Decision in Finance from the Knapsack Point of View

Geir Agnarsson, Jae-Moon Hwang, Matthew Kearney

Mason Experimental Geometry Lab

George Mason University

December 6, 2019

Introduction

The Knapsack Problem has been prevalent in the field of Computer Science and Mathematics for at least a century. Its applications are plentiful as it naturally occurs in resource allocation problems, which are numerous in finance. As such, we are studying the knapsack problem as it applies to optimizing a financial portfolio. Additionally, we use sensible assumptions about our financial application in order to be able to solve and optimize solutions in a reasonable time.

Preliminary Definitions/Concepts

Definition (Linear and Integer Programming)

- **Linear Programming (LP)** is an optimization method that applies to mathematical models that can be characterized by linear relationships.
- In LP, the set of possible choices can be represented as a convex polytope due to the nature of using linear inequalities.
- Solutions are always at one of the vertices of this region. If more than one vertices yield the same maximum value, then the set of optimal solutions is another infinite set of points.
- **Integer Programming (IP)** seeks to solve optimization problems in which variables are restricted to integer values.
- IP and LP are often at odds as restricting solutions to integers prevents LP from helping as seen in certain variants of the knapsack problem which we are studying.

Definition (Knapsack Problem)

- The general **Knapsack Problem** seeks to answer the following:
- Given a set of n elements, with each element having a weight w_i and value v_i, determine what combination of the elements give the highest value while respecting a given constraint on the weights.
- We want to maximize

$$f(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} v_i x_i$$

while satisfying the condition

$$C(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} w_i x_i \leq W$$

where W is the maximum weight capacity and each x_i is a non-negative integer.

More on the Knapsack Problem

- While the general knapsack problem is easy to understand, solving and optimization is difficult. In fact, the knapsack problem belongs to a class of problems labeled as **NP-Complete**.
- The classification implies, in terms of efficiency, that there is no efficient algorithm that can solve the problem better than brute force.
- Although the general case is complex, certain assumptions about the items allow one to solve the problem through quicker algorithms.
- If fractional amounts of items were allowed, then LP techniques would allow us to find a quick solution: one simply sorts the items by their value to weight ratio, and takes as many items as possible to fit with the given constraint.
- With the **0-1 knapsack problem**, we add one additional constraint on the number of items as given:

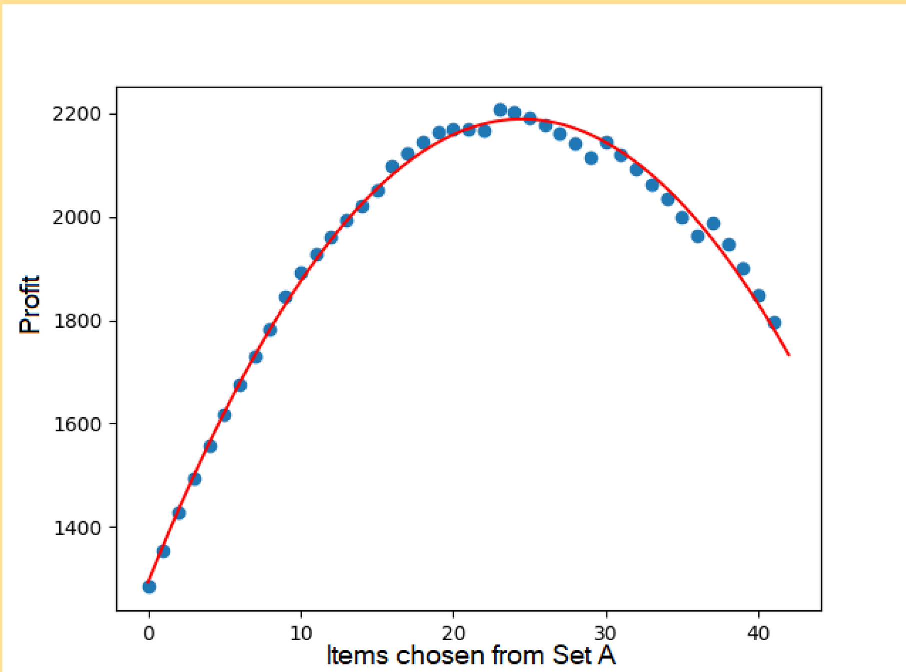
$$C(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} w_i x_i \leq W \textbf{ and } x_i \in \{0, 1\}$$

Finance Point of View

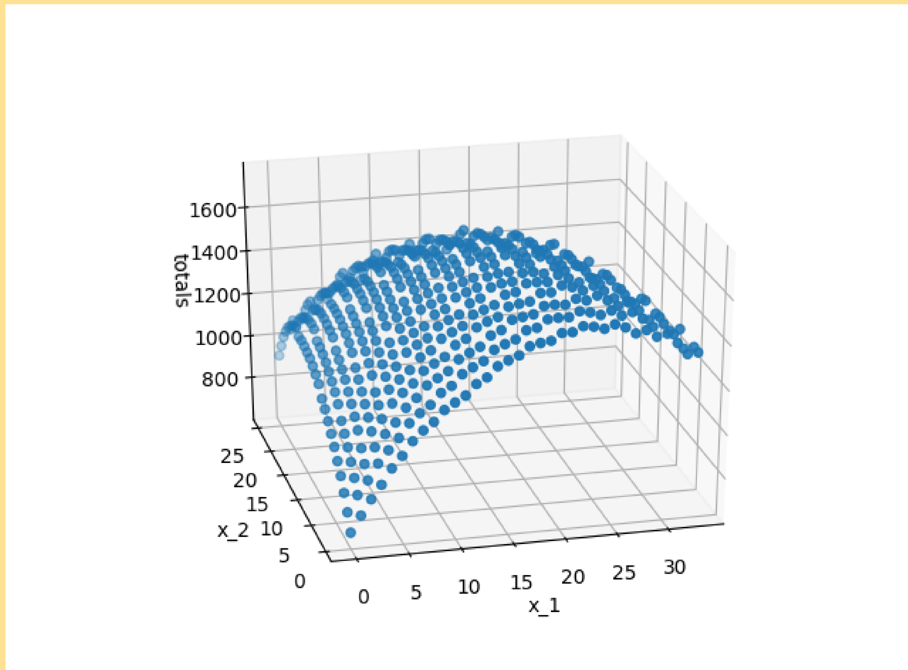
- In our finance application, we look at the following scenario: An investor is looking to buy a collection of companies, c_1 to c_n, with corresponding prices, p_1 to p_n. For those prices, we assume the change in price, $\Delta p_i = p_{i,k+1} - p_{i,k}$, satisfies Δp_i which is fixed around a neighborhood of time k. In this way, we want to maximize $\sum_i x_i \Delta p_i$ for which $\sum_i x_i p_i \leq B$ where B is the budget of the investor and $x_i \in \{0, 1\}$.
- In our research, we looked at some simple cases in regards to p_i where it takes on a single value, two distinct values, and three distinct values. For the single value case, we can simply sort the items by value and pick until we reach the budget.
- For two distinct values, we sort the items based on their weights and fix the amount of one item. From there, we iteratively calculate how many items can we get of the second weight and arrive at a solution. This can be done in linear time, but this can be done in O(nlogn) time overall as the sorting dominates the worst-case runtime.
- For three distinct weights, we do the same as two distinct weights, but we fix the amount of items for two distinct weight values. The calculations are done in quadratic time, which overtakes the previous worst-case run time. Namely, it is $O(n^2)$.

Figures

Two distinct weights, [6,7], W = 250



Three distinct weights, [5,7,8], W = 182



Findings

- While examining the knapsack problem, we came to ask some questions with one being if we can get a "zig-zag" phenomena where a smaller value can be between two larger values creating a "cup"-like situation. We were able to find examples of such situations in both the two- and three-weight case.
- As seen by the figure that shows the two-weight case, there is a strict increase in the values, but it then results in a jagged effect as gaps are created due to only allowing integer solutions.
- The analogous situation happens in the three weight case where the surface looks relatively monotonic, but closer inspection shows that there are local "cups" where higher values surround a lower one. This also shows the difficulty as finding efficient solutions for IP.
- Another question we ask is whether a curve of best fit will stay close to the optimal value? This was found to not be true as, in some cases, the curve of best-fit will completely undershoot the optimal value.

Future Work

- If we were to continue this research, one question we would like to examine closely is whether, in the long run, can we maximize the profit from repeated buying and selling of companies by knowing each $p_i(t) = \sum_{\tau=1}^{t} \Delta p_i(\tau)$ over a long period of time?
- Additionally, we would want to look into ways to quickly find solutions for the case of four or greater distinct weights as each weight increases the worst case by a power. This can be possibly be done by looking more closely at the geometric properties of the possible solutions as given by a convex polytope.

Acknowledgments

We'd like to thank Geir Agnarsson for leading this project and the MEGL program for the research opportunity.

References

"G. van Rossum, Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995."

Alexander Schrijver, Theory of linear and integer programming. Wiley-Interscience Series in Discrete Mathematics. A Wiley-Interscience Publication. John Wiley Sons, Ltd., Chichester, 1986. xii+471 pp. ISBN: 0-471-90854-1