A Genetic Algorithm for Task Scheduling

By Sean Strickland, Theisen Sanders

Artificial Intelligence - Computer Science 472 - Iowa State University

Links

- Web Application
- Code Repository

Project Description

Our project idea is to develop a genetic algorithm for task scheduling. A general overview is that we would like to make a genetic algorithm that can take a large set of tasks, some constraints, a number of resources to run the tasks on, and outputs a schedule for running the tasks on the resources that efficiently uses time and resources while not violating any hard constraints. One possible use case for this is as a scheduler for a group of servers pulling tasks from a task pool.

A more concrete description is that each task could have a duration (amount of time it takes to execute), a priority value, and a set of prerequisite tasks (dependencies) that must be completed before this task can be executed. The input is a set of these tasks and the number of resources available to execute the tasks. The output is a list of resources and the order in which tasks are run on each resource. Additionally, users could set a number of generations they want the algorithm to run for (a larger number of generations yields a better result). The tricky part of this project will be developing a genome representation of a schedule and a fitness function that can accurately determine the efficiency of a given genome/schedule.

These details are flexible at the moment and there are many ways we could add to or change the properties of a task. We would like to make a web application that allows a user to construct the set of tasks and provide other inputs like a maximum running time, maximum number of generations, and/or a "seed sequence" that could be used to influence what the result might look like. The goal being to help others visualize and understand how genetic algorithms work.

Goals

- To develop an efficient algorithm for determining an optimized task schedule
- To create a web application that provides simple inputs for tasks and constraints, and outputs the resulting schedule
- Provides a mechanism for helping users understand how the genetic algorithm operates

Approach

Algorithm

- Inputs - The tasks to schedule, constraints on the possible schedules, and algorithm specific inputs such as max generations, timeout, and/or seed.

- Constraints
    - Hard (if genome does not satisfy, it cannot reproduce)
        - Procedural - The dependencies each task has on other tasks. If task T1 is dependent on task T2, then this constraint prevents T1 from being scheduled before T2 is complete.
        - Temporal - The time constraints put on each task, such as task T1 must be completed before time 10.
        - Resource - Analogous to the number of processors available to run tasks. If there are X resources/processors available then up to X tasks can be executing at any time.
    - Soft (genomes that best satisfy these constraints will be more likely to reproduce)
        - Total Time - The time it takes for the longest running processor to finish executing tasks.
        - Prioritized Throughput - The sum of all the end times of each task, weighted by their priority.
- Solution/Genome - A solution will be represented by a list of resources and the order in which tasks run on each resource.
- Reproduction - Creating new solutions from two old solutions by randomly choosing a crossover index and swapping the tasks after that index in each resource of one solution with the tasks after that index in each resource of the other solution.
- Fitness Function - A function applied to each generating that "sterilizes" solutions that violate any hard constraints and rewards solutions that measure well in terms of the soft constraints with a chance to reproduce.

Development Tools

- Genetic Algorithm
    - Python
- Web Framework for input/output
    - Flask (Python)
    - HTML/CSS/JavaScript

Development Sequence

1. Create genetic algorithm that satisfies the above description
2. Create a REST API with Flask (Python) that accepts the algorithm input and responds with the algorithm output
3. Create a user interface in the form of a web application to connect with the REST API endpoints

Anticipated Results

We expect that upon completion of our project, we will be able to consistently, effectively, and efficiently generate a task schedule using a genetic algorithm.

Sources

- Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System
- Genetic algorithm scheduling

Last Updated 11/14/13