A review of the sequential testing problem and its extensions

Tonguç Ünlüyurt

Sabanci University

Istanbul, Türkiye

Agenda

- Problem definition
- Motivation
- Variations of the problem
- Exact and approximation results
- Conclusions

This presentation is based on:

Ünlüyurt, Tonguç. "Sequential testing problem: A follow-up review." ***Discrete Applied Mathematics*** **377 (2025): 356-369.**
and
Ünlüyurt, Tonguç. "Sequential testing of complex systems: a review." ***Discrete Applied Mathematics*** **142.1-3 (2004): 189-205.**

Possible Titles

- Sequential Testing of …. Systems
- Sequential Diagnosis of … Systems
- Sequential Fault Diagnosis
- (Adaptive or Stochastic) Function Evaluation
- Resolution of Boolean Formulae
- Binary Identification Problem
- …
- ..

Problem Definition

Goal: Efficiently identify system state under uncertainty with the minimum expected cost

Examples:

- Fault diagnosis
- Medical testing
- Network connectivity
- Order fulfillment
- Query optimization
- Inspection of incoming containers at a port
- Testing wafer probes

Problem Definition

- Given $f(\boldsymbol{x})$, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$
  and $f(\boldsymbol{x})$ are discrete valued (in particular Boolean),
- would like to figure out the correct value of $f(\boldsymbol{x})$with the minimum expected cost.
- Cost of testing (learning the value of) x_i is c_i
- Typical assumption: the variables assume values independently.
- $p_i =$ probability that $x_i = 1$, with $p_i + q_i = 1$.

Solution

- A solution is a strategy that tells us which variable to learn next given the values of already learnt variables.
- A natural way to describe a strategy is to represent it by a binary decision tree.

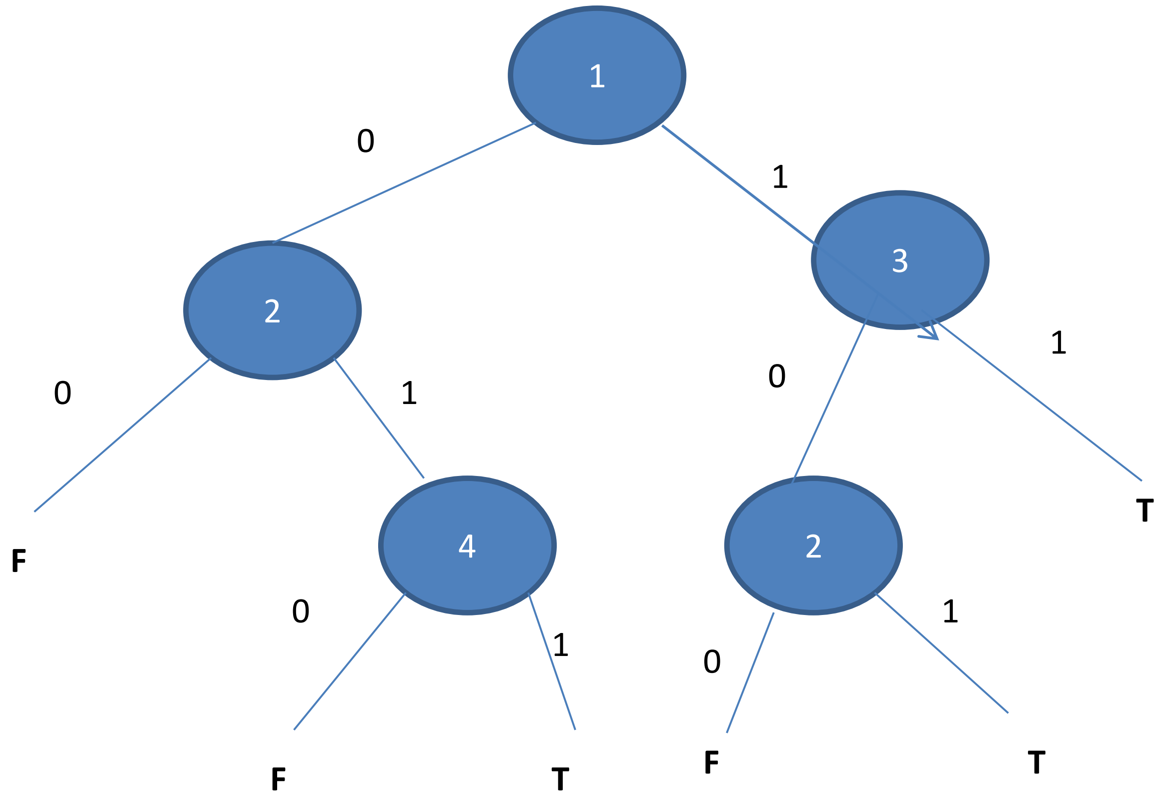
Solution

A strategy = **binary decision tree**

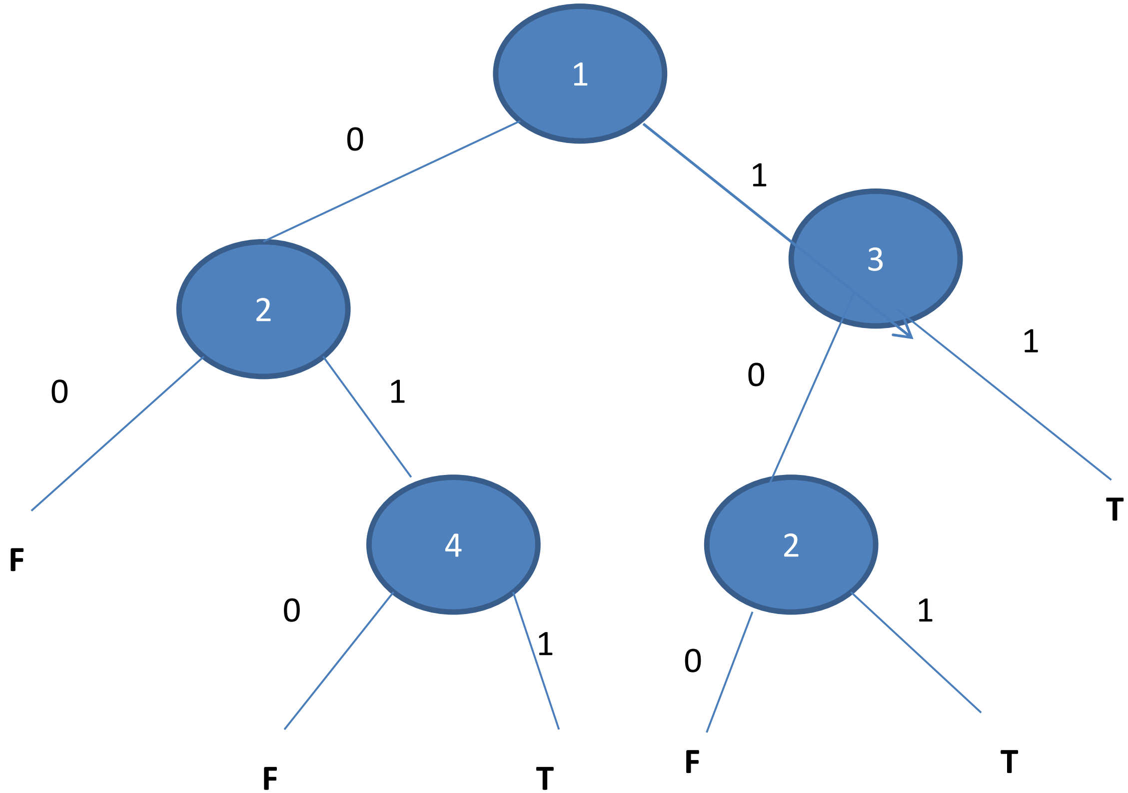
- Each node → test a variable
- Each branch → observed outcome
- Leaves → decisions

Example

$$f(x) = (x_1 \Lambda x_2) V (x_1 \Lambda x_3) V (x_2 \Lambda x_4)$$



Computing the expected cost



$$EC = c_1 + c_2(q_1 + p_1 q_3) + c_3 p_3 + c_4 q_1 p_2$$

$$f(x) = (x_1 \Lambda x_2) V (x_1 \Lambda x_3) V (x_2 \Lambda x_4)$$

Output

- The whole binary decision tree with the minimum expected cost.
- An algorithm that provides the next variable to test given the values of already tested variables.

Problem structure

- Representation of f
  - DNF and/or CNF

    DNF $f(x) = (x_1 \Lambda x_2) V (x_1 \Lambda x_3) V (x_2 \Lambda x_4)$

    CNF $f(x) = (x_1 V x_2) \Lambda (x_1 V x_4) \Lambda (x_2 V x_3)$
  - Binary decision tree
  - Oracle

Problem structure

- Dependence among tests
  - Independent
  - Dependency described by a table
  - Dependency described by conditional probabilites
- Special cases
  - Unit cost and/or identical variables

Adaptive/Non-adaptive strategies

- In general, strategies can be adaptive, in the sense that the next variable to learn depends on the values of already learnt variables.
- If the next variable to learn does not depend on the values of the variables learned so far, we refer to such a strategy "non-adaptive".
- We can efficiently represent a non-adaptive by a permutation.
- It is a relevant question to study how much we lose if we only consider non-adaptive strategies. **(adaptivity gap)**

Variations

- Precedence constraints among tests.
- Batch Testing

  Tests can be administered together as a batch. There is a fixed cost for each batch.
- Score classification problem

  The number of variables that are equal to 1 determines the class of an entity, for an arbitrary number of classes.
- d- Half space evaluation problem

  Would like to find out if d linear inequalities involving binary variables hold or not. (more general version: would like to evaluate a Boolean function of d variables and each variable is 0 or 1 depending one whether a linear inequality hold or not)
- Imperfect tests, classification over a sample, finding the cause of failure, minimize competitive ratio… etc

Hardness

- The problem is NP-hard when
  - f is a linear threshold function.(Cox et al. 1989)
  - f is a series-function with general precedence constraints among tests. (Burge et a.l 2005)
  - f is a series function with dependent variables. (Kaplan et al. 2005)
  - f is a general Boolean function. (Greiner 2006)
  - f is a read-once function with dependent variables. (Greiner 2006)
  - ..

Results

- Results (exact or approximate) typically assume independence and some special structure on the function.
- Numerical results, heuristics.
- Intuition: **Do the cheap tests that resolves the ambiguity the most, earlier.**

Classes of Boolean Functions

- Monotone (Positive) Boolean Functions
- Series/Parallel
- *k*-out-of-*n functions*
- Series-Parallel Systems (Read once functions)
- Double regular functions
- Linear threshold functions

Results –Series (Parallel) Function

$$f(\mathbf{x}) = x_1 \wedge x_2 \wedge \cdots \wedge x_n \quad \text{and} \quad f(\mathbf{x}) = x_1 \vee x_2 \vee \cdots \vee x_n.$$

- Any strategy is a linear ordering of tests.

$$\frac{c_{\sigma_1}}{p_{\sigma_1}} \leqslant \frac{c_{\sigma_2}}{p_{\sigma_2}} \leqslant \cdots \leqslant \frac{c_{\sigma_n}}{p_{\sigma_n}} \quad \text{and} \quad \frac{c_{\pi_1}}{q_{\pi_1}} \leqslant \frac{c_{\pi_2}}{q_{\pi_2}} \leqslant \cdots \leqslant \frac{c_{\pi_n}}{q_{\pi_n}}.$$

- $\pi(\sigma)$ is an optimal ordering for a series (parallel) function.

Results – Series System with Precedence Constraints

- Forest type (Garey 1973)
- Line type (Chiu et al. 1999)
- General - DP, Branch and Price,Heuristics (Rostami et al. 2019)
- General type – Hard

Application in project scheduling: (De reyck and Leus, 2008, Kreemers 2017)

k-out-of-n functions

- Takes the value 1 iff at least k variables take the value 1. (Ben-Dov 1983, Chang et al. 1990.)
- Branch and bound/dynamic programming approaches have been proposed for k-ot-of-n systems with precedence constraints.

(Wei et al. 2017, Rostami et al. 2019, Wei et al. 2013)

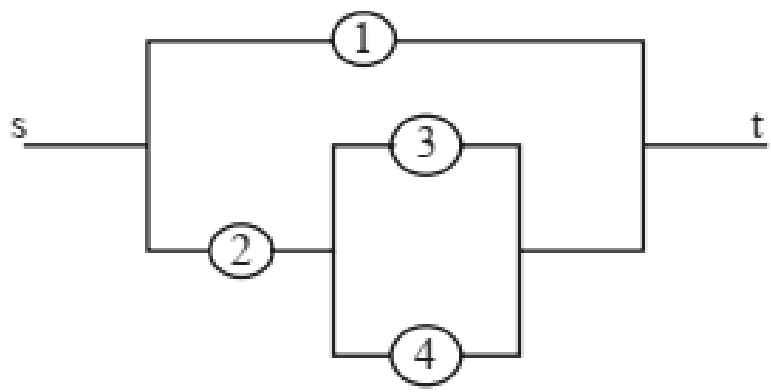
Double Regular Functions

- Regular function with respect to a permutation means there is a partial order that describes the effect of the value of variables on the value of the function.
- Double regular with respect to 2 specific permutations σ and π as defined before.
- Generalizes k-out-of-n functions.
- Polynomially solvable..

Series-Parallel Functions

- Also known as read-once functions.
- Can be constructed from serial or parallel connection of smaller SPSs.
- The simplest SPS is a single component.

Series-Parallel Functions



$f(x) = x_1 \vee (x_2 \wedge (x_3 \vee x_4))$ 3-level Series-Parallel Function

Adaptivity gap is not small.
(Hellerstein et al. 2022)

Series-Parallel Systems

- The "intuitive" depth-first strategy is optimal
  - For 2-level deep general SPSs
  - For 3-level deep SPSs with identical variables

  $(p_i = p, c_i = c)$
  - There are instances when it performs arbitrarily bad.

  (Russel 2006, Boros and Ünlüyurt 1999)
- 4-approximation for 2-level deep SPSs with dependent variables. (Kaplan et al. 2005)

Approximation Results

Approaches that proved to be useful:

- Round robin approach (Kaplan et al. 2005)
- Adaptive Submodularity and Q-value approach (Golovin and Krause 2011)

Approximation Results

- 3-approximation for threshold functions (Desphande et al. 2014, 2016)
- k-DNF ($4/\rho^k$) and k-term DNF ($\max\{2k, 2/\rho(1+\ln k)\}$) (Allen et al 2017) ($\rho = \min\{p_i, 1-p_i\}$)
- 8-approximation for Read Once functions (non-adaptive) (Happach et al 2022)

Approximation Results

- Score classification problem
  - $\log n$ approximation (Desphande et al 2013)
  - 2-approximation algorithm for unit cost case (Grammel et al)
  - $3 + 2\sqrt{2} = 5.82$ –approximation algorithm for the general cost case (Plank and Schewior 2024)
  - Constant factor approximation algorithm for the general case – **non-adaptive,** (Ghuge et al 2024)
  - O($d^2 \ln d$) approximation algorithm for d half-space evaluation problem (also for batched case) (Ghuge et al 2024)

Approximation - Batch testing

- It is possible to conduct multiple tests as a batch with a fixed cost plus variable costs of tests.
- Two versions: When certain subsets are possible and when all subsets are possible.
- For series system a 6.93+ε approximation algorithm (Daldal et al. 2017) has been improved to 1+ ε . (Segev and Shaposhnik, 2024)

Network Connectivity

- Given a network where some arcs may have failed, the goal is to find out whether two specific nodes are connected by a path that consists of working arcs or the network is connected.
- Erdös-Renyi random graphs with unit costs and fixed probability (Fu et al. 2014, 2017)
- Arbitrary costs and probabilities (Fu et al. 2017)
- Another version with limited number of queries (Guo et al. 2023)

Network Connectivity

- We are working on an extension where the total cost is dependent on the previous edge tested.
- Motivated by an application for after-disaster operations.
- After a disaster, some arcs of the road network may have been damaged.
- Would like to find out whether two important points are connected via "working" arcs.
- In order to do this, we send a drone to arcs to test them by image processing.
- In addition to testing costs, we also have travel costs that depend on the previous arc tested.
- The range and recharging of the drone can also be incorporated to the problem.

Network Connectivity

- Studied for a “series” function (Teller et al. 2019)
- Searching for a fuel station after a disaster (Khare et al. 2023)

Some interesting applications

- Order fulfillment (Baron et al 2024)
  - Fulfill an order of m items from n sellers.
  - A fixed cost of c_i is incurred if seller expects to ship any number of items.
  - The probability of accepting to send depends on the number of items left in the order. (binomial distribution).
  - Process stops when the order is fulfilled or there are no more sellers.

Very recent work

- Distributionally Robust k-of-n Sequential Testing (Tan and Nagarajan, 2026)
- Probabilities are drawn from a range.
- Goal is to find a strategy that minimizes the maximum expected cost.
- Result: 2 approximation for unit cost case.

Conclusions

- For what other classes of functions/data can we find exact/approximate algorithms?
- NP hard for general Read Once functions?
- What about special Read Once functions, for instance of constant depth?
- New extensions/variations?