# Optimization Problems, Lecture 1, Segment 1

John Guttag

MIT Department of Electrical Engineering and Computer Science

# Computational Models

- Using computation to help understand the world in which we live
- Experimental devices that help us to understand something that has happened or to predict the future





- *Optimization models*
- Statistical models
- Simulation models

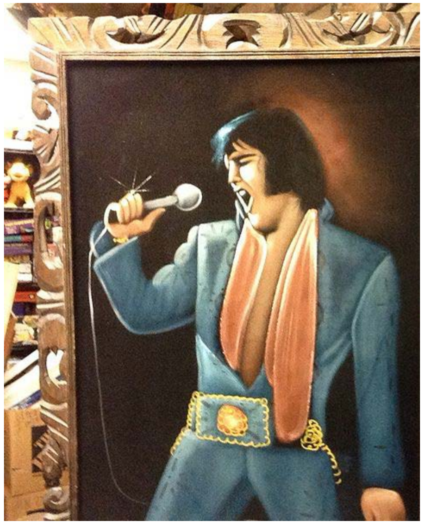
# What Is an Optimization Model?

- An objective function that is to be maximized or minimized, e.g.,
  - Minimize time spent traveling from New York to Boston
- A set of constraints (possibly empty) that must be honored, e.g.,
  - Cannot spend more than $100
  - Must be in Boston before 5:00PM

# Takeaways

- Many problems of real importance can be formulated as an optimization problem
- Reducing a seemingly new problem to an instance of a well-known problem allows one to use pre-existing methods for solving them
- Solving optimization problems is computationally challenging
- A greedy algorithm is often a practical approach to finding a pretty good approximate solution to an optimization problem

# Knapsack and Bin-packing Problems








# Knapsack Problem

- You have limited strength, so there is a maximum weight knapsack that you can carry
- You would like to take more stuff than you can carry
- How do you choose which stuff to take and which to leave behind?
- Two variants
  - 0/1 knapsack problem
  - Continuous or fractional knapsack problem

versus

# My Least-favorite Knapsack Problem

CC BY File Upload Bot (Magnus Manske)

CC BY Zantastik~commonswiki


CC BY JOADL

CC BY ZooFari

CC BY maebmjj

CC BY Gorivero

# 0/1 Knapsack Problem, Formalized

- Each item is represented by a pair, *<value, weight>*
- The knapsack can accommodate items with a total weight of no more than *w*
- A vector, *L*, of length *n*, represents the set of available items. Each element of the vector is an item
- A vector, *V*, of length *n*, is used to indicate whether or not items are taken. If $V[i] = 1$, item $I[i]$ is taken. If $V[i] = 0$, item $I[i]$ is not taken

# 0/1 Knapsack Problem, Formalized

Find a V that maximizes

$$\sum_{i=0}^{n-1} V[i] * I[i].value$$

subject to the constraint that

$$\sum_{i=0}^{n-1} V[i] * I[i].weight \le w$$

# Brute Force Algorithm

- 1. Enumerate all possible combinations of items. That is to say, generate all subsets of the set of subjects. This is called the power set.
- 2. Remove all of the combinations whose total units exceeds the allowed weight.
- 3. From the remaining combinations choose any one whose value is the largest.

# Often Not Practical

- How big is power set?
- Recall
  - A vector, *V*, of length *n*, is used to indicate whether or not items are taken. If *V[i] = 1*, item *I[i]* is taken. If *V[i] = 0*, item *I[i]* is not taken
- How many possible different values can *V* have?
  - As many different binary numbers as can be represented in *n* bits
- For example, if there are 100 items to choose from, the power set is of size 126,765,060,022,822,940,149,670,320,5376

# Are We Just Being Stupid?

- Alas, no
- 0/1 knapsack problem is inherently exponential
- But don't despair