

# GENETIC ALGORITHMS FOR ACTIVITY SCHEDULING

Fernandez, Ryan Austin

Poblete, Clarisse Felicia M.

## OUTLINE

1. Introduction
2. Formal Definition of the Algorithm
3. Analysis of the Algorithm
4. Applications
5. Conclusion

# INTRODUCTION

- Activity Scheduling is a problem
- Evident in professional organizations in universities
- Complex problem

## REVIEW OF RELATED LITERATURE

- Genetic Algorithms<sup>4,5</sup>
- Scheduling single set of exams for single set of students<sup>1</sup>
- Hashmap chromosome (timeslots to classes)<sup>3</sup>
- Two ply chromosome<sup>2</sup>

<sup>1</sup> Corne, D., Fang, H.L. & Mellish, C. (1993). Solving the Modular Exam Scheduling Problem with Genetic Algorithms. Technical Report 622, Department of Artificial Intelligence, University of Edinburgh.

<sup>2</sup> Omara, F.A. & Arafa, M.M. (2010). Genetic algorithms for task scheduling problem, Journal of Parallel and Distributed Computing, 70(1), 13-22, ISSN 0743-7315.

<sup>3</sup> Jankovic, M. (2008). Making a Class Schedule Using a Genetic Algorithm. Retrieved July 4, 2016, from Code Project: <http://www.codeproject.com/Articles/23111/Making-a-Class-Schedule-Using-a-Genetic-Algorithm>

<sup>4</sup> Mitchell, T. (1997). Machine learning. McGraw-Hill.

<sup>5</sup> Russell, S. & Norvig, P. (2010). Artificial Intelligence: A Modern Approach, 3rd Ed. New Jersey: Prentice-Hall.

## RESEARCH GAP

- None of the studies deal with a scheduling problem with tasks with target groups and venues with simple temporal constraints

## RESEARCH PROBLEM

- Can a genetic algorithm effectively generate schedules with the previously mentioned constraints?

## FORMAL DEFINITION OF THE ALGORITHM

### Genetic Algorithm

- Uses the concept of evolutionary genetics
- Answers come from sexual reproduction of previous candidates
- Can search complex problem spaces
- Can be programmed in parallel

# GENETIC ALGORITHM

- Main components:
  - Fitness
  - Chromosomes
    - Crossover
    - Mutation



## FITNESS

How desirable a solution is

**e.g. n-queens problem**

Fitness is reciprocal of number of queens attacking each other.

# CHROMOSOME

Encoding of a solution

Must have clear distinct parts e.g. bits, hashmap

**e.g. n-queens problem**

a chromosome could look like the row of each queen on each column (1 4 3 2)

# CHROMOSOME

## **Crossover:** breeding of solutions

- Crossover point

**e.g. n-queens problem**

breed n-queens solutions 4 | 3 2 and 2 4 3 |

result in 4 | 3 | and 2 4 3 2

## CHROMOSOME

**Mutation:** introducing randomness to a solution

**e.g. n-queens problem**

mutate n-queens solution by randomly swapping two queens

4 1 3 2 can mutate to 4 2 3 1

## THE ALGORITHM

```
population = randomize population
```

```
while !termination condition
```

```
    Create new empty population
```

```
    Carry over elite clones from old population
```

```
    Breed the remainder of the population
```

```
    Mutate random members of the population
```

```
    Get best chromosome
```

## THE ALGORITHM

- Termination Conditions
  - Until a fitness threshold is reached
  - Until a fixed number of generations is reached
  - Until the best solution has remained unchanged for a fixed amount of generations

# CHROMOSOME SELECTION

- Roulette Wheel
- $$\frac{c_{fitness}}{\sum_{x \in population} x_{fitness}}$$
- E.g. Fitness = [1,10,33,6] -> probabilities = [2%,20%,66%,12%]
- Tournament Selection
- Rank Selection

## ANALYSIS

- Commonly used for optimization problems
- Usually used for complex problems e.g. NP-class of problems
- GA's prefer certain schemata
  - E.g. n-queens solution that follow the  $12^{**}$  schemata are better than those with a  $4^{*}4^{*}$  schemata
- Crowding



## VS. NEURAL NETWORKS

- Neural Networks move from one hypothesis to another slowly
- GA may move quickly from one hypothesis to another

## APPLICATION (ACTIVITY SCHEDULING)

- **Fitness** =  $1 / (\text{punishment} + 1)$
- **Punishment** = sum of all conflicts
  - For each date and time conflict regardless of target group, add 2 punishment points.
  - For each date and time conflict with intersecting target groups, add 7 punishment points.
  - For each date and time conflict with same venue, add 10 punishment points.
  - For each pair of activities where one activity takes place after the other on the same day, add 3 punishment points.
- Punishment stacks up

# CHROMOSOME DESIGN

	Chromosome 1	Chromosome 2
Act 1	08/12/16 14:00:00	08/19/16 13:00:00
Act 2	09/01/16 12:30:00	09/08/16 12:45:00
Act 3	08/15/16 08:00:00	09/01/16 10:45:00
Act 4	09/03/16 11:30:00	09/05/16 12:00:00

# CHROMOSOME DESIGN

	Chromosome 1	Chromosome 2
Act 1	08/12/16 14:00:00	08/19/16 13:00:00
Act 2	09/01/16 12:30:00	09/08/16 12:45:00
Act 3	08/15/16 08:00:00	09/01/16 10:45:00
Act 4	09/03/16 11:30:00	09/05/16 12:00:00

## CHROMOSOME DESIGN

	Chromosome 1	Chromosome 2
Act 1	08/12/16 14:00:00	08/19/16 13:00:00
Act 2	09/01/16 12:30:00	09/08/16 12:45:00
Act 3	09/01/16 10:45:00	08/15/16 08:00:00
Act 4	09/05/16 12:00:00	09/03/16 11:30:00

## CHROMOSOME DESIGN

### **Mutation**

- For each activity, with probability 0.7, randomize the schedule of that activity
- Choose a random date
- Choose a random timeslot

## CHROMOSOME DESIGN

### **Algorithm Parameters**

- Population Size: 50
- Fitness Threshold: 0.14
- Elitism Rate: 0.2
- Mutation Rate: 0.4
- 1000 iterations

## RESULTS AND ANALYSIS

- Satisfactory
- Threshold allowed few acceptable
- GOSM for LSCS as test script
- Performed well with 22 activities



## CONCLUSION

- Genetic Algorithm performs well in this context
- Implementation of the data model, algorithm, and system were successful
- It is recommended to add more capabilities and constraints to the data model to better represent real life situations in scheduling

## REFERENCES

- Corne, D., Fang, H.L. & Mellish, C. (1993). Solving the Modular Exam Scheduling Problem with Genetic Algorithms. Technical Report 622, Department of Artificial Intelligence, University of Edinburgh.
- Czarn, A., et al. (2004). Statistical Exploratory Analysis of Genetic Algorithms. *IEEE Transactions on Evolutionary Computation*, 8(4), 405-421.
- Omara, F.A. & Arafa, M.M. (2010). Genetic algorithms for task scheduling problem, *Journal of Parallel and Distributed Computing*, 70(1), 13-22, ISSN 0743-7315.
- Jankovic, M. (2008). Making a Class Schedule Using a Genetic Algorithm. Retrieved July 4, 2016, from Code Project: <http://www.codeproject.com/Articles/23111/Making-a-Class-Schedule-Using-a-Genetic-Algorithm>
- Mitchell, T. (1997). *Machine learning*. McGraw-Hill.
- Russell, S. & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*, 3rd Ed. New Jersey: Prentice-Hall.
- Sadegheih, A. (2006). Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on GA performance, *Applied Mathematical Modelling*, 30(2), 147-154, ISSN 0307-904X.
- Wall, M.B. (1996). *A Genetic Algorithm for Resource-Constrained Scheduling*. Massachusetts Institute of Technology.