# Analysis of Genetic Algorithms for Feature Selection

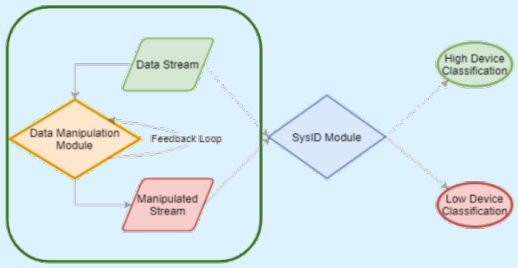
Christian Meinzen – Supervised by Nathan Chenette

## Introduction

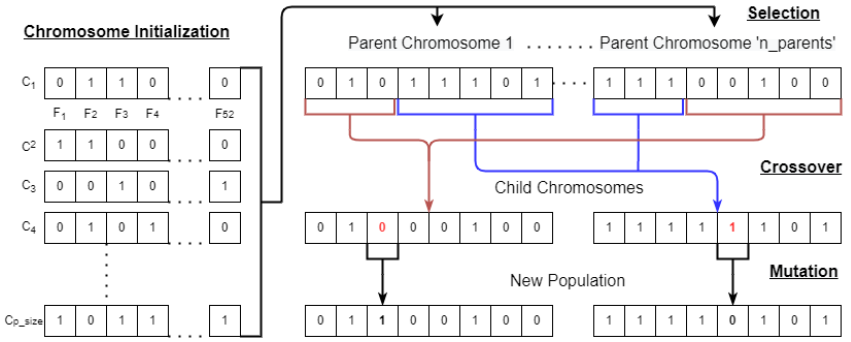
- The main component in successful system identifiers over a network is proper Feature Extraction
- Recent frameworks implement genetic algorithms (GAs)
- GAs have disadvantages of local convergence and slow runtimes
- My Goal: Analyze GA configuration for optimizing efficiency and effectiveness based upon fitness

## Original Intent and Design

- Original Goal: Design a manipulation module on packets causing a GA to be less effective in feature selection
- Challenges:
  - GA implementation was quite vague in previous studies
  - Determining effectiveness of the module was inconclusive due to overabundance of variables
  - Previous GA implementations were not optimized for runtime




## Genetic Algorithm Description




### GA Parameters

**p_size** = population size
**g_size** = generation size

**m_rate** = mutation rate
*(0 < m_rate < 1)*

**n_parents** = # of parents
*(n_parents < p_size)*

### Implemented Algorithm

**Algorithm 5:** Updated Genetic Algorithm

```
Data: p_size, n_parents, m_rate, g_size
Result: chromo, score
best_chromo, best_score ← [];
pop_nextgen ← initialize_population(p_size, 52);
for 0 < i < g_size do
    scores, pop_after_fit ← fitness_score(pop_nextgen);
    pop_after_cross ← crossover(selection(pop_after_fit, n_parents));
    population_nextgen ← mutation(pop_after_cross, m_rate);
    best_chromo ← pop_after_fit[0];
    best_score ← scores[0];
end
best_chromo, best_score;
```

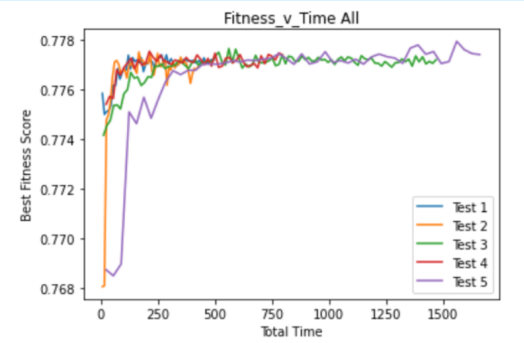
| | T 1 | T 2 | T 3 | T 4 | F |
|---|---|---|---|---|---|
| **p_size** | ? | 50 | 50 | 100 | 50 |
| **g_size** | 50 | ? | 50 | 50 | 100 |
| **m_rate** | .005 | .025 | ? | 0.01 | 0.25 |
| **n_parents** | 18 | 30 | 30 | ? | 35 |

## Experiment

- We ran the GA to extract a subset of 52 features based upon $F_1$ score of ML classifier (Decision Tree)
- Choose one of the four parameters to vary for each test (population size, generation size, mutation rate, # of parents)
  - Listed of the best and mean fitness score
  - Record the time required to complete all the generations
- Generate plots for two comparisons:
  - Fitness score vs current generation iteration
  - Fitness score vs cumulative time to reach iteration

## Results

- The number of parents should be at least half of the population
  - Fitness score was high at 0.7769, but the variance was the largest
  - Larger generation size leads to less variance but lower fitness of 0.7752
- Any additional computation past 612s (~ 10 min) becomes insignificant
  - All tests converge around a fitness of 0.775
  - Only few tests get higher, but it takes at least twice as long




## Moving Forward

- Avoiding local Convergence:
  - Implement **Hamming Distance** to compare feature set is consistent with population
  - Randomize the starting chromosome during initialization
- Implement other ML classifiers for $F_1$ score to ensure consistency