# A Genetic Algorithm for Task Scheduling

Sean Strickland, Theisen Sanders

# Introduction

Goals

- Apply a genetic algorithm to a well-known problem
- Provide a mechanism to help others understand how genetic algorithms work

# Task Scheduling (I)

**Problem:** An optimization problem in which tasks $T_1$, $T_2$, …, $T_n$ are allocated to any number of processors.

**Goal:** To reduce the <u>total time</u> it takes to complete all the tasks and to minimize the <u>prioritized flow time</u>.

# Task Scheduling (II)

**Constraints:**

- Procedural
- Temporal

This is a *NP-Complete* problem so no optimal solution can be found in polynomial time when there are 3+ processors.

# Genetic Algorithms

Genetic algorithms are local search algorithms in which a population of solutions is evolved over generations to produce better solutions.

Key Components:

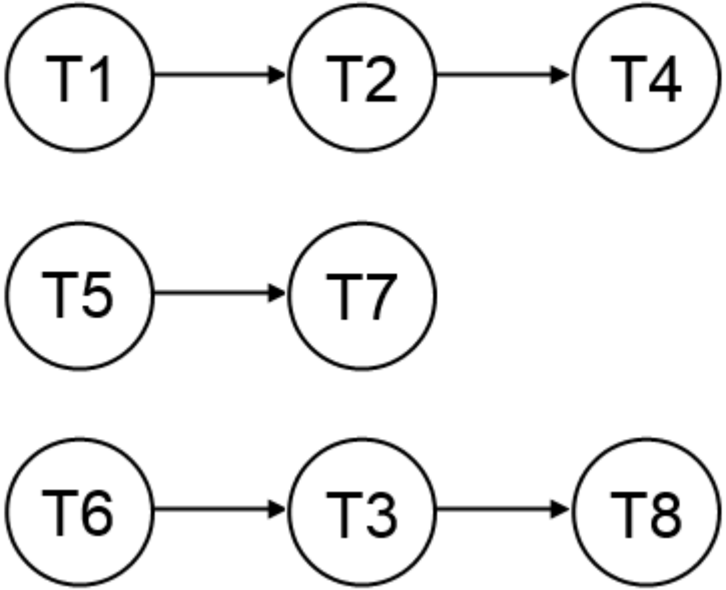
- Solution Encoding
- Fitness Function
- Crossover Function

# Our Algorithm (I)

**Initialization** - Use minimum completion time to generate initial population

**Fitness** - Weighted sum of total time and prioritized flow time

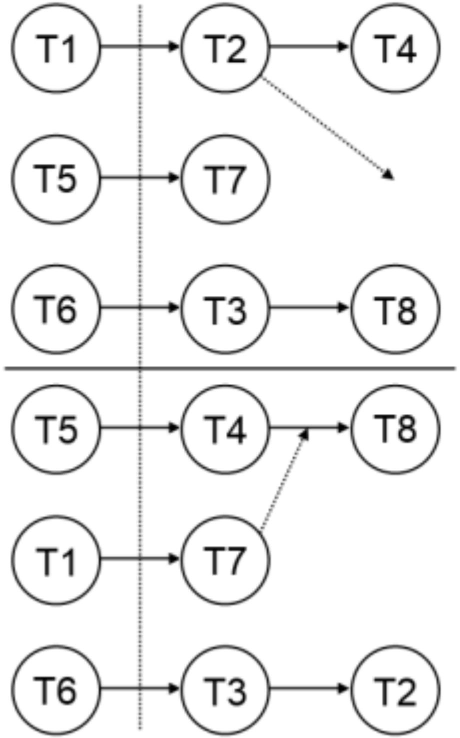
**Selection** - Roulette selection based on fitness values



# Our Algorithm (II)

**Crossover** - Randomly select crossover index to split each parent into two sections. Combine diagonally adjacent sections to produce two children.

**Mutation**- Randomly choose a task and move it to a random (dependency obeying) position in the schedule.



# Implementation

**Frontend:**

The constraints and tasks are created on the frontend using an AngularJS framework.

**Backend:**

The tasks/constraints are submitted to the algorithm written in Python and results are returned to the frontend for display.

# Demo

A Genetic Algorithm for Task Scheduling
Theisen Sanders
Sean Strickland
Documentation
Generations:
Processors:
Total Time:
Name
Priority
Length
Dependency
Add Task
Processor
Generate Schedule

# Results

| Test # | Tasks | Dependencies | Processors | MCT Total Time | MCT Flowtime | GEN* Total Time | GEN* Flowtime |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 0 | 2 | 5 | 25 | 4 | 23 |
| 2 | 6 | 3 | 2 | 16 | 211 | 15 | 201 |
| 3 | 10 | 5 | 3 | 16 | 433 | 14 | 417 |
| 4 | 15 | 8 | 3 | 17 | 495 | 15 | 455 |

* Run for 10 generations

# Conclusion

- Genetic algorithms can improve the solution obtained from basic task scheduling heuristics while having a minimal effect on performance.

- As with all genetic algorithms, performance and optimality of results are in the fine-tuning.

# References

Garey, M. R., D. S. Johnson, and R. Sethi. "The Complexity of Flowshop and Jobshop Scheduling." *Mathematics of Operations Research* 1.2 (1976): 117-29. Print.

Kaur, Kamaljit, Amit Chhabra, and Gurvinder Singh. "Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System." *International Journal of Computer Science and Security* 4.2 (2010): n. pag. Print.

# Questions?