Multi-Objective Knapsack Problem

Genetic Algorithms and Wisdom of Crowds

Sarah Mullins

Speed School of Engineering

University of Louisville

Louisville, USA

skmull02@louisville.edu

Ashley Revlett

Speed School of Engineering

University of Louisville

Louisville, USA

anrevl01@louisville.edu

**Abstract – The purpose of this project was to explore the application of genetic algorithms and wisdom of crowds to the multi-objective knapsack problem. This is a variation of the classic NP-complete knapsack problem, in which the contents of a container should have maximum value without outweighing the capacity of the container. A multi-objective version of this problem involving the maximization and minimization of given values was explored. Large values for the number of objects and knapsack capacity quickly increase the computational complexity of the problem, making brute-force techniques intractable. Instead, genetic algorithms were combined with the wisdom of crowds to attempt to find optimal solutions. The genetic algorithm converged relatively quickly toward near-optimal values, but often got stuck in local optimal solutions. Applying the wisdom of crowds approach to the results did not necessarily yield improved solutions.**

**Introduction**

The original Knapsack Problem is based on the following premise: if there are *n* objects with weights *w* and values *v*, what is the optimal configuration of objects within a knapsack of capacity *c*? That is, how can *v* be maximized so that the total of all *w* is less than *c*?

There are several different variations of the Knapsack Problem. For example, in the 0-1 version, no object can be placed into the knapsack more than once [1]. A generalized form of 0-1 called MOKP (multi-objective Knapsack Problem) will be explored in this paper. In this type of problem, instead of having only a weight and value, other features affect the fitness of an object. In our project, each object has a price, weight, and value. The goal is to maximize the value while keeping the price and weight below arbitrary limits. This problem utilizes the concept of Pareto efficiency or optimality, which describes situations in which improving the status of one individual degrades the status of another individual [2].

**Approach**

When *n* and *c* have large values, the number of possible knapsack “packings” becomes exponentially large. Our approach was based on using a genetic algorithm to explore this search space for a near-optimal solution in a more computationally efficient manner by using a genetic algorithm (GA). Because genetic algorithms can get stuck in local maxima and miss the global optimum, the GA is run repeatedly, and all GA solutions are considered using a “wisdom of crowds” approach. The program structure and algorithms work like so:

1. Build an initial population of random solutions, encoding the solution’s boxes into a “chromosome”-like list of 1’s and 0’s.
2. Evolve the population over many generations by mutating and crossing over the chromosomes (more information below). The fitness of a solution increases with the value of the boxes contained in the solution, and decreases with the price and weight. To assess fitness, normalize the scores for each objective, then aggregate them. Return the best solution found during the evolutionary process.
3. Repeat steps a-b to create each member of the crowd.
4. Try to create a “wiser” solution by selecting boxes common to most of the crowd’s solutions, then adding any other boxes that will fit.
5. Return the best solution found.

For step b), the following crossovers and mutations were performed:

1) Remove chromosomes exceeding the knapsack capacity  
2) Clone the chromosomes with the highest value

3) Breed children using crossover until the population reaches its original size  
Note: Crossover involved combining the first half of the first parent with the   
second half of the second parent.

4) Mutate children with the lowest value  
Note: Mutation involved removing or adding a random number of boxes at   
random chromosome indices.

**Experimental Results**

**Data**

**Results**

**Conclusions**

**Acknowledgements**

**References**

[1] M. Hristakeva and D. Shrestha, "Solving the 0-1 Knapsack Problem with Genetic Algorithms," Simpson College, Midwest Instruction and Computing Symposium. 2004.

[2] C. L. Mumford, "Comparing Representations and Recombination Operators for the Multi-Objective 0/1 Knapsack Problem," *Evolutionary Computation,* vol. 2, pp. 854-861, 8-12 Dec. 2003 2003.