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.

* Locations having bad weather on particular dates has to 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 contains 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 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). Higher the weather index, worse the weather will be and hence, greater will be the increase in conflicts. Since a high value of conflicts implies low fitness, the fitness is inversely proportional to the conflicts.

Fitness=1/(1+conflicts)

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

Schedule A- A1, A2, A3, A4, A5

Temporary Schedule B- B1, B2, B3, B4, B5

Traverse schedule A and replace with elements from B at random. For example, replace A2 with B2 and A4 with B4. (Swaps chosen randomly)

The new schedule becomes A1, B2, A3, B4, A5

*Crossing Over:* The crossover takes traits from both parents, in this case two schedules. The logic for the crossover involves randomly choosing which parent’s gene is to be taken into the child.

Schedule A- A1, A2, A3, A4, A5

Schedule B- B1, B2, B3, B4, B5

The child schedule will contain fixtures from each parent. For example , if fixtures 1, 2 and 3 are chosen from parent A , and the rest from parent B, the new schedule will be A1, A2, A3, B4, B5

*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**

Test Cases:

1. Perfect Schedule with 0 conflicts
2. Multiple games on the same day
3. Team playing on several consecutive days
4. Team playing more than 2 games with a particular opponent
5. Location hosting more than allowed games
6. Games scheduled on a rainy day
7. Each Team playing exactly 2\*(N-1) games.

A screenshot of a social media post

Description generated with very high confidence

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 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

For the graph with 5 vertices, each trial produced the same cycle - 3, 1, 2, 0, 4. Since it is a small graph, this is most likely the actual solution to the problem for the graph generated. When rerunning the program with the corrected fitness function, the final fitness score changed but the same path was selected, consistent with this hypothesis.

One interesting effect is the size of the population on runtime and the solution. With a larger population the algorithm took longer to run, and even with culling the population size compounded, but with more individuals creating more opportunities for mutations and randomness, there was a positive relationship between population size, number of generations, and the final "most fit" score.

The relationship between number of survivors and run time appears to be logarithmic: