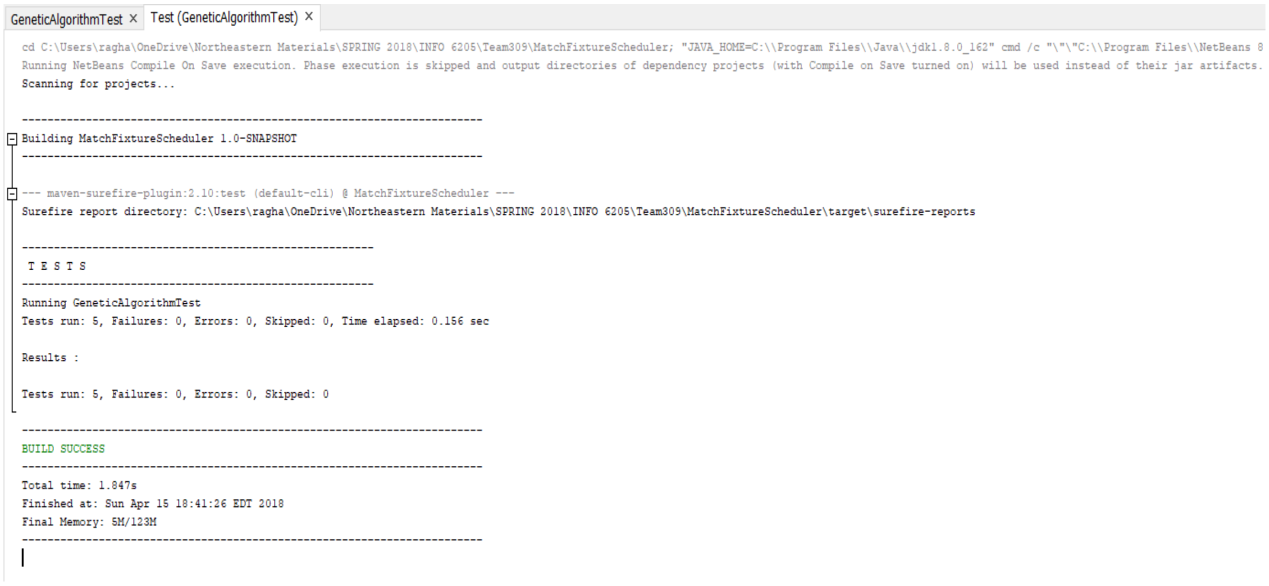
Mutated Schedule A	A_1, B_2, A_3, B_4, A_5

Evolution: The first step in evolution is to select the best individuals that are to be taken to the next generation. Two fittest individuals taken from samples of the population are crossed over to obtain a new individual. By taking individuals with better fitness, we make sure that the next generation has a better solution. Next, the elements of the individual are mutated to create the most optimal result.

Results

The test cases are written for testing if the fitness function and the algorithm is working correctly. In our case, the following cases are tested to validate our fitness function.

1. A schedule that satisfies all the criteria should produce 0 conflicts
2. Having multiple fixtures scheduled on a same day should generate a conflict for each violation
3. A team playing matches on consecutive days should increase the number ofconflicts for every violation
4. Each violation of “2 matches with each opponent” contract should increase the number off conflicts
5. Location hosting more than allowed games should create a conflict
6. Games scheduled on a bad weather (index > 80)
7. Each Team playing exactly 2*(N - 1) games, where N is the number of teams

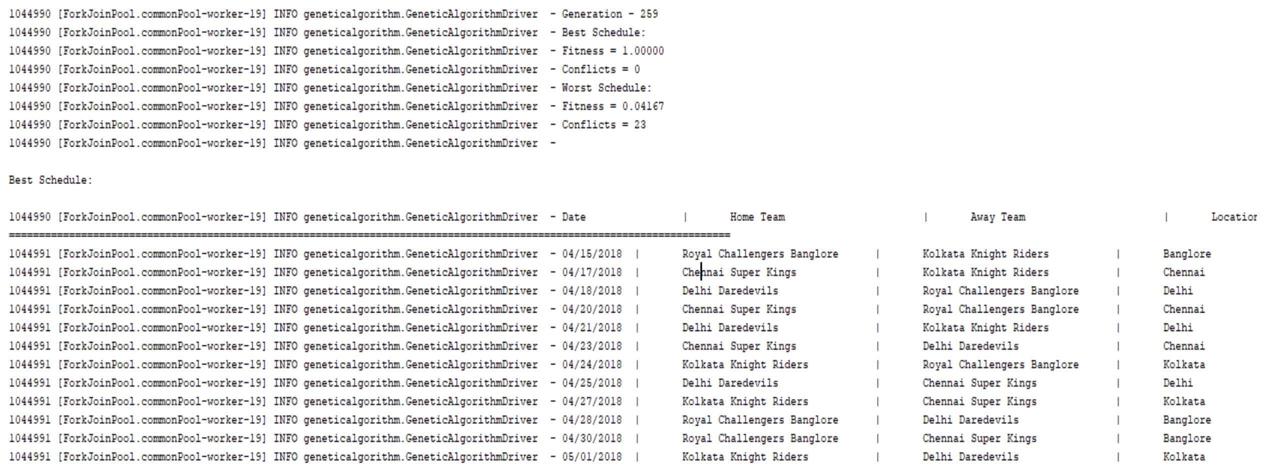


GeneticAlgorithmTest × | Test (GeneticAlgorithmTest) ×

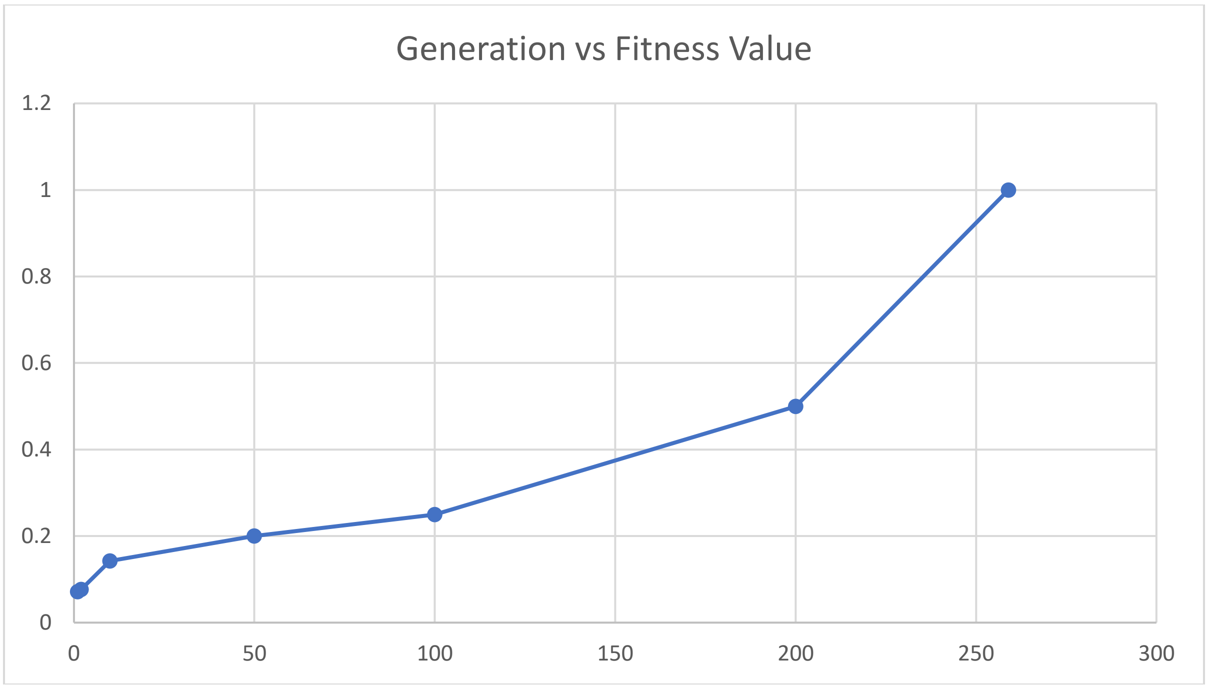
```
cd C:\Users\ragha\OneDrive\Northeastern Materials\SPRING 2018\INFO 6205\Team309\MatchFixtureScheduler; "JAVA_HOME=C:\\Program Files\\Java\\jdk1.8.0_162" cmd /c "\"\"C:\\Program Files\\NetBeans 8
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used instead of their jar artifacts.
Scanning for projects...

------------------------------------------------------------------------
Building MatchFixtureScheduler 1.0-SNAPSHOT
------------------------------------------------------------------------

--- maven-surefire-plugin:2.10:test (default-cli) @ MatchFixtureScheduler ---
Surefire report directory: C:\Users\ragha\OneDrive\Northeastern Materials\SPRING 2018\INFO 6205\Team309\MatchFixtureScheduler\target\surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running GeneticAlgorithmTest
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.156 sec

Results :

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 1.847s
Finished at: Sun Apr 15 18:41:26 EDT 2018
Final Memory: 5M/123M
------------------------------------------------------------------------
```

The no of conflicts and fitness value is logged for each generation. When the no. of conflicts reaches a minimum value of zero, the program terminates.

Generation	No of Conflicts	Fitness Value
1	13	0.07143
2	12	0.07692
10	6	0.14286
50	4	0.2
100	3	0.25
200	1	0.5
259	0	1



```
1044990 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - Generation - 259
1044990 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - Best Schedule:
1044990 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - Fitness = 1.00000
1044990 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - Conflicts = 0
1044990 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - Worst Schedule:
1044990 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - Fitness = 0.04167
1044990 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - Conflicts = 23
1044990 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  -

Best Schedule:

1044990 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - Date            |       Home Team                 |       Away Team                 |       Locatior
==================================================================================================================
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 04/15/2018  |      Royal Challengers Banglore      |      Kolkata Knight Riders           |      Banglore
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 04/17/2018  |      Chennai Super Kings             |      Kolkata Knight Riders           |      Chennai
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 04/18/2018  |      Delhi Daredevils                |      Royal Challengers Banglore      |      Delhi
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 04/20/2018  |      Chennai Super Kings             |      Royal Challengers Banglore      |      Chennai
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 04/21/2018  |      Delhi Daredevils                |      Kolkata Knight Riders           |      Delhi
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 04/23/2018  |      Chennai Super Kings             |      Delhi Daredevils                |      Chennai
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 04/24/2018  |      Kolkata Knight Riders           |      Royal Challengers Banglore      |      Kolkata
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 04/25/2018  |      Delhi Daredevils                |      Chennai Super Kings             |      Delhi
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 04/27/2018  |      Kolkata Knight Riders           |      Chennai Super Kings             |      Kolkata
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 04/28/2018  |      Royal Challengers Banglore      |      Delhi Daredevils                |      Banglore
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 04/30/2018  |      Royal Challengers Banglore      |      Chennai Super Kings             |      Banglore
1044991 [ForkJoinPool.commonPool-worker-19] INFO geneticalgorithm.GeneticAlgorithmDriver  - 05/01/2018  |      Kolkata Knight Riders           |      Delhi Daredevils                |      Kolkata
```



The above graph is plotted between the Generation value and the Fitness value. As we can see from the graph that the fitness value reaches 1 (i.e. no conflict) in the 259th generation.

The relationship between the no. of generations and the fitness value appear to be linear. Hence, it is proved that there is an improvement in adaptation at every generation.