CS 344: Nature Inspired Computation

**Programming Assignment 1: Evolutionary Systems**

**Genetic Algorithms vs. Population Based Incremental Learning**

Richardo Hopkins, Luis Rosias, Amy Sham

February 20, 2013

**Introduction/Background**

This programming assignment implemented a genetic algorithm and a population based incremental learning (PBIL) algorithm for solving the MAXSAT problem. Both algorithms use an incremental approach to finding the “fittest” solution.

With the genetic algorithm, the population is initially randomly generated, and through various selection methods and the use of a fitness function, a breeding pool is created. This breeding pool is modified via crossover and mutation to generate a new population. The process is repeated a given number of times incrementally optimizing the solution through each generation.

The PBIL algorithm, instead of modifying a population of individuals, produces populations with each generation based on a probability vector that encodes the likelihood of a given individual of having that value set. It selects the fittest individual using a fitness function, and uses that individual to modify the probability vector. Then it selects the least fit individual and adjusts probability vector “away” from its values. As the process repeats, the probability vector continually improves over each generation, which, consequently, improves the population it models.

The MAXSAT problem describes a series of Boolean clauses, each a disjunction of variables and their negations, and to solve the problem is to generate a solution that satisfies (i.e., makes true) the maximum number of clauses possible. With regard to the genetic and PBIL algorithms, the MAXSAT problem is modeled as a population of possible solutions, where a solution’s fitness is evaluated as a function of the number of clauses it satisfies.

Brief Overview of Work and Findings

We created a genetic algorithm and a PBIL algorithm to solve the MAXSAT problem. For the genetic algorithm, we implemented Boltzmann selection, tournament selection, and rank selection and one-point crossover and uniform crossover. For mutation, we wrote a general method that ran through each bit of each individual in the population and mutated it based on a small mutation probability. For the PBIL algorithm, we implemented each individual in the population as a Boolean array.

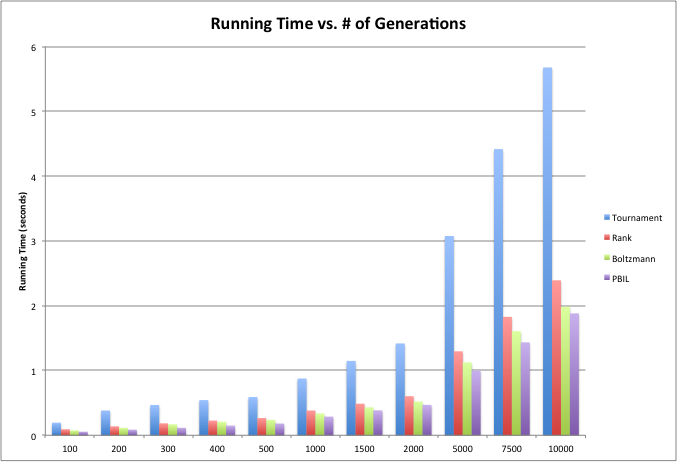
We found that PBIL consistently produced fitter solutions. For the genetic algorithm, we found that all of the selection methods produced highly variable results for solution fitness. With regard to running time, PBIL consistently performed above all of the genetic algorithm methods. Of the genetic algorithm types, Boltzmann selection ran the fastest.

**Experiments and Results**

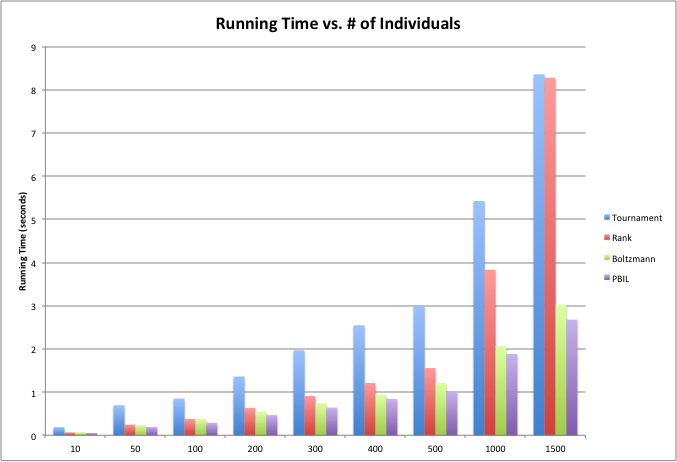
Testing Methodology

We tested all of the genetic algorithm selection methods and PBIL by finding fitness and running time as functions of population size and generation size. In total, we have four sets of data (see graphs below).

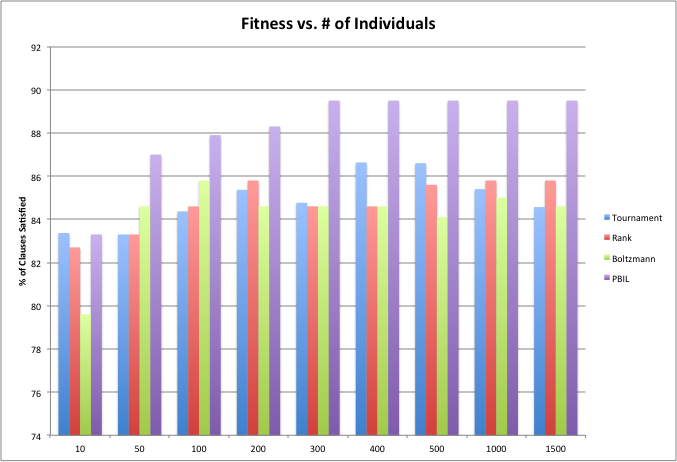
Findings



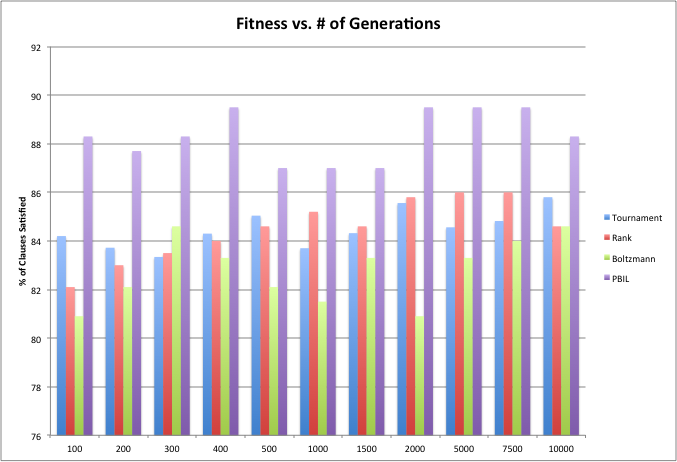
As expected, tournament selection vastly underperformed compared to the other genetic algorithm selection methods and PBIL. This may be because it needs to allocate additional memory in the selection process. PBIL did not significantly outperform the other genetic algorithm selection methods; it is only slightly better than performance with Boltzmann selection, which, in turn, is only slightly better than rank selection.



The PBIL algorithm handled larger population sizes better than did the genetic algorithm. For the genetic algorithm, all of the selection types performed at comparable times for small populations, but differences in running time emerged for larger populations.



PBIL clearly produced fitter solutions than the genetic algorithm, with the levels of fitness being consistent across the large population sizes tested. The genetic algorithm produced similar fitness results across all selection types and all population sizes. These findings suggest that increasing population size does not significantly increase the fitness of the solutions.



Once again, PBIL produced fitter solutions than the genetic algorithm types. As was the case with modifying the population size, changing the number of generations did not produce significantly fitter solutions (with the exception of very small generation sizes).

**Conclusion**

We recommend the PBIL algorithm over the genetic algorithm. PBIL outperformed the genetic algorithm for both running time and fitness for varying population sizes and generations. The PBIL algorithm was also far more consistent with regard to fitness than was the genetic algorithm, whose solutions varied by multiple percentage points. The PBIL algorithm was also a lot easier to implement.

Crown goes to PBIL!