**A2: Optimization with Genetic Algorithms**

**The Job Shop Problem or the Job-Shop Scheduling Problem**

**Peana Florin Cosmin**

GitHub Repo:<https://github.com/FlorinCP/JSSP>

**1. Chromosome Representation and Problem Translation:**

A screenshot of a computer

Description automatically generated

A diagram of a job shop

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Selection Methods:**

**1. Tournament Selection:**

A screenshot of a computer game

Description automatically generated

**2. Roulette Wheel Selection:**

A diagram of a pie chart

Description automatically generated

**Mutation Methods:**

**1. Swap Mutation:**

A screenshot of a computer

Description automatically generated

A screenshot of a number

Description automatically generated

A screenshot of a number

Description automatically generated

A screenshot of a computer

Description automatically generated

**2. Inversion Mutation:**

Before mutation, with the indexes for inversion selected.

A screenshot of a computer screen

Description automatically generated

After mutation, the elements from the prev. selected indexes were reversed.

A screenshot of a computer

Description automatically generated

**Crossover Methods:**

**1. PPX (Precedence Preserving Crossover) [1 – Parent 1, 0 – Parent 2] [Convention]**

A screenshot of a computer

Description automatically generated

A screenshot of a number

Description automatically generated

A screenshot of a number grid

Description automatically generated

A screenshot of a number chart

Description automatically generated

**2. One-Point Crossover**

A diagram of a crossword puzzle

Description automatically generated

**Population Size and Convergence Detection:**

The population size is chosen based on practical considerations and theoretical guidelines:

* Default size of 100 provides a good balance between diversity and computational efficiency
* Large enough to maintain genetic diversity, out of my observations, smaller populations didn’t get the optimum result but some bigger ones performed way better but took a long time to compute.
* Small enough to converge in reasonable time as the training took some time even for some medium problems, as

The system identifies convergence through several mechanisms:

1. Tracking population diversity using **calculate\_diversity** method
2. Monitoring improvement rate in fitness values
3. Using a sophisticated convergence detection algorithm in **GAStatisticsAnalyzer**

The code uses an early stopping mechanism (MAX\_STAGNANT\_GENERATIONS\_STOP = 30) that triggers when:

* No improvement in best fitness for 30 generations
* Population diversity falls below a threshold
* Rate of improvement becomes negligible

Elitism is implemented with an **elite\_size** parameter (default 2), which preserves the best solutions across generations. This ensures that good solutions are not lost while still allowing for population evolution.

**2. Results**

* 1. **Instance FT06 [6x6]**

A screenshot of a game

Description automatically generated

A chart with different colored squares

Description automatically generated

A graph of a graph

Description automatically generated

**2.2 Instance ABZ8 [20x15]**

**A screenshot of a computer

Description automatically generated**

**A chart of a schedule

Description automatically generated with medium confidence**

**A graph showing a number of people

Description automatically generated**

**2.3 Instance LA34 [30x10]**

**A screenshot of a game

Description automatically generated**

**A schedule of a company

Description automatically generated with medium confidence**

**A graph showing a number of people

Description automatically generated with medium confidence**