Part 3: Bayesian Optimization of the flexible Job Shop Scheduling Problem

Group 06
Janette Rounds
David Rice
Mitch Vander Linden

October 19, 2015

(NOTE: this is just a sample outline, and you are free to deviate from this format).

1 Introduction

The following algorithm attempts to find an optimized way to schedule the Job-shop Scheduling Problem (JSP) in a more flexible manner called the flexible Job-shop Scheduling Problem (fJSP). Due to the problem being an NP-Hard solution, the algorithm takes advantage of hybrid evolutionary algorithms that use particle swarm optimization and basian network structuring to learn the optimal relationships between machine, task, and objective function.

The metrics of the algorithm's success in [**?**] compare the algorithm run on the following seven environments: Genetic Algorithm (GA), Binary Genetic Algorithm (Binary GA), Differential Evolutionary Algorithm (DE), Particle Swarm Optimization (PSO), Particle Swarm Optimization with Adaptive Grouping Differential Evolution Algorithm (SaNSDE), Cooperative Coevolution Group (CCPSO), and Bayesian Grouping (CCBhEA). CCBhEA had the most optimal max, mix, and mean time to compute for all scalings of the problem.

2 Motivation

The Job-Shop Scheduling Problem (JSP) is an NP-Hard problem in computer science[**?**]. Let us imagine we have a set of machines and a set of jobs to be completed. Each job is a set of operations and the operation order is fixed. In the classical formulation of JSP, each operation has a fixed processing time, and a required machine. However, these authors used the flexible job-shop scheduling problem (fJSP) that assumes that one machine can perform multiple kinds of operation, and as such, there is no fixed processing time [**?**]. There are several potential constraints we could use including: a job does not visit the same machines twice; there are no precedence constraints among operations of different jobs; operations can not be interrupted; etc. Both JSP and fJSP are minimization problems, that is, we generally want to minimize the overall time to completion of all the jobs. As we stated before, there is no way to solve this problem in less than exponential time. Therefore, there are several algorithms we can use to approximate the solution. The authors combined Bayesian Optimization and Evolutionary Algorithms approaches to approximate a solution for the fJSP.

3 Algorithm

Provide pseudocode for the algorithm that your team is investigating. Provide pseudocode for the algorithm, and be sure to

Algorithm 1 AwesomeAlgorithm

```
1: Input: TODO:list input
2: Output: TODO:list output
3:
4: k ← Random(0, 10)
5: if n = 0 then
6:     while i < k do
7:         a[i] ← b[i]
8:     end while
9:      return a
10: else
11:      return b
12: end if
```

4 Analysis

Optional: you are welcome to provide a brief proof of correctness or a running time analysis of the algorithm. The difficulty of doing this will, of ocurse, depend on your algorithm.

5 Discussion

For the scope of this project, it will be most beneficial to only look at the Bayesian Grouping variant of the algorithm given the CCBhEA implimentation brought the most optimal results for all scalings of the problem size. It will also be helpful to keep the context of the project generalized to any grouping of machines that share a multitude of functions rather than applying the algorithm to one specific scenario. A task of primary interest will be to see if there are any ways to improve the current CCBhEA algorithm. There may be an time improvement optimization ready to be found within the structure of the update functions found in the while loop of the pseudocode.