Population Based Methods

classical search and optimization methods:
- one solution in each iteration
- outcome is a single optimized solution

population based methods:
- use a population of solutions in each iteration
- outcome is also a population of solutions
Advantages of Populations

- if problem has single optimum: population members can be expected to converge to optimum
- if it has multiple optimal solutions: capture multiple optimal solutions in its final population

Population based methods are appropriate to solve multi-objective optimization problems
Evolutionary Algorithms

General scheme

BEGIN
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
  1 SELECT parents;
  2 RECOMBINE pairs of parents;
  3 MUTATE the resulting offspring;
  4 EVALUATE new candidates;
  5 SELECT individuals for the next generation;
END

Components of Evolutionary Algorithms

- Representation (definition if individuals)
- Fitness assignment
- Population
- Parent selection mechanism (Mating)
- Variation operators
- Survivor selection mechanism

- Genetic Algorithms
- Genetic Programming
  - (Evolution Strategy)
  - (Evolutionary Programming)
Genetic Algorithms (GA)

Representation of individuals as binary vector

<table>
<thead>
<tr>
<th>Genotype</th>
<th>Phenotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0 1 0</td>
<td>$26 = 2^4 + 2^3 + 2^1$ Integer</td>
</tr>
<tr>
<td>1 1 0 1 0</td>
<td>$6.69 = 2.5 + 26/31 (7.5 - 2.5)$ Real number</td>
</tr>
<tr>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>
GA - Mutation

Each bit is flipped with probability $p$.

Swap:

```
1 1 0 1 0
```

```
1 1 1 1 0
```

Scramble:

```
1 1 0 1 0
```

```
1 0 0 1 1
```

```
1 1 0 1 0
```

```
1 0 1 0 1
```

```
GA - Crossover

### One-Point Crossover

\[
\begin{array}{c}
1\ 1\ 0\ 1\ 0 \\
0\ 0\ 1\ 0\ 1
\end{array}
\rightarrow
\begin{array}{c}
1\ 1\ 0\ 0\ 1 \\
0\ 0\ 1\ 1\ 0
\end{array}
\]

### N-Point Crossover

\[
\begin{array}{c}
1\ 1\ 0\ 1\ 0 \\
0\ 0\ 1\ 0\ 1
\end{array}
\rightarrow
\begin{array}{c}
1\ 1\ 1\ 0\ 0 \\
0\ 0\ 0\ 1\ 1
\end{array}
\]

N=2

### Uniform Crossover

\[
\begin{array}{c}
1\ 1\ 0\ 1\ 0 \\
0\ 0\ 1\ 0\ 1
\end{array}
\rightarrow
\begin{array}{c}
1\ 0\ 0\ 1\ 1 \\
0\ 1\ 1\ 0\ 0
\end{array}
\]
GA – How can it work?

Mutation and crossover are quite arbitrary, how can it converge?

- If a bad solution is created
  - it will get eliminated in the next reproduction operation

- if the offspring is good
  - it is likely to get more copies in the next reproduction operation
  - it is likely to achieve a crossover with other good solutions
GA – Parent Selection

Fitness proportional selection

- An individual $i$ is selected for mating with probability

\[ p_i = \frac{F_i}{\sum_j F_j} \]

- Disadvantages:
  - Outstanding individuals take over entire population quickly
  - When fitness values are close together, almost no selection pressure

Ranking selection

- Sort population, $R_i = \text{rank of individual } i$
- $p_i \sim R_i$
Tournament Selection

- Relies only on ranking 2 individuals

Advantages

- Does not require knowledge of entire population
  - Distributed computation on parallel system
- There might be no suitable definition for the fitness
  - eg. game playing strategy
  - Compare them by simulating a game played with these strategies
The same methods used for parent selection can be used

Elitism
  - Prevent loss of the current fittest member
  - e.g. keep best 5% of the population (copy to next generation)
GA – Conflicting Goals

- **Exploration:**
  - Mutation, Crossover
  - Increase diversity (avoid getting stuck)
  - Too much exploration $\rightarrow$ random search

- **Exploitation:**
  - Selection operator
  - survival selection $\rightarrow$ converge fast
  - Risk of premature convergence
Genetic Programming (GP)

- Genetic Programming
  - Genetic algorithm applied to computer programs
  - Instead of a variable, a program represents procedure to solve the task

- Needs
  - Terminal set of constants and variables, e.g. $T = \{x, 1, 2, \ldots \}$
  - Function set of operators and basic functions, e.g. $F = \{ +, -, *, /, \exp, \sin, \ldots \}$

- Start with a set of randomly, correct programs

\[
y(x) = 2x^2 - 2x + 1 \quad \text{and} \quad y(x) = \exp(x) + 2
\]
GP – Mutation - Switch

```
add
  /  
mul   sub
     /  
   x   2
     /  
   x   1
```

```
add
  /  
mul   sub
     /  
   x   1
```
GP – Mutation - Shrink

![Mutation Diagram]

1. Original expression:
   \[ \text{add} \left( \text{mul} \left( 1, \text{x} \right), \text{sub} \left( 2, \text{x} \right) \right) \]

2. Mutation operation: Shrink

3. Mutation result:
   \[ \text{add} \left( 1, \text{div} \left( \text{x}, 3 \right) \right) \]
GP – Mutation - Grow

Initial function: \( \text{add} \left( 1, \frac{x}{3} \right) \)

Transformed function: \( \text{add} \left( \text{mul} \left( 2, x \right), \text{sub}(x, 1) \right) \)
GP - Crossover

Parent 1

```
add
  1
  mul
    2
    x
  sub
    x
    1
```

Parent 2

```
add
  2
  exp
    x
```

exchange
Thanks for your attention!