SET10117 – Emergent Computing for Optimization

**Approach**

My approach to this problem has been modelled on the solution to the knapsack problem presented in the tutorials due to the similarities between the two problems; individuals are made up of bits, there is a weight constraint (team budget), and the evaluation functions share some characteristics.

I used the eaSimple algorithm provided

Individual generation

Individuals are represented by an array of 523 bits. Each bit represents a player, if the bit is 1 then that player is in the team and vice versa.

Initially I wrote a brute force approach to generating individuals, which would generate individuals with 11 randomly flipped bits. Though this approach led to very quick individual generation, it yielded poor results as many of the individuals were invalid.

I decided to write the constraints into my individual generation, so that only valid individuals are generated. When generating a population this takes significantly longer for large populations (2000+) but yields much better results when it comes to the evolution stage.

To create valid individuals, I have two stages; the first stage of generation is about meeting the minimum requirements for the individual (1 goalkeeper, 3 midfielders, 3 defenders and 1 striker). These are randomly selected. To achieve random selection, I generate a random integer in the range 0-523, and if the number is below the threshold I set (in this case 8) then it considers the player for the team. This gives each player an 8/523 probability of consideration. I chose this because we need 11 players randomly spread across 523 spaces, and if players are too likely to be considered then the team is likely to be full before the players at the end of the list could be considered. If a player is considered for the team, then I have added if statements to check whether the initial player constraints have been satisfied. If there is room for them on the team then they are added. This process is completed using a for loop to go through the whole length of the individual, and that is inside a while loop that only breaks when the minimum requirements have been satisfied.

The second stage of individual generation is like the first, using the same consideration probability, for and while loops. It has different constraints, and randomly adds players to the team (using the same method for adhering to the constraints) and is inside a while loop that breaks when the team is full.

**Algorithm/operator design**

**Experimental design/analysis**

* Population experimentation
* Mutation probability experimentation
* Generational experimentation

**Solution Quality**

**Evaluation**