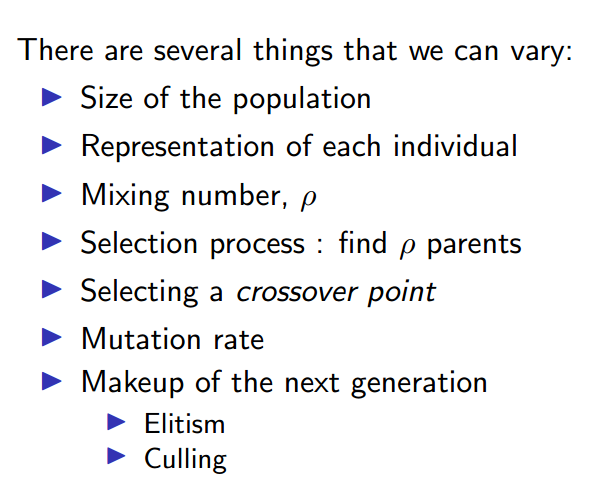
Ideas for Optimizing the Genetic Algorithm



1. Changes in the Mutation Function [IMPLEMENTED]

* Often it can happen that the mutation function will mutate an existing fit vertex. This is detrimental.
* To avoid this, while calculating the fitness function, we will also return the vertices which are actually unfit.

Possible Drawbacks:

* + Mutation could lead to more vertices becoming unfit than the current fitness level.

Possible Solutions:

1. Accept a mutation only if it increases the fitness of the current state
2. Elitism [IMPLEMENTED], Works brilliantly
   * Just the normal version of elitism.
   * Only ‘N’ number of fittest individuals will be allowed to reproduce, thereby reducing the number of unfit genes in gene pool for the next generation.
   * Also, Idea 1. Can be implemented along this, but by increasing the probability of mutation
   * ACTUAL IMPLEMENTATION
     1. Take the top 10 fittest parents from the previous generation into the next as well.
     2. Reproduce to find the remaining 40.
3. Hyper-parameter tuning

* The normal thing, try for various values of hyper-parameters, to check which set works best.
* Optimize for various things – such as time taken to train, number of generations to reach optimum
* These things can be considered as a cost/fitness function for a ML model that will run to train the hyperparameters to find least cost combination.

1. Mixing Number

* Important factor this one. More variation in the number of parents is beneficial. Like the advantages from asexual to sexual reproduction.
* Also, then rho, the mixing number will become a hyper parameter, opening up a chance for tuning using point 3.

1. Simulated Annealing
   * Decay the mutation probability
   * Start with higher MP then gradually decay with generations
   * UPDATE: We tried a basic version, but it does not seem to work really right now
2. Mutation Probability proportional to Number of Edges
   * Seems to work for some reason
3. Random Restart
   * When we get stuck because the cumulative sum of the fitness matrix of the previous generation is all zeroes, we can start the process again by randomly generating an extra generation. Just an idea, worth checking at least.