Evolutionary Computing Course Summary

We have two forces that can converge the Evolutionary algorithms:
1- Combination methods: Recombination and mutation
2- Selection methods

Combination methods

The combination methods are based on the type of representation, so we will list methods based on representation
First about the properties of the combination methods:

- Exploration
- For mutation:
    - Mutation rate: the chance of changes in a gene
    - Mutation Step (گام جهش)
- Recombination:
    - Recombination probability
    - Parents respect
    - Three type: (1) Structural (2) Statistical (3) Arithmetic

So now the representations are listed here and their specific combination methods

1. **Binary Representation**
    - One Parent: Mutation methods:
        - Bit flipping
    - Two Parents: CrossOver methods:
        - Single Point
        - Double Point
        - Uniform
    - More than two parents:
        - Population-wise uniform cross-over
        - Probabilistic uniform cross-over
2. **Integer Representation**
    - One Parent: Mutation methods
        - Random resetting
        - Creep
    - Two and More parents crossOver are the same before
3. **Permutation Representation**
    - One Parent: Mutation methods
        - Swap

        - Inverse
        - Scramble
    - Two Parents: CrossOver methods
        - Cut and Crossfill CrossOver
        - Order CrossOver
        - Cycle CrossOver
        - Edge CrossOver
    - More than two parents:
        - Not spoken in this course, but we can assume the probabilistic uniform crossover with a fringe set

4. **Floating Point Representation**
    - One Parent: Mutation methods
        - Uniform Mutation
            - Random resetting on a boundary
        - Non-uniform Mutation
            - Gene summation with a random value
            - The random value can be from any distribution
                - Gaussian with parameters (0, σ^2)
                - Polynomial
    - Two Parents: CrossOver methods
        - Arithmetic CrossOver
            - Can be Linear: $z = \alpha x_i + (1 - \alpha) y_i$
            - The methods are
                - Single Arithmetic CrossOver
                    - for one gene
                - Simple Arithmetic CrossOver
                    - From one point to the end
                - Whole Arithmetic CrossOver
                    - Whole genes (In other words the chromosome)
                - Blend CrossOver
                    - A random value between a boundary
                    - Distance between $d_i = |y_i - x_i|$ and boundary is $[x_i - \alpha d_i, y_i + \alpha d_i]$
                    - Note that x is assumed the minimum gene and y is the maximum gene comparing the gene of the parents
                - Binary simulated CrossOver
                    - It's good to have a discussion about it in class

5. **Tree Representation**
    - For non-planar genes
        - E.g.: Where the priority of genes are important such as calculator
    - One Parent: Mutation
        - Change a part of tree with another randomly generated tree

- Two Parents: CrossOver
    - Cut parents from one node and attach the splitted one to another one

Selection

Selection methods have two parts: (1) the probability functions (2) the Selection methods itself

(1) Probability functions:

Uniform

Fitness Proportionate

Ranking methods

- Linear Ranking
- Exponential Ranking

Boltzman probability

- It gives a selection probability even for worse fitnesses

(2) Selection methods:

Truncation

Fitness proportionate:

- Roulette wheel
- Stochastic Universal Selection (SUS)

Tournament Selection:

- Deterministic: truncate the best ones in each tournament
- Non-Deterministic: select the best one with probability of *p,* the second best one with probability of *p*1-p,* and the third best one with probability of *p*(1-p)*^2, …

Over-selection:

- Split all population into the better fitnesses and worse ones (2 or more splits) and select the population we need from both splits
- Each selection method introduced above can be used to select from the splits

Replacement

Replacement methods have a property that is:

(1) Generational

(2) Steady-State

To distinguish the properties we can use a metric which is the **generational gap.**

Generational gap definition: The rate of changes in population, which can be measured by the count of population changes.

For generational methods, the generational gap is set to μ and
For steady-state, the generational gap is set to the smallest value $1/\mu$

Diversity Preservation

Through the process of an evolutionary algorithm, Diversity or variety is a blessing. In other words, having many different solutions in our population can have two advantages **(1)** To avoid local optimum **(2)** For multi-modal problems in which different peaks in objective function landscape are our solutions we need diversity in the population.

Diversity preservation methods

1. **Fitness sharing**
   In this method, all fitnesses are updated based on the sparsity of the population.
   If the population was too dense the fitness sharing decrease the fitnesses or otherwise if the population was sparse the fitnesses will be increased.
   This method will compare each solution with all other solutions' distances.

2. **Crowding method**
   The idea here is to compare a solution with similar ones, not all as in the fitness sharing method. To find similar ones different metrics such as distance, label (labeling all the solutions), etc will be used.
   Note: In the process of evolution the labeled solutions can be evolved based on changes in their genes.
3. **Island Model**
   This method uses the idea of migration. After having multiple isolated populations that each use the evolutionary concept, It migrates some solutions after some iterations. By using this idea the diversity after some iterations will be increased therefore the diversity is preserved.
   Note: In this method, there are multiple hyperparameters to be set as the iteration count migration, number of solutions to migrate, what type of solutions to migrate (best, worst, random, …), and …
4. **Cellular Method**
   This method is based on distributed computing and each solution has some neighbors. It's similar to the crowding method with the difference of neighbors are assumed the similar chromosomes.
   In this method, a grid will be used in which each point is a solution and solutions will have recombination with just their neighbors.

Evaluation of EA algorithms

To evaluate and compare EA algorithms with each other, there are many metrics to use. Here we would list some.

Performance Metrics

To have a comparison between a variation of EA algorithms, there are some performance metrics that are listed below.

1. **Effectiveness:** The quality of solutions given the time and resources.
2. **Efficiency:** The computation resources and time needed to find a suitable answer.
3. **Success rate:** It measures Within the computation time and resource how many valid answers are produced.
   Note: An example of this metric is that in mutation if the answer was better then it is assumed that the mutation was a success.
4. **Robustness:** With one or multiple changes in problem configuration, how the EA algorithm performance changes.
   Examples can be scale-up (more population), configuration changes, etc.

Results comparison

To compare the solutions (fitnesses for example) for each generation or the result of multiple EA algorithm run, we could use statistical properties such as

- Minimum, Maximum, Mean, Variance, box plots and more
- Statistical tests that compare values distributions: ANOVA, t-test, p-test and …

Test Problems

We could also compare EA algorithms' configurations based on the problems available.

- **Benchmark Problems:** Problems that the solutions and objective function landscape is transparent to user
- **Problem Generator:** Generate a synthetic problem for algorithm to solve. Again everything may be available to the user.
- **Real World problems:** This cannot be a suitable way to evaluate different algorithms but is an available option. The disadvantages are **(1)** The results may not be publishable because of privacy and … **(2)** The objective function landscape is not fully observable.

Co-evolutionary Systems

Sometimes evaluating a solution's fitness individually is not possible. The reasons may be that we should consider the environment or other individuals in the population to evaluate one.

An example can be a football robot which it's fitness should be computed using the team's victory or defeat. So to evaluate one, we let the team play and if they won we could consider the number of goals and the fitness of the whole team (fitness can be averaged or all to be the same).

There are two types of these systems

1. **Co-operative co-evolutionary system:** in which a team will work together to achieve a same goal
2. **Competitive co-evolutionary system:** in which individuals don't have the same objective functions.

An example of each can be the football team, in which each team plays together to defeat the other team. (Teams are competitive and the inner team robots are co-operative)

To find individuals to evaluate ones fitness, we could randomly select an individual or use the similarity between them.

Hybrid EAs

The idea behind this type of algorithm is to use knowledge from out of the algorithm in the process of evolution.
There are two methods

1. *Lamark method:* Change the genes' value or structure based on the knowledge from outside
2. *Baldwin method:* Just change the interpretation of chromosomes based on the knowledge from outside. (for example change just the fitness values)

The knowledge from outside is called meme and to use them there are some ideas listed below

1. **The initial population:** Choose the initial population based on a meme and not randomly
2. **Use meme for recombination or mutation:** Select a special range for mutation values or use a special hyperparameter of specified recombination in the process of EA algorithm.
3. **Post-Process chromosomes:** Select a known value that is good for the solution and change the value of genes after mutation and recombination.
4. **Selection/Replacement method:** Use a kind of search instead of casual available methods.

Also for memes, we have three types of them:

1. **Static meme:** Static meme that is used in the process and has not been changed.
2. **Multiple memes:** Use multiple different memes for different parts of the evolutionary algorithm

3. **One/Multiple adaptive meme/s:** To use one or multiple memes that will be adapted and their values change in the process of the algorithm. An example would be the adaptation of the mutation rate.

Multi-Objective EA

In different problems there is not always one objective to be optimized. For them, there may be multiple objectives that are in contrast. (for similar objectives we could assume one in the optimization process)

In the evolutionary process first we need to define some preliminaries before diving into multi-objective solutions.
Pareto Dominance: If the fitness of a solution `x` was the same or better than `y` (at least there must be one objective that `x` is better) the we could say that `y` is dominated by `x`.
Pareto Optimal: `X` is named as Pareto optimal set iff it does dominate all other values in the decision space.
Pareto-Front: The fitness values of multiple functions for the set `X` introduced above.

Why EA is suitable for multi-objective problems?
1. It does have a population of multiple answers and multiple answers at the end of the algorithm process will be given.
2. It's assumed a good optimizer since it works better than random function in many problems.

Two items are evaluated for a pareto-optimal set:
1. **Convergence:** How the found pareto-optimal are near the real pareto-optimal set
2. **Spread:** how the spread of the pareto-optimal values are

There are many methods available to solve a multi-objective problem and we briefly list the summary of them below:
1. **Vector-Evaluated genetic algorithm (VEGA):** This algorithm assigns multiple solutions to each objective function and optimize them separately.
2. **Multi-Objective Genetic Algorithm (MOGA):** In this algorithm first the pareto-optimal set is found and then it updates the fitness of pareto-optimal based on the fitness sharing method.
3. **Non-Dominated Sorting GA II (NSGA-II):** After finding the pareto-optimal set, sort the values using their crowding distance.
   Crowding Distance: the distance of each individual based on its neighbors and normalized using the maximum - minimum fitness values in pareto-optimal set (sum of it for each dimension that the dimension is an objective function)
4. **Combine fitnesses using coefficients:** Use negative and positive coefficients based on the contrastive objective functions.

The idea of MOEA/D algorithm is that it uses multiple subproblems of single objective optimization by having different coefficients for each subproblem.

5. **Comparison with the worst or the best point in each objective space:** This gives the idea of computing volume or hypervolume, but it's computationally in-efficient and to elucidate that the HYPE algorithm is used to approximate the values.

Noisy, and Non-Stationary Problems

There are three type of noisy and non-stationary problems

1. **Noise in converting genotype to phenotype:** To solve this issue a population of solutions is used.
2. **Noise in calculating the fitness values:** We could calculate the fitness for multiple times and use an average value in order to reduce the noise effect.
3. **Dynamic Fitness function:** Parameter adaptation can be used here.

Another answer to all three type of problems above can be to an operator that uses uncertainty.

Constraint Problems

There are two types of constraints available for these problems that are

1. **Inactive constraints:** Constraints that does not affect the optimal solution of the problem.
2. **Active constraints:** Constraints that do affect the optimal solution. For example, having this kind of constraints can force the solution space to be changed (the solution before is not now a solution because it's not a feasible one)

To solve the constraint optimization problems 4 type of solutions are available

1. **Use constraint operators:** use operators in the optimization process that has attention to the constraints
2. **Convert constraint problem to an unconstraint one:** lagrangian methods or the count of satisfied constraint can be used for this type
3. **Use operators that have closure probabilities:** Satisfy all constraints in all generations' populations
4. **Use decoder or repairment method:** correct the chromosomes that have unsatisfied constraints
    - *Decoder:* Use an interpretation of feasible phenotypes for unfeasible genes
    - *Repairment:* convert genotypes to a near feasible genotype

Intractive EAs

For some problems the objective function cannot be specified because it's subjective. For this kind of problems a human in the loop is needed to give accurate feedback for each phenotype (Using human as the fitness function). An example of this kind of algorithms can be giving a loveliness score to a painting or to a song.

Using human can have cons and pros:

1. *Pros:* More accurate fitness values
2. *Cons:* (1) Less evaluation count can be done (Both population count and generation count must be low). Because human cannot answer many questions (2) High computation time and expensive

To solve the first problem we could learn the answers of a human and approximately predict them for future fitness requests instead of using him again (surrogate model).

Evolutionary Robotics

For evolution of robotics very briefly we give the items that should be consider in a same problem.

- How a robot could evolve? (1) morphologcaly (2) or its controller can be evolved
- Before the problem or while being in the problem? (1) Offline evolution or (2) Online evolution (for online evolution needed to be fast and give valid answers)
- Co-evolution of robotics (see example in `co-evolutionary systems` section)
- Simulating the robot or it's in real world?
- Centeralised or distributed evolution of robots? The evolution of population is in one system or each robot has its population
- Constraint, Noise, or dynamic?

Mainly the evolution of robots is time consuming because they have to be ran in an environment.