**Football Fixture Using Genetic Algorithm**

1. **Introduction:**

Genetic algorithms is heuristic search algorithm that reflects the process of natural selection where the fittest individuals from the existing population are selected for reproduction in order to produce offspring of the next generation.

One of the application of Genetic Algorithm is **Scheduling** different events like University classes schedule, meeting fixtures schedule. We will study the use of GA for Football Fixture, which is an NP complete problem and can be easily solved using Genetic Algorithm.

1. **Problem Statement:**

Implemented Genetic Algorithm for Football Fixture problem. It is a scheduling problem which will create schedule for football fixtures of premier league. Given the team names, match locations and match dates, the algorithms creates a schedule for the tournament. Each team have their respective home ground. So, each team plays with each other teams twice. One match at their home ground and one at opponents home ground.

Therefore,

**Total number of matches in a tournament = (Number of teams) \* (Number of teams - 1)**

Following are used Hard Constrains for developing Fixture

1. Team cannot play with itself
2. Each team plays exactly 2 matches against EACH OTHER team in the league
3. Each team plays 2 \* (n-1) matches in the league
4. Each team plays one match at Home ground and one match at opponents home ground
5. Two matches cannot take place on the same day and same location
6. One team cannot play 2 matches on the same day
7. **Implementation Details:**

Basic Algorithm for GA:

1. Initialize generation = 0;
2. Create initial population

population[generation] = initialPopulation(populationSize);

1. Compute fitness for each individual

evaluatePopulation(population[generation]);

1. do{
2. Select the best fit individuals for reproduction

parents = selectParents(population[generation]);

1. Bread new individuals through crossover and mutation

population[generation + 1] = crossover(parents);

population[generation + 1] = mutate(population[generation + 1]);

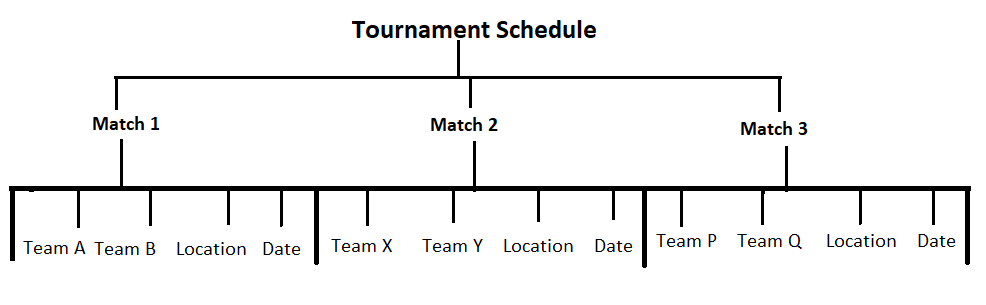
1. Evaluate fitness of new Individual

evaluatePopulation(population[generation] + 1);

1. Replace least fit population with new individuals
2. generation++;

} while( ! terminationCondition )

1. return the fittest indivisual



//need to write about Gene, chromosome