Final Project Specification

Sam Kim[1], Nick Merrill[2], Crystal Stowell[3]
[1]samuelkim@college.harvard.edu
[2]merrill@college.harvard.edu
[3]cstowell@college.harvard.edu

Overview

In the physical world, many of the problems that surround us can be represented by mathematical equations and models. The class of optimization problems is a popular one as most companies and individuals struggle to maximize utility. One such problem is the nurse scheduling problem. This problem comprises the difficulties that many round-the-clock companies and organizations face: how to best spread their finite resources of employees. Several restraints arise in all scheduling problems. Namely, no individual can work more than two back-to-back shifts. Also, most employees do not want to have more than x number of night shifts in a given week. There may also be personal preferences of vacation time and other factors. In order to maintain the functionality of the hospital while meeting the requests of the most employees possible, an optimization must occur.

To solve this optimization problem, we will be focusing on using the Cuckoo Search algorithm, which is a meta-heuristic optimization algorithm published in 2009. This algorithm attempts to find the optimal solution using a genetic algorithm approach, in which new solutions are randomly generated from the previous generation of solutions. Several modifications exist, such as the use of Levy flights in place of the random walk, as well as multi-objectivity. Our goal for the project is to implement a basic version of the cuckoo search to optimize the nurses' schedules. Said modifications will be included if time allows. The algorithm can also be applied to other optimization problems.

Feature List

Core features

- Optimize an objective function.
- Provide a set of optimized solutions (See next section for implementation details).
- Interface the problem to be solved with the back-end optimization algorithm.
- Optimize a multi-objective function.

Cool extensions

- Create a user friendly frontend for users to enter a problem and fitness function of choice.
- Create a pretty way to visualize the results.

Modularization

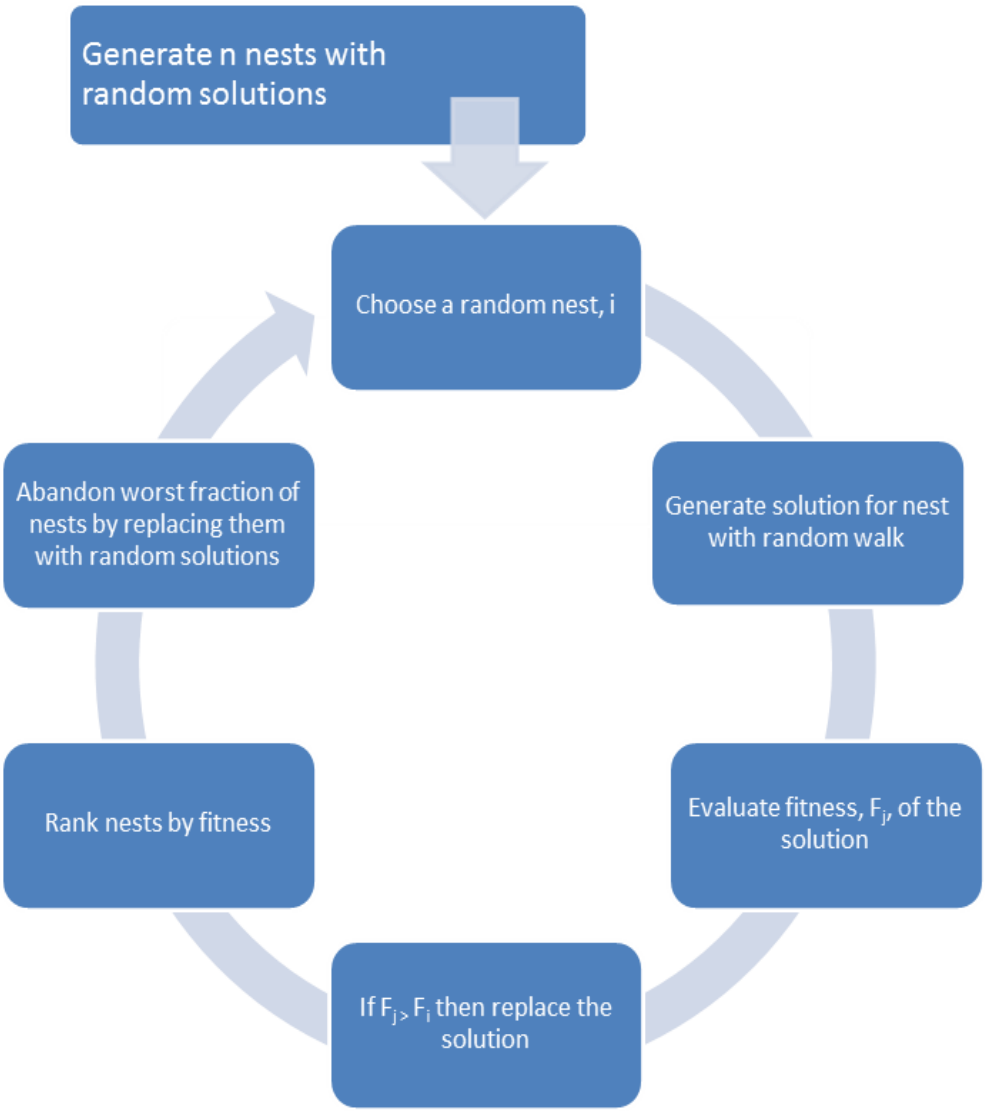
1. *Initial phase*
    - Back End:
        i. Objective/fitness function
        ii. Solution
        iii. Walking - Generating new solutions
            1. Random walk
            2. Levy flights
        iv. Overall algorithm (which includes above functions)
            1. Cuckoo Search (CS)
    - Front End:
        i. Defining the problem to be optimized
2. *Modification phase*
    - Multiobjective
    - Variants of Cuckoo Search: Quantum-inspired, modified CS
    - Compare to different optimization algorithms
3. *Cool features phase*
    - Graphical frontend interface
        i. allow manipulation of objective(s)
        ii. visualization of results
    - Build Particle Swarm Optimization (PSO) and compare the results to CS

Code Framework

```
abstract class OptimizationAlgorithm {
      public OptimizationAlgorithm(Walk w);
      public SolutionSet newSolutionSet(SolutionSet seed);
      public void solve();
      public SolutionSet getSolutions();
}
public static class CS implements OptimizationAlgorithm {
```

```
	abstract class Walk {
		public Solution walk(Solution seed);
	}
	public class RandWalk implements Walk {}
	public class LevyFlights implements Walk {}
}
public static class PSO implements OptimizationAlgorithm {}
abstract class OptimizationProblem {
	public double fitness(Solution s);
	public void printSolutions();
}
abstract class SolutionSet {
	private Solution solutions;
	public void addSolution(Solution s);
}
abstract class Solution {
	public Solution();
	// n is the length of the solution vector
	public Solution(int n);
}
// Cuckoo Search
public class CSSolution extends Solution {}
public class CSSolutionSet extends SolutionSet {}
// Particle Swarm Optimization
public class PSOSolution extends Solution {
	public void setVelocity(double v);
	public double getVelocity();
}
public class PSOSolutionSet extends SolutionSet {}
public class NurseSchedProb extends OptimizationProblem {}
```

Flow Diagram



Whats Next? Goals before the next checkpoint:

- Confirm the choice of Java as the final language
- Set up environments.
- Decide on an optimization algorithm.
- Decide on the optimization problem.
- Decide on the project.
- Get Crystal on Java :)