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].

MOKP is an NP-complete problem; that is, a nondeterministic polynomial one. To understand this concept, we must first define polynomial problems (P-problems). The time taken to perform a given algorithm can be described in terms of Big-Oh Notation. For example, a simple for-loop that goes through iterates *n* times will have a time complexity of O(*n*)*.* If two of these loops were to be nested, the Big-Oh time complexity of the algorithm would be O(n2), and so on and so forth [4]. Polynomial problems can be solved in time O(nk), for some value of *k*. In nondeterministic polynomial problems (NP-problems), heuristics can be used to estimate a solution, and can confirm its accuracy in, best case scenario, time O(nk) . However, this is assuming adequate computational resources and/or lucky guessing; it is possible that the optimal solution cannot be found given the time and resources available. NP-complete problems are considered the most difficult problems within the realm of those in the NP category, and MOKP is included in this group [5].

We sought to find optimal solutions to the MOKP using both genetic algorithms (GA) and wisdom of crowds (WoC). Genetic algorithms can be used to apply the organic processes of evolution and natural selection to NP-complete problems. A population of random potential solutions for a given problem is first generated. Next, some of the best solutions are “bred” together to create new children solutions, while others are mutated or removed from the population altogether. This takes place for a given number of generations, or until a certain fitness threshold has been met [5].

WoC refers to the concept that problems can sometimes be better solved through an amalgamation of solutions given by several people, or even by the same person on several different occasions [6]. In particular, we implemented wisdom of artificial crowds (WoAC), in which the solutions provided by each iteration of the genetic algorithm were combined to create a hopefully optimal solution [7].

Coello describes the application of genetic algorithms to multiobjective optimization (EMOO) in his 2001 article. We used his aggregation method, in which the objectives are combined into one function, due to its simplicity and ease of implementation [3].

**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 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 the WoAC 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**

**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.

[3] C. A. C. Coello, "A Short Tutorial on Evolutionary Multiobjective Optimization," in *Evolutionary Multi-Criterion Optimization*. vol. 1993, ed: Springer Berlin Heidelberg, 2001, pp 21-40.

[4] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed.: Pearson Education, Inc., 2010.

[5] M. A. Weiss, *Data Structures and Algorithm Analysis in C++*, 3rd ed.: Pearson Education, Inc., 2006.

[6] S. K. M. Yi, M. Steyvers, M. D. Lee, and M. J. Dry, "Wisdom of the Crowds in Traveling Salesman Problems”

[7] R. Yampolskiy, L. Ashby and L. Hassan, "Wisdom of Artificial Crowds—A Metaheuristic Algorithm for Optimization," *Journal of Intelligent Learning Systems and Applications*, Vol. 4 No. 2, 2012, pp. 98-107.