A Knapsack Problem: Fitting a Subset of Song Tracks for a Given Time Interval

Arun Drelich, Chris Hinstorff, Sam Slate and Josh White

24. October 2015

Problem Statement

We define our problem as a subset of the Knapsack Problem. Given a set of n songs $S = \{s_0, s_1, \ldots, s_n\}$, how can we generate a subset $R \subseteq S$ such that the duration of R, defined as $||R_L||_1$, is constrained by a given duration C? We include the additional optimisation constraint requiring that expression $\Delta t = |C - ||L||_1|$ be minimised.

Notation and Formal Definition

Define S as the set of songs, then $n = |S|$. Define additionally vectors $\mathbf{l}$ and $\mathbf{w}$, representing the length (duration) of each song and the weight attributed to each song, respectively. Lastly, define C as the maximum-allowed duration of time for songs to play within. Note that $||\mathbf{x}||_1$ represents the L1-Norm of a vector $\mathbf{x}$. Throughout this document, convention will be to **bold** vectors, use capital Roman letters for matrices, and use regular typeface for scalar values.
We wish to find a result set $R \subseteq S$ such that the following hold:

$$\max_{\mathbf{w}} ||R_w||_1 \tag{1}$$

$$||R_l||_1 \leq C \tag{2}$$

$$\min_{\mathbf{l}} \Delta t \tag{3}$$

Where R_w and R_l are defined as the weights and lengths of the songs in R, in respective ordering.
Thus, this problem is one of optimisation. We disregard a brute-force solution with runtime $O(2^n)$ and instead opt for a dynamic programming solution with runtime $O(nC)$.

DP-Solution to the Knapsack Problem

Let the $(n+1) \times (C+1)$ matrix B store the partial solutions to the problem. Then B is recursively defined as follows:

$$B(k,l) = \begin{cases} B(k-1,l) & : \mathbf{l}_k > l \\ \max\{B(k-1,l), B(k-1,l-\mathbf{l}_k) + \mathbf{w}_k\} & : \text{else} \end{cases}$$

For $k \in [1,n] \subseteq \mathbb{Z}^+$ and $l \in [0,C] \subseteq \mathbb{Z}^+$.

Minimising Δt

To minimise Δt, we must define a linear function $\mathbf{w}_i(\mathbf{l}_i) = k\mathbf{l}_i$. Fortunately, since all that matters is relative proportionality, we can use $k = 1$ and so $\mathbf{w} = \mathbf{l}$.

A Python Implementation

An implementation in Python is used by our inTime API Server in order to compute a set of song tracks that can be played back to fill up a given time interval.