A Problem in Optimization

Summer Project

Formal Problem Statement

Given K bitonic discrete functions $f_1(\tau_1), \ldots, f_k(\tau_k)$; $\tau_i \in \mathbb{W}$, we need to find the sequence $A = \{a_1, a_2, \ldots, a_T\}$, where $a_i \in \{0, 1, 2, \ldots, K\}$, which maximizes the function

$$R = f_{a_1}(\tau_{a_1,1}) + f_{a_2}(\tau_{a_2,2}) + \ldots + f_{a_T}(\tau_{a_T,T}) = \sum_{a_i \in A} f_{a_i}(\tau_{a_i,i})$$

where $\tau_{a_i,i}$ denotes the value of τ_{a_i} at the i^{th} time instant. $f_0(\tau_0)$ is trivially defined to be zero for all τ_0.

The above is equivalent to sampling one of the K given functions at every instant of time, for T instants of time. Consider the t^{th} time instant. Let t_i be the largest number such that $t_i < t$ and $a_{t_i} = i$, and zero if no such t_i exists. Then, the parameter τ_i corresponding to each f_i at the time instant t is given by:

$$\tau_i = t - t_i - 1$$

Intuition

The problem statement can be visualized in the following manner:

- There are K arms present. Pulling an arm results in a reward, which is a function of τ. The rewards associated with each arm are a known function of the parameter τ.
- There are T time instants, in each of which we may choose to pull any one of the k arms or none at all.
- The parameter τ associated with a particular arm is representative of the time elapsed since the arm was last pulled. In other words, pulling the i^{th} arm resets τ_i to 0, while increasing the τ values of all the other arms by one.
- We need to find the sequence in which the arms must be pulled to obtain the maximum total reward.

Variations

Possible variations in the problem statement are:

- The maximum delay between two pulls of a particular arm is given. Thus, there is an upper bound on the value of τ.
- The maximum number of times an arm may be pulled in the T time instants is given.

Potential problem-solving strategies

The following are potential strategies to devise an algorithm to solve the problem:

- Brute Force
- Dynamic Programming
- Divide-and-Conquer: If the problem can be solved by finding solutions to a number of subproblems, a divide and conquer algorithm can be devised.
- Incremental solution: This can be applied in one of two ways. The first is if we are given the optimum sequence A for K arms, which we are able to use to obtain the optimum sequence for k+1 arms. An alternative approach would be to increment the time instead of the number of arms.
- Linear programming