Accepted Manuscript

Operations Research Applications of Dichotomous Search

Refael Hassin, Anna Sarid

PII: S0377-2217(17)30658-6
DOI: 10.1016/j.ejor.2017.07.031
Reference: EOR 14581

To appear in: *European Journal of Operational Research*

Received date: 28 February 2017
Revised date: 2 July 2017
Accepted date: 8 July 2017

Please cite this article as: Refael Hassin, Anna Sarid, Operations Research Applications of Dichotomous Search, *European Journal of Operational Research* (2017), doi: 10.1016/j.ejor.2017.07.031

Highlights

- This survey contains necessary background on dichotomous search.
- It is the first survey on Operations Research applications of dichotomous search.
- The focus is on models incorporating economic cost structure and constraints.

Operations Research Applications of Dichotomous Search

Refael Hassin[1] [2] and Anna Sarid[3]

Abstract An object is searched for in $\{1, \ldots, N\}$. Queries for the object are sequentially conducted. A query at x reveals whether the object's location is greater than x. The objective is to find the object within a minimal expected number of queries. This problem is called the "dichotomous search" problem and has many versions. This paper surveys dichotomous search problems with the emphasis on Operations Research applications.

Keywords: Combinatorial optimization; dichotomous search; alphabetic trees

[1]Department of Statistics and Operations Research, Tel Aviv University, Tel Aviv 69978, Israel. hassin@post.tau.ac.il
[2]Corresponding author
[3]Department of Statistics and Operations Research, Tel Aviv University, Tel Aviv 69978, Israel. annashva@yahoo.com

Contents

1. Introduction

Dichotomous search, as the name indicates, refers to algorithmic procedures that search for a *target* in an unknown location within an interval (the *interval of uncertainty*, or the *search interval*) by repeatedly dividing the interval into two parts. At each iteration, the searcher selects a point in the search interval and places there a *query*, determining at which side of the chosen point the target is located. This approach is ubiquitous and it is applied naturally not just by sophisticated scientists but also in everyday intuitive trial and error experimentation.

In the simplest form of dichotomous search, the searcher has no prior information on where the target is located (or assumes it is uniform over the interval of uncertainty), and the goal is to minimize the worst-case or expect cost of the search. In this simplest form, dichotomous search is reduced to the well-known *binary search* where the search interval is repeatedly halved.

This survey focuses on more sophisticated implementations of dichotomous search, for example when there is some prior information on the target location, when facing search constraints, or under specific forms of the objective function.

Formally, we consider search over ordered sets. The generic form of such a problem is the following: An object (the target, or the search key) lies at location x in the initial interval of uncertainty, $\{1, \ldots, N\}$. Queries for the object are sequentially conducted. Queries are comparison questions, presenting an integer y and returning whether or not $x \leq y$, thus creating a smaller interval of uncertainty. The objective is to minimize the expected cost of the search.

The literature on dichotomous search comes from several disciplines - computer science, applied mathematics, operations research, statistics, industrial engineering and economics.

This paper surveys dichotomous search problems and solutions in the area of Operations Research. For completeness we also briefly describe the closely-related theoretical contributions in other areas, mainly Computer Science. There are two relevant earlier surveys concerning dichotomous search. **Nagaraj (1997)** surveys the literature on computational methods for optimal binary trees, focusing on efficient algorithms, bounds and approximations. For completeness we briefly describe this necessary background. The more recent survey **Rytter (2005)** concentrates on Huffman tree problems, while very briefly relating to the alphabetic tree problem. It also includes a detailed illustration of the proof of the algorithm of Garsia and Wachs (1977) for the alphabetic tree problem.

Our mode of description differs from the above-mentioned surveys. We cover the literature by classifying it into sub-topics and focusing on each contribution separately. We add pointers to relevant results that focus on other variations of the problem. The order in each part is chronological, naturally creating a logical flow of the developments.

2. Preliminary information

2.1 Formulation in terms of alphabetic binary trees

A *t-ary tree* (or, a *t-way tree*) is a rooted tree such that every node has at most t children. A 2-ary tree is also called a *binary tree.* Nodes having no children are *terminal nodes*, or *leaves*, or *external nodes*, while the other nodes are *internal nodes.* It is common to denote by $l(j)$ the *path length* (also called the *depth* or *level*) of node j, corresponding to the number of arcs in the path from the root to j. The *depth of a tree* (also its *height*) is the maximal depth among its nodes. The *path length of the tree* is $\sum l(j)$. When there are positive weights $w_1, \ldots, w_N$ attached to the nodes, the *weighted path length of the tree* is $\sum_{j=1}^{N} w_j l(j)$.

A tree with minimum weighted path length is *optimal.* Without loss of generality, the weights w_i can be normalized to that their sum is 1. In this case, w_i represents the a priori probability of the object to be at location i. An optimal alphabetic tree is therefore associated with a search strategy minimizing the expected search time.

Alphabetic binary trees. These are binary trees for which there exists a planar embedding such that terminal nodes $1, \dots, N$ appear from left to right consecutively. Such binary trees are *alphabetic* or *order-preserving*. Alphabetic binary trees represent solution strategies for the dichotomous search problem. Say we are searching for a point among $\{1, \dots, 5\}$. The tree in Figure 1 represents the following strategy: First search at 3. If the object lies among $\{1, 2, 3\}$ then search at 2, otherwise search at 4. After at most three stages the object will be found. The path length of node x is the number of queries needed to find the object, if it lies at x. Thus the depth of the tree corresponds to the maximal number of questions needed to find the object (three in our example) and the (weighted) path length of the tree is N times the (weighted) average number of queries needed to find the object.

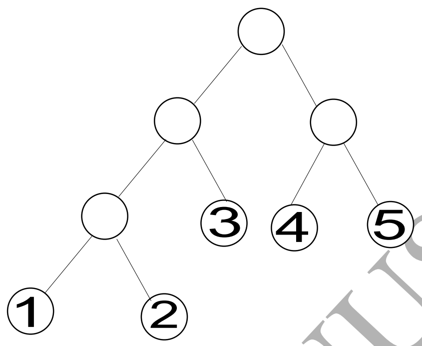

FIGURE 1. A 2-tree defines a search strategy

The *entropy* of the system is $H = -\sum w_i \log w_i$. **Gilbert and Moore (1959)** used a theorem of Shannon that the minimum weighted path length of a non-alphabetical tree is between H and $H + 1$ to prove that if the tree is restricted to be alphabetic then this intervals extends to $[H, H + 2]$. Refined upper bounds are derived in **Nakatsu (1991)**, **Sheinwald (1992)**, **Yeung (1991)**, **De Prisco and De Santis (1993)** and **Bose and Douïeb (2009)**.

Binary search trees. There are N names $A_1, \dots, A_N$ and $2N + 1$ frequencies $\beta_1, \dots, \beta_N, \alpha_0, \dots, \alpha_N$ with $\sum \beta_i + \sum \alpha_i = 1$. β_i is the frequency of A_i, and α_j is the frequency of names located between A_j and A_{j+1}. A binary search tree has N interior nodes corresponding to the given names and $N + 1$ leaves corresponding to the intervals. An algorithm based on a search tree assumes a three-way comparison (asking if the present key is equal to, less than, or greater than the search key). It generalizes the two-way search associated with alphabetic binary trees when $\beta_1 = \cdots = \beta_n = 0$. **Andersson (1991)** shows how to perform the search applying a single two-way comparison at each internal node and when an external node is reached a final equality comparison is performed. **Spuler (1993)** and **Hu and Tucker (1998)** use this idea to solve this problem by the alphabetic-tree algorithm.

2.2 Prefix-free code formulation of dichotomous search

Let $\{\sigma_1, \dots, \sigma_t\}$ be a set of *characters*. *Word* v is a *prefix* of word $v' \neq v$ if $v' = vu$. A *prefix-free code* is a collection of words $C = \{v_1, \dots, v_N\}$ such that for all $i \neq j$ v_j is not a prefix of v_i. $cost(v)$ is the number of characters in v. Given probabilities $p_1, \dots, p_N$, the cost of C is $\sum_{i=1}^{N} p_i cost(v_i)$. The minimum-cost prefix-free problem is equivalent to the minimum weighted path-length t-ary tree problem where $cost(v_i)$ is the path length of node v_i and p_i is the weight attached to node v_i.

The *alphabetic* coding problem additionally requires that the alphabetic order of the codewords preserves the given order of the words to be encoded. It is equivalent to the alphabetic t-ary tree problem.

2.3 Basic algorithms

Gilbert and Moore (1959) present an $O(N^3)$ dynamic programming algorithm. Let $|T|_{i,j}$ denote the cost of an optimal tree for the interval of uncertainty $\{i,\ldots,j\}$, $i \le j$, and let $W_{i,j} = w_i + \cdots + w_j$. Then:

$$\begin{aligned} &|T|_{i,i} = W_{i,i} = w_i \text{ for } 0 \le i \le n, \\ &|T|_{i,j} = W_{i,j} + \min_{i<k\le j}(|T|_{i,k-1} + |T|_{k,j}) \text{ for } 0 \le i < j \le n. \end{aligned} \tag{2.1}$$

Knuth (1971) proved a monotonicity property which enables reducing the algorithm's complexity to $O(N^2)$. We elaborate on this method in §2.5.

Hu and Tucker (1971) present an $O(N \log N)$ *T-C algorithm* for optimal alphabetic trees. A *combination phase* constructs a tree T' which does not necessarily conform with the ordering restriction. It starts with the initial sequence $V_1,\ldots,V_N$ of terminal *square nodes*, and successively generates $N-1$ new *construction sequences* by combining and replacing a pair of nodes by a parent *round node* in each step. The parent then takes the position of its left child in the construction sequence.

Two nodes are *tentative-connecting* (T-C for short) if the sequence of nodes between them is empty or consists entirely of round nodes. The pair chosen to be combined is the pair of T-C nodes having the minimum sum of weights, said to be a *local minimum compatible pair* (lmcp).

The second phase of the algorithm converts T' into an alphabetic tree T'_N with the same level set and cost.

Hu and Tan (1972) show that when the weights are monotonically increasing, the algorithm of Huffman (1952) produces an (optimal) alphabetic tree. **Hu (1973)** provides another, simpler, proof of the T-C algorithm.

Garsia and Wachs (1977) propose an $O(N \log N)$ algorithm, closely related to the Hu and Tucker algorithm. **Kingston (1988)** provides a simpler proof of correctness of the Garsia-Wachs algorithm.

Belal, Selim, and Arafat (2002) compute an optimal alphabetic tree by recursively merging optimal trees on subsets of the nodes. At each step N/k disjoint sublists, each containing k nodes, are merged into $N/(2k)$ sublists each containing $2k$ nodes. The algorithm has complexity of $O(N \log N)$.

Belal, Selim, and Arafat (2004) present an $O(N)$-time algorithm for inserting an element into an optimal alphabetic tree with N external nodes, keeping the resulting $(N+1)$-leaves tree optimal.

Algorithms with improved complexity for special cases are presented in **Klawe and Mumey (1995)**, **Hu and Morgenthaler (1996)**, **Larmore and Przytycka (1998)**, and **Hu, Larmore and Morgenthaler (2005).**

2.4 Approximations

Allen (1982) shows that the cost errors of the following three closely related heuristics are not bounded by constants.

- **Weight-balanced tree:** Knuth (1971); Rissanen (1973); Leipälä (1979). The next query is chosen so that the weight difference of the left and right subtrees is minimal.
- **Bisection tree:** Mehlhorn (1977). The i-th query is placed near the $k/2^i$ percentile, for the value of k resulting from the search.
- **Min-max tree** Bayer (1975). The query is placed so as to minimize the maximum weight of its left and right sub-intervals.

Larmore (1987) provides two $O(N^{1.6})$ algorithms. One algorithm approximates the solution within $o(1)$ error, and the other one computes the optimal solution when for every $i = 1,\ldots,N$, the probability that the target is at i is at least ϵ/N for some $\epsilon > 0$.

Hwang and Tsai (2003) derive bounds and asymptotic approximations for the sequence $f(n)$ defined recursively by $f(n) = \min_{1\leq j<n} \{g(j, n-j) + f(j) + f(n-j)\}$. Functions $g(x, y) = ax + by$ and $g(x, y) = ax(x + y) + b(x + y)$ appear in dichotomous search problems with direction-dependent costs (see §4.1) and travel costs (see §4.2).

2.5 Knuth's method

Knuth (1971) refines the $O(N^3)$ algorithm of Gilbert and Moore (1959) and reduces its running time to $O(N^2)$. Let $R_{i,j}$ be the minimizer of (2.1). Then, there is always a solution satisfying $R_{i,j-1} \leq R_{i,j}$ and $R_{i,j} \leq R_{i+1,j}$ for $0 \leq i < j - 1 < n$. Thus only $R_{i+1,j} - R_{i,j-1} + 1$ values need to be examined when $R_{i,j}$ is calculated and the calculations are conducted in increasing order of $j - i$. Summing for fixed $j - i$ gives a telescopic series, therefore the time complexity is $O(N^2)$.

Yao (1980) (see also Yao (1982)) proves the following result: Consider the following function c, defined for $1 \leq i \leq j \leq N$:

$$\begin{aligned} &c(i,i) = 0 \\ &c(i,j) = w(i,j) + \min_{i<k\leq j} \{c(i,k-1) + c(k,j)\} \quad \text{for } i < j. \end{aligned} \tag{2.2}$$

Suppose that the function w satisfies the following conditions:

- *Quadrangle inequalities:* $w(i, j) + w(i', j') \leq w(i, j') + w(i', j)$ for all $i \leq i' \leq j \leq j'$. (Such a function is also called *supermodular.*)
- $w(i', j) \leq w(i, j')$ for $i \leq i' \leq j \leq j'$.

Then the monotonicity property holds with respect to c.

Hassin and Henig (1993) generalize Yao's results by considering more general cost functions. Define Problem (m, n) as the instance an object is known to be located in the interval $\{\min(m, n), \max(m, n)\}$. Two types of search costs are considered:

- $D^l(m, n, k)$ for placing the l-th query at k in Problem (m, n).
- C_{li} if the object is discovered at i after l queries.

Denote $p_{ij} = p_i + \cdots + p_j$, where p_i is the probability the object is at i. Let $F^l(m, n)$ be the minimum expected cost for Problem (m, n) when l queries have already been placed. Then, for $l = 0, \ldots, N-1$, $m = 1, \ldots, N$, $F^l(m, m) = C_{lm}$, and for $m < n$ and $l = 1, \ldots, N - (n - m)$:

$$F^{l-1}(m,n) = \min_{m\leq k<n} \left\{ D^l(m,n,k) + \frac{p_{mk}}{p_{mn}} F^l(k,m) + \frac{p_{k+1,n}}{p_{mn}} F^l(k+1,n) \right\}$$

and for $m > n$ and $l = 1, \ldots, N - (m - n)$:

$$F^{l-1}(m,n) = \min_{n\leq k<m} \left\{ D^l(m,n,k) + \frac{p_{nk}}{p_{nm}} F^l(k,n) + \frac{p_{k+1,m}}{p_{nm}} F^l(k+1,m) \right\}.$$

The minimum cost of the search is $F^0(1, N)$, and the complexity of the algorithm is $O(N^4)$. It follows from Knuth (1971), that this can be reduced to $O(N^3)$ when the following property is satisfied:

The monotonicity property: If $F^l(m, n-1)$ is minimized at k', then for some $k \geq k'$, $F^l(m, n)$ is ,minimized at k.

The authors prove the monotonicity property assuming submodularity of $d^l(m, n, k) = D^l(m, n, k)p_{mn}$ and that $c_{lm} = C_{lm}p_m$ is nonnegative and nondecreasing convex in l. (Knuth proved the monotonicity property when C is a linear function of l and independent of i, and D is constant.)

Generalizations of the basic alphabetic search that fit this model include restricted number of queries, travel costs (possibly depending on direction), costs depending on the sign of deviation costs depending on the location of the query, and parallel (t-ary) search. Some authors provide direct proofs for special cases of

Hassin and Henig's general model. These include Fujiwara and Jacobs (2014); Gotlieb (1981); Itai (1976); Schulz (2008); Wessner (1976).

Hinderer and Stieglitz (2000) extend Hassin and Henig (1993). They apply lattice programming results and derive weaker conditions for the applicability of Knuth's method. They consider a family of problems SPD_K of dichotomous search of at most K queries in the interval of integers $[1, N]$. They present a method for problem SPD_K that derives natural conditions under which at each stage k, $1 \leq k \leq K$, the smallest optimal search location in $[i, j]$ increases in both i and j.

2.6 Applications

Garey and Hwang (1974) investigate *group-testing* procedures that isolate a single defective item within a set of N items. Item i is defective with an a priori probability p_i. A group test is a test of any set of items which determines whether all members of the set are non defective.

An optimal testing can be obtained by ordering the items so that $p_1 \geq p_2 \geq \cdots \geq p_N$ and constructing an optimal alphabetic binary tree for the sequence of weights $w_1, w_2, \ldots, w_N$ with w_i proportional to $p_i \prod_{j=1}^{i-1} [1 - p_j]$.

Anily and Hassin (1989) investigate the problem of computing a K-best alphabetic binary tree. The problem arises, for example, when there is no efficient algorithm known for constructing the best tree under certain constraints. We can then rank the trees until the best tree obeying the constraints is reached. The authors develop two algorithms for this problem, with complexities $O(KN^3)$ and $O(KN^4)$.

3. Uniform probability distribution

3.1 Algorithms

The *Fibonacci numbers* are defined by $u_0 = u_1 = 1$ and $u_i = u_{i-1} + u_{i-2}$ for $i \geq 2$.

The *Fibonacci search* is as follows: If at some point in the process the item is isolated to an interval of size u_i beginning at A, then inspect $A + u_{i-1}$. If the item is to the left, then it is isolated to an interval of size u_{i-1}, otherwise it is in an interval of size u_{i-2}. The expected number of comparisons while searching a list of N elements (N is some Fibonacci number), each having an equal probability to be the searched target, is $O(\log_2 N)$, and the maximum search time is $O(\log_\varphi N)$ where $\varphi = (1 + \sqrt{5})/2$ is the "golden section."

Ferguson (1960) analyzes the performance of the Fibonacci search when N is a Fibonacci number. The motivation for exploring the Fibonacci search is to replace divisions by additions and subtractions, and **Nishihara and Nishino (1987)** also note another possible advantage, that it requires a smaller "travel distance" relative to binary search.

Wong (1964) derives optimal solutions for the search problem with a uniform a priori distribution. (See also Gottinger (1977).) Three possible outcomes are possible when comparing x with x_i: $x > x_i$, $x < x_i$, or $x = x_i$. Let $n^*(N)$ be the set of optimal first-step comparisons. Then:

For $N = 2^{k+1} + 2m$, if $m < 2^{k-1}$ then $n^*(N) = \{2^k, 2^k + 1, \ldots, 2^k + 2m + 1\}$. If $m \geq 2^{k-1}$ then $n^*(N) = \{2^k + 2m + 1, 2^k + 2m + 2, \ldots, 2^{k+1}\}$.

For $N = 2^{k+1} + 2m - 1$, if $m \leq 2^{k-1}$ then $n^*(N) = \{2^k, 2^k + 2, \ldots, 2^k + 2m\}$. If $m > 2^{k-1}$ then $n^*(N) = \{2m, 2m + 1, \ldots, 2^{k+1}\}$.

For example, for $N = 2^4 + 9 = 25$, $n^*(N) = \{9, \ldots, 16\}$. The author also computes the optimal value produced by applying the optimal strategies described above.

Morris (1969) uses the dynamic program (3.1) and the convexity property of the function $G(N) = NF(N)$ and proves the lower bound of $\lceil \log_2 N \rceil$ for optimal expected search-cost $F(N)$: $F(1) = 0$, and for $N >$

1

$$F(N) = 1 + \min_{k=1,\dots,N-1} \left\{ \frac{k}{N} F(k) + \frac{(N-k)}{N} F(N-k) \right\}. \tag{3.1}$$

If N is a power of 2 then $F(N) = \log_2 N$. The function $G(N)$ is piecewise linear and coincides with $N \log_2 N$ at the points N which are powers of 2: $G(2^l + j) = l \cdot 2^l + j(l+2)$ for $0 \le j \le 2^l$ (see Carlitz (1971)).

Overholt (1973) shows that the average length of the Fibonacci search exceeds ordinary binary search by approximately 4% and also has a much greater maximum search length and standard deviation. In contrast to ordinary binary search where the greatest search length is one or two tests longer than the mean search length, the Fibonacci maximum search length is nearly 40% greater than the mean.

3.2 Search for the maximum of a unimodal function

A search procedure based on Fibonacci numbers can be used in approximating the maximum of a unimodal function when N f-evaluations are available. A *unimodal* function satisfies for some x that $f(y)$ is strictly increasing for $y \le x$ and strictly decreasing for $x \le y$. The maximum point x is the search argument. The property of unimodality enables, after two evaluations of f, obtaining a smaller interval of uncertainty regarding x. The method divides the initial interval, whose length itself is a Fibonacci number, to two intervals such that the proportion of their lengths is the proportion of sequential Fibonacci numbers. **Kiefer (1953)** shows that Fibonacci search minimizes the maximum possible size of the interval of uncertainty. **Oliver and Wilde (1964)**, **Avriel and Wilde (1966)**, **Karp and Miranker (1968)** and **Rastsvetaev and Beklemishev (2002)**, and **Hassin (1981)** modify the algorithm for cases of symmetric final query, finite accuracy, parallel computations, and initial interval lengths that are not Fibonacci numbers, respectively. When the number of function evaluations is large one can use the asymptotic *golden-section* approximation.

The optimality of the Fibonacci search is often ignored and authors select the queries in a less efficient way. For example **Dahmani, Hifi, and Wu (2016)** also maintain one internal point with a known value of the function, but the next query is placed at the middle of the longer side of the search interval. In another example, **Lei, Jasin, and Sinha (2014a)** divide the search interval $[x_l, x_u]$ to three equal parts and place the next queries at two new points $(2x_l + x_u)/3$ and $(x_l + 2x_u)/3$.

3.3 Interpolation search

Suppose $x_1 < \dots < x_N$ is a random sample from a distribution with cdf F. Given a target value $x \in \{x_1, \dots, x_N\}$ one can compute the probability that $x = x_i$ $i = 1, \dots, N$ and construct an optimal alphabetic tree. **Peterson (1957)** proposed interpolation search, a heuristic that simulates human search through a dictionary: Let $\alpha = F(x)$. The search is directed by starting with the natural guess at α's percentile of the interval of uncertainty, learning whether that target item is smaller, equal, or greater, and updating the percentile accordingly. **Yao and Yao (1976)** and **Perl, Itai and Avni (1978)** prove that the *expected* number of comparisons is of order $\log\log N$, and this is asymptotically optimal. **Manolopoulos, Kollias, and Hatzopoulos (1986)** assume that M given sorted values are searched for. Binary search is conducted for these values in increasing order and each result serves as a lower bound for the initial interval of uncertainty of the next item. **Manolopoulos, Kollias, and Burton (1987)** combine this idea with interpolation search. **Santoro and Sidney (1985)** and **Bonasera, Ferrara, Fiumara, Pagano, and Provetti (2015)** combine ideas from interpolation and binary search.

3.4 Exponential costs

Baer (2010) solves the problem of minimizing $\log_a \left(\sum_i w_i a^{l(i)}\right)$ when $a < 1$. (See Hu, Kleitman and Tamaki (1979) for the case $a \ge 1$.) Note that when $a < 1$, $\log_a(x)$ is monotone decreasing and therefore the objective is to *maximize* $\sum_i w_i a^{l(i)}$.

An $O(N^3)$ dynamic program is straightforward: Let $W_{j,k}$ be the maximum tree weight for items j through k. Then $W_{j,j} = w_j$ and for $j < k$:

$$W_{j,k} = a \max_{s\in\{j+1,\ldots,k\}} \left[W_{j,s-1} + W_{s,k}\right].$$

An example demonstrates that Knuth's monotonicity fails for $a < 1$, and the author constructs approximation algorithms, similar to Hassin (1984). The problem's variables are the leaf-levels $l(i)$ that must satisfy $\sum_i 2^{-l(i)} \le 1$ and integrality constraints, and the tree must be alphabetic. The approximate solution is obtained by relaxing the last two requirements and rounding up the resulting solution. Finally, the algorithm generates from the obtained non-alphabetic tree an alphabetic tree with approximately the same cost.

3.5 Worst order of leaves

Kleitman and Saks (1981) solve the following problem: given a leaf set $E = \{e_1, \ldots, e_N\}$, with weights $w_1 \le \cdots \le w_N$, what order of E maximizes the minimum cost of an alphabetic tree? They show the most expensive order is the *sawtooth order*: $e_1, e_N, e_2, e_{N-1}, \ldots$. **Chu (1985)** shows that the sawtooth order is also the most expensive sequence for the K-restricted alphabetic binary tree (see §4.6).

4. Other costs and objectives

4.1 Asymmetric (direction-dependent) costs

In many applications, there is a different cost (say α) if the target is to the left of the query and a different cost (say β) if it is to its right. The tree associated with this problem is an (α, β) (*lopsided) tree.* In a more general context, the problem is to compute a t-ary tree and the cost of the edge from a parent to its i-th child is c_i. For example, the cost associated the i-th symbol of the alphabet.

The models discussed in §7.1 can be viewed as having asymmetric costs. See also Abigadol and Ben-Tal (1985) on a search for the smallest root among a set of functions with asymmetric costs, a given budget. See §5.5.1 for parallel-search models with asymmetric costs.

Cameron and Narayanamurthy (1964) represent the target by a point from a uniform distribution on an interval. The cost of a query is 1 if the target is to the left of the query and $k > 1$ otherwise. The problem is to locate the target within a unit-length interval at a minimum expected cost. A policy is represented by a function $g(x)$, $x > 1$, such that the query divides an interval of length x in the ratio $g(x) : 1 - g(x)$. Let $f(x)$ be the expected cost of an optimal policy for the interval $[0, x]$. Then

$$f(x) = \begin{cases} 0 & x \le 1, \\ \min_{g(x)} g(x)\big[1 + f(xg(x))\big], [1 - g(x)][k + f(x(1 - g(x)))]\big] & x > 1. \end{cases}$$

Considering functions $g(x)$ that approach a constant value r asymptotically, then for a large x

$$f(x) = \min_r \left[r[1 + f(rx)] + (1 - r)[k + f((1 - r)x)]\right].$$

The optimal value is then

$$f(x) = p\ln(x) + c, \tag{4.1}$$

with p calculated by:

$$p\log(1 - r^{1-k}) = k, \tag{4.2}$$

and r uniquely determined by $r^k + r = 1$. There is a unique positive p that satisfies equation (4.2) and lies in the range $0 < p < \frac{k}{\log 2}$. The resulting heuristic divides the interval in the ratio $r : 1 - r$ until the search terminates.

Murakami (1971) derives an explicit representation of the constant c for the function $f(x)$ in (4.1):

$$c = 2 - \frac{1}{1-r} + p\ln(r + k(1-r)).$$

The author also explicitly presents an optimal search strategy. Consider the equivalent problem where the search is for an object located uniformly in an interval of length 1 with the objective being to find it in an interval of length $\frac{1}{n}$ for a given n. $E(n,x)$ is defined to be the expected cost of the search when first selecting a test point x and thereafter using an optimal policy. The objective function is then $f(n) = \min_{0\le x\le n} E(n,x)$. For the exact optimal solution series $N(i)$ and $g(n)$ are computed:[4]

$$N(i) = \begin{cases} 1 & i = 2-k, 3-k, \ldots, 0, 1. \\ N(i-1) + N(i-k) & i = 2, 3, 4 \ldots \end{cases}$$

$$g(n) = \begin{cases} 1+k & n = 1 \\ g(n-1) + \delta(n) & n = 2, 3, ... \end{cases}$$

where $\delta(n) = 1$ if there exists an integer j satisfying the equation $n = N(j)$ and $\delta(n) = 0$ otherwise. Then:

$$f(n) = g(n) - \frac{N(g(n))}{n}, \quad \text{for } n = 1, 2, ...$$

The author also presents the set of optimal test points for any fixed n.

Choy and Wong (1977) provide a graphical description of a linear time algorithm for the optimal (α, β) tree problem. These authors extend in Choy and Wong (1978) the method to obtain an $O(N)$ algorithm under a constraint on the number of consecutive α-edges.

Horibe (1982) (see also **Ottmann, Rosenberg, Six and Wood (1984)**) shows that the optimal $(1,2)$-tree with F_k terminal nodes, T_k, has F_{k-1} terminal nodes of cost $k-2$ and F_{k-2} nodes with cost $k-1$, where F_k is the k-th Fibonacci number. The tree T_{k+1} is obtained from T_k by splitting all terminal nodes of cost $k-2$ in T_k. In **Horibe (1983)** the author characterizes the values c such that the Fibonacci tree with F_k terminal nodes is an optimal $(1,c)$ tree.

Shing (1983) assumes that the right and left penalties depend on the (uniformly distributed) location of the searched object. The $O(N^3)$ complexity of the dynamic program is reducible to $O(N^2)$ by using the monotonicity property.

Kapoor and Reintgold (1989) consider both min-max and min-sum versions of the optimal (α, β)-tree problem. A greedy algorithm similar to Varn (1971) can also solve the min-max binary problem. In particular, if the optimal root for a problem with N terminal nodes is k then it is k or $k+1$ for the problem with $N+1$ terminals. It follows that there is a common optimum tree for both the min-max and min-sum problems.

Hinderer (1990) assumes $\frac{\beta}{\alpha} = \frac{m}{k}$ for positive integers m and k. Let $f(s)$ be the minimal expected search cost for an object initially hidden in $N_s = \{1, 2, \ldots, s\}$, $s \ge 2$. Let $D^*(s)$ be the set of minimizers of $f(s)$, and let

$$N(i) = \begin{cases} 1 & \text{if} \quad 2 - \max(m,k) \le i \le 1 \\ N(i-m) + N(i-k) & \text{if} \quad i \ge 2. \end{cases}$$

Define $H(s) = sf(s)$ and let $S_i = \{s \in \mathbb{N} \,|\, N(i) \le s < N(i+1)\}$. The main theorem states:

(1) If $i \ge 2$ and $S_i \ne \emptyset$, then $H(s+1) - H(s) = i + m + k - 1$ for $s \in S_i$.

[4]The formula assumes that k is a positive integer. This assumption is not required in Cameron and Narayanamurthy (1964).

(2) $D^*(s) = \{s \in \mathbb{N} \,|\, L(s) \leq s \leq M(s)\}$, where for $i \geq 2$ and $s \in S_i$
$L(s) = \max(N(i-m), s - N(i-k+1))$ and $M(s) = \min(N(i-m+1), s - N(i-k))$.

Baer (2007) assumes a general probability distribution for the target's location and that the cost of a query is c_0 or c_1 depending on whether the object is to the left or to the right of the query. A special feature of this model is that the decision maker is free to decide for each location which one of C_0 and C_1 will correspond to each outcome of the query. The optimal tree and assignment of costs can be computed by dynamic programming in $O(N^3)$ time. An example demonstrates that Knuth's monotonicity property doesn't hold for this variation.

Efraimidis (2010) contributes new algorithmic aspects of the problem, highlighting the inherent relation of Fibonacci search with asymmetric-cost search and presenting an efficient algorithm which solves asymmetric-cost binary search for integer (or rational) costs (α, β).

4.2 Search with travel costs

Let ax be the travel cost required for a searcher to move distance x at any stage of the search in any direction, and suppose each query costs b. After each query the search continues from the point of the last query, either being the left or the right end of the new uncertainty interval.

Murakami (1976) views the problem of determining a sequence of queries to minimize the maximum cost required to diminish the existing interval of length N to unit length. Let $h(N)$ denote the cost of an optimal search over an interval of length N, then:

$$h(N) = \begin{cases} 0 & N \leq 1 \\ \min_{0 \leq x \leq n}[ax + b + \max(h(x), h(N-x))] & N > 1. \end{cases}$$

Define $g(N)$ as the unique integer such that $2^{g(N)} < N \leq 2^{g(N)+1}$. Then $h(N) = (N-1)\alpha + (g(N)+1)b$, for $N > 1$.

A.J. Hu (1986) develops a heuristic when the objective is to minimize the expected cost of the search. The a priori distribution of the object's location is uniform. A *uniform partition search* divides $[1, \ldots, N]$ into sublists of uniform size, and travels among sublists from left to right conducting queries such as "does the desirable record lie in sublist x?". When a positive answer is obtained we narrow our search to that sublist. It takes k reads to find that the desired record lies in the k-th sublist. A searcher moving through sublists $(1, \ldots, k)$, each of size $\frac{N}{n}$, pays $\frac{ak}{n}$ for travel expenses.

An integer p, $2 \leq p \leq N$, denotes the number of parts into which the list is divided. The expected cost of the search is $f(p) = br(p) + at(p)$, where

$$r(p) = \lceil \log_p N \rceil (p+2)(p-1)/(2p)$$

is the expected number of queries and $t(p) = N(p+2)/(2p)$ is the expected distance traveled. The optimal p satisfies

$$\frac{p^2}{\ln p} = \left(\frac{a}{b}\right) \frac{(\sqrt{N})^2}{\ln(\sqrt{N})}.$$

Hu and Wachs (1987) consider a discrete interval and queries"is $x < m, x = m$, or $x > m$." Every unit distance movement costs one unit, as does the cost of a single query, that is, $a = b = 1$. Let $T(n)$ be the cost of the search when the searcher is at n and the target is known to be in $\{0, 1, \ldots, n-1\}$, then

$$T(n) = \begin{cases} \min_{0 \leq m < n}(m+2)n + T(m) + T(n-m-1) & \text{if } n > 0 \\ 0 & \text{if } n = 0. \end{cases}$$

The solution is unique for certain values of N that are recursively defined and in these cases the solution is a *tape-complete tree* where the subtrees hanging on the right-most path are of non-increasing sizes m_i and the number of subtrees of size i is 2 or 3 except for the highest size that can also be 1. The optimal trees for intermediate values of N are obtained by adding one leaf on each subtree of a subset of allowable trees until we get the next tape-complete tree.

The optimal tree can be efficiently computed using the property that the optimal location for the first query when N increases by 1 either remains unchanged or increases by 1.

The authors also consider the more general problem where each movement costs a, each comparison costs b, and $2a/b$ is an integer. Also here an optimal tree with $N+1$ nodes can be obtained by attaching a leaf to the optimal tree with N nodes, thus giving an $O(N)$ algorithm.

Nishihara and Nishino (1987) assume costs $a = 1$ and $b = 0$, i.e., moving one unit of distance costs 1, while conducting a query is free. The objective is to minimize the expected cost of the search. This special case is trivially solved by inspecting the points $1, \ldots, N$ sequentially. Three sub-optimal algorithms are compared: binary search (BS), Fibonacci search (FS) Ferguson (1960), and movement-minimizing Fibonacci search (mFS) which modifies the basic FS algorithm such that the amount of movement is kept small by moving "lazily".

Wachs (1989) considers the following problem: For locations $\{\alpha_1, \ldots, \alpha_N\}$, $0 \le \alpha_1 \le \alpha_2 \le \cdots \le \alpha_N$, p_i is the probability that the object lies at α_i and let q_i be the probability that the search argument lies between α_i and α_{i+1} (with obvious definitions for q_0 and q_N). A query costs b and reveals if the object lies at the searched location, to its left, or to its right. The cost of traveling from α_i to α_j is $a\,|\alpha_i - \alpha_j|$. The goal is to find a search strategy with expected minimal cost.

Let $u(i,j) = a(\alpha_j - \alpha_i) + b$. Let $c(i,j)$ and $d(i,j)$ be the minimum cost of a search strategy for the reduced problem on locations $\{\alpha_i, \ldots, \alpha_j\}$, $i < j$, such that the search begins at α_i and α_j, respectively. Let $w(i,j) = p_{i+1} + \cdots + p_{j-1} + q_i + \cdots + q_{j-1}$ for $0 \le i < j \le N+1$. The following $O(N^3)$ recurrence solves the problem:

$$\begin{aligned}
&c(i,j) = d(i,j) = 0 && \text{for } i = j-1,\\
&c(i,j) = \min_{i<k<j}\left[u(i,k) + \tfrac{w(i,k)}{w(i,j)}d(i,k) + \tfrac{w(k,j)}{w(i,j)}c(k,j)\right] && \text{for } i < j-1,\\
&d(i,j) = \min_{i<k<j}\left[u(k,j) + \tfrac{w(i,k)}{w(i,j)}d(i,k) + \tfrac{w(k,j)}{w(i,j)}c(k,j)\right] && \text{for } i < j-1.
\end{aligned}$$

For $u(i,j) \equiv 1$, this equation has the form of equation (2.2) and thus Yao's result applies to this problem. Wachs extends Yao's result to a problem where $u(i,j)$ is not identically 1 enabling a reduction of the complexity to $O(N^2)$. This result is further generalized in **Hassin and Henig (1993)**. **Hornick, Maddila, Mücke, Rosenberg, Skiena and Tollis (1990)** examine heuristics for the problem.

A *block search algorithm* with parameter r partitions the interval of uncertainty to blocks of size rN, and sequentially places queries at the last point of each block. Once the block containing the target is found, the search continues by applying binary or linear search.

The optimal r for the binary search is asymptotically equal to $r^* = \sqrt{b/(3aN)}$ and the expected cost is $\bar{S}_{BB(r^*)}(N) = a(N/2) + \sqrt{3baN} + o(\sqrt{N})$.

The optimal r for the linear search is $r^* = \sqrt{\frac{b}{N(2a+b)}}$ and the expected cost is: $\bar{S}_{BL(r^*)}(N) = a(N/2) + \sqrt{b(2a+b)N} + \frac{a}{2} + b$.[5]

Hassin and Hotovely (1992) denote by $F_\pi(N)$ the expected cost associated with searching an interval of length N while using policy π. A policy π is *asymptotically optimal* if $\lim_{N\to\infty} \frac{F_\pi(N)}{F(N)} = 1$. If an arbitrary function $g(N)$ is an approximation for another function $f(N)$, then the *relative error* of this approximation is $\left|\frac{f(N)-g(N)}{f(N)}\right|$. The authors analyze the performance of several approximations: fixed-step policies, fixed-ratio policies and myopic policies.

According to *fixed step policy*, queries are placed at fixed step sizes as long as the direction of movement is fixed. A step of size $\sqrt{Nb/a}$ is asymptotically optimal, and its relative error converges to 0 as fast as

[5] The authors also considered the *fixed-ratio heuristic*, which was independently analyzed in Hassin and Hotovely (1992), but their formula for the cost function contains a redundant factor of 2 which erroneously led to the conclusion that it is dominated by the linear search version of block search.

$1/\sqrt{N}$. The expected cost of this policy satisfies

$$h(N) = 0.5aN + \sqrt{Nba} + o\left(\sqrt{N}\right). \tag{4.3}$$

According to the *fixed-ratio policy*,[6] given an interval of uncertainty of size $N > 1$, the next query is placed at a distance of Np from the searcher's location. The fixed-ratio policy with $p = \frac{2}{\sqrt{N}}\sqrt{\frac{b}{a}}$ is asymptotically optimal and its cost also satisfies (4.3). The proof of this result employs the notion of *binary entropy* $H(p) = -p\log p - (1-p)\log(1-p)$. The expected number of queries for a fixed-ratio policy with parameter p, $S(N,p)$, fits the following approximation for $p < \frac{1}{3}$:

$$\left|S(N,p) - \frac{\log N}{H(p)}\right| < \frac{1}{pH(p)}\log\frac{4}{p}.$$

As a corollary, the relative error of the approximation $\frac{\log N}{H(p)}$ to $S(N,p)$ converges to 0 at least as fast as $\frac{1}{\log N}$.

Let $D(N,p)$ be the expected travel distance under a fixed-ratio policy with parameter p. It is shown that the relative error of the function $d(N,p) = \frac{N-1}{2(1-p)}$ to $D(N,p)$ converges to 0 at least as fast as $\frac{\log N}{N}$.

Define $\widehat{F}(N,p) = b\frac{\log N}{H(p)} + ad(N,p)$. Differentiating with respect to p, equating to zero and letting $N \longrightarrow \infty$ gives $p = \frac{2}{\sqrt{N}}\sqrt{\frac{b}{a}}$ and $\widehat{F}(N,p) \approx \sqrt{Nab} + aN/2$, which is asymptotically optimal.

Myopic policies maximize the reduction in the interval's length per unit of query cost. By investing $b + xa$ the size of the interval reduces to x with probability $\frac{x}{N}$, and to $N - x$ with probability $\frac{N-x}{N}$. A myopic policy maximizes the expected reduction in the size of the interval $\frac{x}{N}(N-x) + \frac{N-x}{N}x$ per unit of cost. The optimal result for this class of policies is achieved with a first step of $\lfloor\sqrt{Nb/a}\rfloor$. The cost of this policy is of order $aN + 2\sqrt{Nba}$.

Chung, Chen, and Lin (1992) analyze the same special case as Nishihara and Nishino (1987) and study the expected costs of the same heuristics. Assume $N = F_n - 1$ where F_n is the n-th Fibonacci number. Then the expected costs of the sequential search, BS, FS, and mFS are asymptotically equal to $0.5F_n$, F_n, $0.882F_n$ and $0.809F_n$, respectively.

Navarro, Barbosa, Baeza-Yates, Cunto, and Ziviani (2000) investigate heuristics when $w(i,j)$ is the cost of placing a query at j given that the last query was placed at i. They analyze the problem assuming that the probability that the location chosen for the next query is uniformly distributed over the interval of uncertainty, and describe an application to text retrieval.

Szwarcfiter, Navarro, Baeza-Yates, Oliveira, Cunto, and Ziviani (2003) present an $O(N^{k+2})$ algorithm for a generalization of travel costs where the cost of placing a query depends not only on the previous query but on the location of the k preceding locations.

See also Navarro, Barbosa, Baeza-Yates, Cunto, and Ziviani (2000).

4.3 Location-dependent search costs

Knight (1988) assumes that with the same probability $1/(N+1)$ the target lies in any of the locations $\{1,\ldots,N\}$ or not in any of them. The cost of search at k is $P(k)$. Let T_N be the binary search tree corresponding to a unique search strategy. There are N internal nodes at which the item may be located and $N+1$ leaves.

[6] The asymptotic formulas for the number of comparisons and the traveled distance for this policy have also been obtained in an unpublished manuscript **Hofri (1987)**.

$W_k(T_N)$ denotes the number of internal nodes in the subtree of T_N rooted at k. The expected cost of the search using T_N is $\frac{1}{N+1}\sum_{k=1}^{N} P(k)W_k(T_N)+\frac{1}{N+1}\sum_{k=1}^{N} P(k)$. Discarding the constants $\frac{1}{N+1}$ and $\sum_{k=1}^{N} P(k)$, the remaining sum $S(T_N)=\sum_{i=1}^{N} P(i)W_i(T_N)$ is called the *search cost of* T_N.

For an inspection cost function $P(k)=\alpha k+\beta$ ($\alpha,\beta \geq 0$):

$$S(T_N) \geq \frac{\alpha}{2}(N+1)^2\left[\log(N+1)-1.070\right]+\frac{\alpha}{2}(N+1)+\beta\left[(N+1)\log(N+1)-N\right].$$

For a polynomial function $P(k)=k^p$ where p is a positive integer:

$$S(T_N) \geq \frac{1}{p+1}(N+1)^{p+1}\log(N+1)-(N+1)^{p+1}.$$

For these cases, ordinary binary search is nearly optimal.

Damaschke (1998) describes a situation where the concentration threshold d allowing a substance A to dissolve in a solvent B is unknown. Tests start with tubes containing one of these liquids, and at each step the contents of two of the existing tubes are mixed to form a new test point. Given an integer n, the goal is to find i such that $d \in (i/2^n,(i+1)/2^n)$.

The possible sequences of test points is modeled as follows: Initially there are unbounded reservoirs of *pebbles* at 0 and 1. There are two types of steps. A *move* corresponds to selecting two pebbles located at x and y and moving them to $(x+y)/2$. A *test* takes place at a location x that currently hosts a pebble and asks whether $d<x$ or $d>x$. Since n tests are necessary and sufficient, the problem is to minimize the maximum number of required moves. The unique feature of this model is that the cost of a test depends on its location through the current location of the pebbles.

A natural application of the bisection algorithm that greedily moves pebbles before each test requires approximately $n^2/6$ moves. However, by pre-planning movements of pebbles, there exists for every fixed ϵ a search strategy with $O(n^{1+\epsilon})$ moves.

Navarro, Barbosa, Baeza-Yates, Cunto, and Ziviani (2000) investigate heuristics when $w(i,j)$ is the cost of placing a query at j given that the last query was placed at i. They analyze the problem assuming that the probability that the location chosen for the next query is uniformly distributed over the interval of uncertainty, and describe an application to text retrieval.

De Bonis, Gargano and Vaccaro (2001) suggest a different formulation of the problem of Damaschke (1998). It is desired to estimate the unknown concentration $c>1$ of a substance. A test reveals whether the concentration rc of the original liquid mixed with water satisfies $rc>1$. The decision variable r is obtained by merging any integer number of units from a given sample with water. The authors compare strategies with respect to the number of tests required to reduce the interval of uncertainty, the number of merge operations, and quantity of water used. They consider both the version where a test destroys the sample and when it isn't.

Laber, Milidiú, and Pessoa (2002) analyze an $O(N^2)$ *ratio heuristic* for the model of Knight (1988). The first comparison is made at k which minimizes $P(k)/\min\{k,N-k+1\}$. The expected cost of the resulting search is at most $4\ln(N+1)\sum P(i)$. The authors present constant-factor linear-time approximation algorithms for both minimum expected cost and minimax cost in **Laber, Milidiú, and Pessoa (2002a)**.

Charikar, Fagin, Guruswami, Kleinberg, Raghavan, and Sahai (2002) provide a three-way comparison algorithm that costs no more than $\log N+O(\sqrt{\log N}\log\log N$ times the cheapest way to verify membership or non-membership of a number q in a sorted list. The cheapest cost is either the cost of a single query verifying that q is a member or the sum of costs of two adjacent entries verifying that q is not in the list.

Cicalese, Jacobs, Laber, and Valentim (2012) show that Knuth's monotonicity does not hold for the min-max problem and therefore cannot be used to reduce the time complexity of the $O(N^3)$ time of the simple dynamic program to $O(N^2)$. The authors derive however a different, more complex, $O(N^2)$ present

$O(N)$-time algorithms with $O(2+\epsilon+o(1))$ approximation factor to both min-sum and min-max versions of the problem.

4.4 Minimax trees

This section contains results concerning alphabetic tree for which $\max_i w_i 2^{l(i)}$, or equivalently $\max_i\{w_i + l(i)\}$, is minimized. In an equivalent formulation of the problem, the weight of each internal node is 1+ the maximum weight of any of its children, and the goal is to construct an alphabetic tree such that the weight of the root is minimal.

Hu, Kleitman and Tamaki (1979) find that the Hu-Tucker algorithm can be modified to construct optimal alphabetic trees for various cost functions including $\sum_i w_i a^{l(i)}$ $(a \geq 1)$ and $\max_i w_i a^{l(i)}$ in the same $O(N \log N)$-time complexity. Their results concerning the min-max problem also applies for ternary (3-ary) trees.

Zhang (1984) solves the problem of minimizing $\sum w(v)$ over the nodes of the tree, where for a leaf v $w(v)$ is its given weight and for an internal node v $w(v)$ is the maximum weight among its two children (rather than their sum as in the common alphabetic tree problem). The algorithm starts with the given ordered sequence of weights, and recursively combines the node with minimum weight and its smaller neighbor and replaces them by a new node with that neighbor's weight.

Klawe and Mumey (1985) present a linear-time algorithm for a minimax full t-ary problem (each internal node has t children) with integer weights. By solving $O(\log N)$ integer instances one can solve the problem with real weights. Also, by solving k integer problems with overall complexity $O(kN)$ it is possible to approximate the solution to the real-weights problem with error at most $\frac{1}{2^{k-1}}$.

Coppersmith, Klawe and Pippenger (1986) allow internal nodes of the tree to have varying degrees at most t. As in Klawe and Mumey (1985), the authors obtain a linear-time algorithm for integer weights and an $O(N \log N)$ algorithm for real weights. They also prove that the cost of the solution is upper-bounded by $1 + \log_t 2 + \log_t \left(\sum_i t^{w_i}\right)$, and this bound is tight.

Gagie (2009) develops an $O(Nd \log\log N)$-time algorithm for the t-ary problem with real weights where d is the cardinality of $\{\lfloor w_i \rfloor \,|i = 1, \ldots, N\}$. The algorithm improves upon the $O(N \log N)$ result of Klawe and Mumey (1985) when d is small.

Gawrychowski (2013) improves the previously known $O(N \log N)$ algorithms for the binary problem and presents an $O(N)$ algorithm. The algorithm is complex but simpler versions exist when the weights w_i are all integer.

4.5 Maximizing the probability of finding a hidden object

See Rivest, Meyer, Kleitman, Winklmann and Spencer (1980) for a model with a limited number of incorrect answers.

Berry and Mensch (1986) assume that a searcher has n queries available, and wishes to maximize the probability of finding an object hidden with probability p_i in $i \in \{1, \ldots, N\}$.[7] The information gathered from each specifies whether the searched object is in that site, to its right, or to its left. The optimal strategy is a bisection strategy on the set of $\min(N, 2^n - 1)$ points with the largest probabilities.

The author also provides partial characterizations of the optimal strategies for special cases of a variation where there is a fixed positive probability that a search at i will reveal that the object lies in some $X < i$ while the correct answer is $X = i$ or $X > i$ (note the asymmetry with respect to left and right).

[7]A similar setting is assumed in **Hinderer and Stieglitz (2000)** where a penalty is imposed if the object is not found after n searches.

4.6 Depth-dependent costs and depth-restricted trees

Note that a constraint on depth is a special case of depth-dependent costs where the costs become very large if the depth exceeds the constraint. A K depth limitation means that the object must be found after at most K queries.

Recall the notation $|T|$ of the weighted path length (cost) of a tree T. Consider an optimal tree T, and an internal node of v. Then, the subtree of T rooted at v is an optimal $(K - l(v))$-restricted tree for its weights.

The basic $O(KN^3)$ algorithm in **Itai (1976)** is the following: Let $T[i, j, k]$, $1 \le i \le j \le n$ and $0 \le k \le K$, denote the optimal tree of depth at most k for $(w_i, w_{i-1}, ..., w_j)$. If $k < \lfloor \log_2(j - i + 1) \rfloor$, then no solution exists and we set $|T[i, j, k]| = \infty$. Otherwise:

$$|T[i,i,k]| = 0 \quad i = 1, 2, \ldots, n,$$
$$|T[i,j,k]| = \sum_{r=i}^{j} w_r + \min_{i \le b < j} \{|T[i,b,k-1]| + |T[b+1,j,k-1]|\},$$
$$i = 1, \ldots, n-1, \quad j = i+1, \ldots, n.$$

Itai (1976) and **Wessner (1976)** reduce the complexity to $O(KN^2)$ by proving that Knuth's monotonicity holds.

Larmore and Przytycka (1994) introduce an $O(NK \log N)$ variation of the T-C algorithm for the restricted-depth problem based on a similar *Package Merge* algorithm for non-alphabetic trees.

The algorithm also solves in $O(N^2 \log N)$ time the depth-dependent problem of computing a tree with minimum weight, where a nondecreasing convex weight matrix $w_{i,l}$, $i = 1, \ldots, N$ is given and the cost of a tree T is $|T| = \sum_{i=1}^{N} w_{i,l(i)}$, where $l(1), \ldots, l(N)$ is the list of leaf depths.[8]

Larmore and Przytycka (1996) develop an $O(K \log N)$-time N-processor parallel algorithm for the optimal alphabetic binary tree with K-restricted depth. This complexity matches the best known sequential time. The authors provide a parallel implementation of the Package Merge procedure (see Larmore and Przytycka (1994)). They also obtain for any constant k an $O(k \log^2 N)$-time N processors algorithm that constructs a tree with cost exceeding that of the optimal tree by at most $\frac{1}{N^k}$ (assuming normalized leaf weights of unit sum).

Gupta, Prabhakar and Boyd (2004) extend Yeung (1991) to the depth-restricted problem. The author presents an $O(N \log N)$ algorithm for constructing an alphabetic tree whose average depth differs from the optimal value by at most 2.

Fujiwara and Jacobs (2014) assume that the cost incurred when the search terminates at leaf i having depth $l(i)$ is $f_i(l(i))$. For minimizing $\max_i f_i(l(i))$ they prove it is sufficient to assume that the cost functions are nondecreasing, i.e., $f_i(x) \ge f_i(x-1)$ for every i and $x > 1$, in order to prove the monotonicity property. The special case with $f_i(l) = (1+d+d^2+\cdots+d^{l-1})$ for a constant $d \ge 1$ is considered by Schulz (2008).

5. Variations of the dichotomous search problem

5.1 Unreliable answers

In many cases the answer to a query is not reliable. In some cases it is possible to conduct independent queries at the same location and use statistical tools to estimate the direction of the target. In other cases this is not possible. See Waeber, Frazier, and Henderson (2013) and §6.4 for some applications.

Michael Horstein (1963) assumes that with a given probability p_0 the answer to a query is false, pointing at the wrong direction. The author suggests a modified bisection algorithm that poses the next query at the median of the posterior density function. **Ben Or and Hassidim (2008)** refine this scheme and analyze

[8]Knuth's monotonicity is proved for achieving this complexity. It also follows from the general results in Hassin and Henig (1993).

its complexity. **Waeber, Frazier, and Henderson (2013)** assume that the location of the target is an absolutely continuous random variable, and prove that the expected value of the query converges to the target's location at a geometric rate.

Rivest, Meyer, Kleitman, Winklmann and Spencer (1980) consider dichotomous search for $x \in \{1, 2, \ldots, N\}$ where up to k of the answers may be incorrect. Let $Q(N, k)$ denote the number of comparisons needed *in the worst case* to identify x, then $Q(N, k) = \log N + k \log\log N + O(k \log k)$.

The authors relate this problem to that of Gal, Bachelis and Ben-Tal (1978) described as identifying the smallest root in $(0, 1]$ of a set of continuous increasing functions g_i where $g_i(0) < 0 < g_i(1)$, $i = 0, \ldots, k$. A query amounts to asking whether $g_i(c) > 0$ for a function g_i and $c \in (0, 1]$. This problem is equivalent to identifying an unknown $x \in (0, 1]$ by testing "Is $x < c$" when up to k of the "No" answers are erroneous (but all "Yes" answers are reliable). It is shown that this search requires in the worst case at least $Q(N, k)$ comparisons.

Aslam and Dhagat (1991) and **Borgstrom and Kosaraju (1993)**derive upper bounds on the length of the search in the *linearly bounded* model where for each initial sequence of i queries there cannot be more than ri errors, for some $0 < r < 0.5$.

Karp and Kleinberg (2007) consider N coins with heads probabilities $p_1 \leq \cdots \leq p_N$. For given the target $0 \leq \tau \leq 1$ and accuracy $\epsilon > 0$, the goal is to identify in a minimum number of coin flips an index i such that $[\tau - \epsilon, \tau + \epsilon] \cap [p_i, p_{i+1}] \neq \emptyset$. The authors provide an optimal (up to a constant) algorithm and describe applications of the model.

Repeated sampling is a common method in stochastic convex optimization. Here, each of a small number of new points in the current search interval is repeatedly sampled and the outcome is used to reduce the interval. See **Agarwal, Foster, Hsu, Kakade, and Rakhlin (2013)** and **Lei, Jasin, and Sinha (2014a)** and their extensive literature reviews for applications of this method to unconstrained and constrained optimization, respectively.

5.2 Delayed and lost answers

Ambainis, Bloch, and Schweizer (2002) assume that the answer to a query is obtained only after additional d queries are posed.[9] They show that the largest interval $\{1, \ldots, B(t)\}$ where the search is guaranteed to success in t queries is computed by

$$B(t) = \begin{cases} 1 & \text{if } t \leq 0 \\ B(t-1) + B(t-d-1) & \text{if } t > 0. \end{cases}$$

This means that $\log_\phi(n) + O(1)$ queries are necessary and sufficient on an interval of size n, where ϕ is the positive real root of $x^{d+1} = x^d + 1$.

Cicalese and Vaccaro (2003) consider a variation of Ambainis, Bloch, and Schweizer (2002) where the answer to one of the queries may be lost. They show that in this case

$$B(t) = \begin{cases} \lfloor 0.5t \rfloor + 1 & \text{if } t \leq d+1 \\ B(t-1) + B(t-d-1) & \text{if } t \geq d+2. \end{cases}$$

5.3 Search for the smallest root in a set of functions

Gal, Bachelis and Ben-Tal (1978) consider search for the left-most object among k objects located on the unit interval. Each query selects an object i and a point $x \in (0, 1)$ and reveals whether the object lies to the left or to the right of x. The searcher is given n queries and wishes to minimize the maximum possible size

[9]This setting is closely related to that of parallel/polychotomous search.

of the interval of uncertainty. Computing the optimal strategy requires solving a high-complexity dynamic program. Therefore they suggest the simplified procedure described below.

Consider a given set of k objects that were not excluded yet by the search and a current interval of uncertainty normalized to $(0,1)$. Select $x \in (0,1)$ and sequentially conduct queries to the items at x . If after $i \leq k$ queries we learn for the first time that the object lies to the left of x the first $i-1$ objects can be excluded and the new interval of uncertainty is $(0,x)$. If all objects lie to the right of x then the new interval of uncertainty is $(x,1)$. This leads to the following dynamic program where $V_k(n)$ denotes the guaranteed size of the final interval of uncertainty relative to the size of the current one:

$$V_k(n) = \min_x \max[xV_k(n-1), xV_{k-1}(n-2), \cdots, xV_1(n-k), (1-x)V_k(n-k)].$$

For all $1 \leq i < k$ $V_k(n-1) \geq V_{k-i}(n-i-1)$ and therefore the recursion simplifies to

$$V_k(n) = \min_x \max[xV_k(n-1), (1-x)V_k(n-k)].$$

The solution is obtained at the point x_n^* where

$$V_k(n) = x_n^* V_k(n-1) - (1-x_n^*)V_k(n-k)$$

so that

$$x_n^* = \frac{V_k(n)}{V_k(n-1)} = \frac{V_k(n-k)}{V_k(n-1)+V_k(n-k)}.$$

Substituting x_n^* in the equation of $V_k(n)$ gives a recursive solution. When $n \to \infty$, the search guarantees an accuracy of $(0.5+\epsilon_k)^n$ where ϵ_k rapidly decreases to 0, almost like the bisection procedure for a single object.

An example demonstrates that randomization can used in order to improve performance.

Abigadol and Ben-Tal (1985) extend Gal, Bachelis and Ben-Tal (1978) allowing for asymmetric costs as in §4.1. The cost incurred is an integer m if the object lies to the left of the query and 1 otherwise. The goal is to guarantee the smallest final interval of uncertainty given the budget n. Let z_i be the location of object i, $i = 1, \ldots, k$, and $z^* = \min_i(z_i)$. The *relevant interval* of z_i, L_i, is a subinterval of $(0,1]$ such that $z_i \notin L_i \implies z_i \neq z^*$. The objective is to minimize the maximum possible final size of a relevant interval.

This definition is illustrated with the case of two components, z_1 and z_2, both having the relevant interval $[a,b]$. "Suppose the next observation is on z_1 at x, $x \in (a,b)$. If the outcome is (+), i.e., x is to the right of z_1, then the relevant interval of both points is $[a,x]$, while if the result is (-), the relevant interval of z_1 is $[x,b]$ and one of z_2 remains $[a,b]$."

For $k=1$, denote $v(n)$ as the maximal length of the relevant interval resulting from an optimal search. Then

$$v(n) = \begin{cases} 1 & n < m \\ \min_{0<x<1} \max\{xv(n-m), (1-x)v(n-1)\} & n \geq m. \end{cases}$$

Let x_n be the minimizer of $v(n)$ and $\lambda_n = \frac{1}{v(n)}$. λ_n is the solution of the following difference equation, and $x_n = \frac{\lambda_{n-m}}{\lambda_n}$

$$\lambda_n = \begin{cases} 1 & n < m \\ \lambda_{n-m} + \lambda_{n-1} & n \geq m. \end{cases}$$

The authors develop an algorithm for the general case and test it numerically.

5.4 Multi-objects search

Hassin and Henig (1984) compute the *jumps* of a step-function $f(t)$ that counts the number of objects in $\{1, \ldots, t\}$, $t \leq N$, using the minimum expected number of queries. (Point t is a jump if $f(t) > f(t-1)$, where $f(0) = 0$.) Let I be the *information function* defined on subintervals of $[0, N]$. Initially, an information $I(0, N)$ is known and an a priori joint distribution of the jumps is given. At stage $m = 1, 2 \ldots$ a list of points $0 = t_0 < t_1 < \cdots < t_m = N$ is given, the information $I(t_0, \ldots, t_m)$ is obtained, and the joint distribution of the jumps is updated. An interval (t_{i-1}, t_i) is *resolved* at stage m if $I(t_0, \ldots, t_m)$ identifies all the jumps in the interval. A strategy is called *optimal* if it minimizes the expected number of queries until $(0, N]$ is resolved. The model imposes two conditions:

- Condition A: Selection of points outside the interval $(t_{i-1}, t_i]$ does not have an effect on whether $(t_{i-1}, t_i]$ is resolved.
- Condition B: The probability an interval $(a, a + d] \subseteq (t_{i-1}, t_i]$ is unresolved given $I(t_0, \ldots, t_m)$ depends only on its length d and is monotone increasing and concave.

The following strategy is optimal under these general conditions: In each stage, select any unresolved interval $(t_{i-1}, t_i]$ and split it at $t_{i-1} + 2^{t(d)}$ or at $t_i - 2^{t(d)}$, where $d = t_i - t_{i-1}$ and $t(d)$ is the unique integer satisfying $3 \cdot 2^{t(d)-1} \leq d < 3 \cdot 2^{t(d)}$.

The authors provide examples satisfying conditions A and B. In the most basic example, the objects are independently uniformly distributed over $(0, N]$. At stage m of the search we discover whether $(t_{i-1}, t_i]$, $i = 1, \ldots, m$, is empty (contains no objects) or not. Thus, $(t_{i-1}, t_i]$ is resolved if it is either empty or $t_i - t_{i-1} = 1$.

Hassin and Megiddo (1985) consider a monotone nondecreasing step function $f : \{0, 1, \ldots, N\} \longmapsto \{0, 1, \ldots, K < N\}$ satisfying $f(0) = 0, f(N) = K$. The objective is to locate all the jumps of f using the minimal number of f-evaluations in the worst case. Obviously, by performing K binary searches one recognizes $f(i)\, \forall i$, so that $K \lceil \log_2 N \rceil$ f-evaluations should suffice.

The number of f-evaluations required for identifying the jumps of f is $K \left\lfloor \log\left(\frac{N}{K}\right) \right\rfloor + \left\lfloor (N-1) 2^{-\log\left(\frac{N}{K}\right)} \right\rfloor$, where $\log(x) = \max(0, \log_2 x)$. The following strategy guarantees the bound: First search at $i = 2^m$ such that $m = \left\lfloor \log\left(\frac{N}{K}\right) \right\rfloor$. Suppose $f(i) = K_1$. Proceed recursively finding all the K_1 jumps of f over the set $\{0, 1, \ldots, i\}$ (if $K_1 > 0$) and $K - K_1$ jumps over $\{i + 1, \ldots, N\}$ (if $K_1 < K$).

Karp (1993) characterizes the solutions for the problem discussed in Hassin and Megiddo (1985). While Hassin and Megiddo (1985) finds an optimal algorithm for this problem, there is usually more than one solution. In fact each $i \in [0, ..., N]$ such that i or $N - i$ is a multiple of $2^{\left\lfloor \log\left(\frac{K}{N}\right) \right\rfloor}$ constitutes a first step of an optimal algorithm.

See Manolopoulos, Kollias, and Hatzopoulos (1986); Manolopoulos, Kollias, and Burton (1987) for other multi-object search algorithms.

5.5 Parallel (polychotomous) search

See Herer and Raz (2000) for another model involving parallel search.

Itai (1976) analyzes optimal alphabetic t-ary trees, i.e., t-ary trees with minimum weighted path length. This model is equivalent to search in which $t - 1$ queries are made simultaneously.

An s-forest is a sequence of s trees, and its cost is the sum of costs of these trees. Denote a minimal cost (optimal) alphabetic s-forest on weights $(w_i, \ldots, w_j)$ by $F_s[i, j]$: $WF_1[k, k] = w_k$ for $k \in \{i, \ldots, j\}$, and for $s > 1$ and any s' such that $1 \leq s' < s$,

$$WF_s[i, j] = \min_{i \leq b < j} \{WF_{s'}[i, b] + WF_{s-s'}[b + 1, j]\}.$$

We are interested in $WF_1[1,N]$. The cost of a t-ary tree is $f[i,j] = W_{ij} + WF_t[i,j]$, where $W_{ij} = \sum_{r=i}^{j} w_r$. For each $\delta = 1,\ldots,N-1$, the author finds optimal t-trees and s-forests for weights $(w_i,\ldots,w_j)$, $j = i+\delta$. By choosing s' in an economic way at each stage (for example $s'_i = 2^{\lfloor \log s'_{i-1} \rfloor} - 1$), the resulting complexity is $O(N^3 \log t)$.

Gotlieb (1981) and **Gotlieb and Wood (1981)** show that the monotonicity principle does not hold for t-ary trees, contradicting a claim in Itai (1976).

Abrahams (1994) considers optimal partitioning of the interval of uncertainty into k subintervals and searching them simultaneously. The goal is to minimize the expected search length needed to locate the target. This is done by activating the Hu-Tucker algorithm for $N-k$ merge steps and searching the k trees of the resulting forest simultaneously. The author also investigates the consequences of increasing k and this provides a basis for deciding whether such an increase is desired. However, no economic model (involving costs per searcher and search time) is offered for this problem.

Ben-Gal (2004) demonstrates that the weight-balanced testing heuristic (see Allen (1982)) for the multi-searcher search is in general far from being optimal.

5.5.1. Asymmetric inspection costs in parallel search Let the cost of an edge (a,b) of a t-ary tree be c_i, where b is the i-th child of a, $1 \le i \le t$. In general, the tree need not be full (a node may have a first and third child without the second). The cost of a tree is $\sum_{i=1}^{N} p_i w_i$ where w_i is the weight of leaf i and p_i is the sum of edge costs of the path from the root to i. The problem is to compute a minimum-cost alphabetic tree.

The unequal-inspection-costs problem in parallel search can be also reformulated in terms of coding theory, where the encoding letters have different costs.

Varn (1971) first considers average-cost minimizing *exhaustive* t-ary search (constructing a full t-ary tree) and proves that the optimal tree with N terminal nodes is obtained by replacing the minimum-cost leaf by an internal node with t terminal nodes as children. This property is then used to compute the optimal non-exhaustive solution in $O(tN^2 \log N)$ time. **Perl, Garey and Even (1975)** describe more efficient $O(tN \log N)$ and $O(t \cdot N)$ algorithms for this problem.

Itai (1976) solves the problem by dynamic programming. Let $F_{\alpha,\beta}[i,j]$ be the cost of an optimal tree for weights $(w_i,\ldots,w_j)$ in which the root has no child smaller than α or greater than β, $1 \le \alpha \le \beta \le t$, $1 \le i \le j \le N$.

$$F_{\alpha,\alpha}[i,i] = c_\alpha w_i \text{ for } \ 1 \le i \le N,$$

$$F_{\alpha,\beta}[i,i] = \min_{\alpha \le \gamma \le \beta} \{F_{\gamma,\gamma}[i,i]\} \text{ for } \ 1 \le i \le N,$$

For $i < j$ and $1 \le \alpha \le t$:

$$F_{\alpha,\alpha}[i,j] = c_\alpha W_{ij} + \min_{\substack{i \le b < j \\ 1 \le \gamma < t}} \{F_{\gamma,\gamma}[i,b] + F_{\gamma+1,t}[b+1,j]\},$$

and for $i < j$, $\alpha < \beta$:

$$F_{\alpha,\beta}[i,j] = \min\left\{F_{\alpha+1,\beta}[i,j], \min_{i \le b < j} \{F_{\alpha,\alpha}[i,b] + F_{\alpha+1,\beta}[b+1,j]\}, F_{\alpha,\alpha}[i,j]\right\}.$$

Altenkamp and Mehlhorn (1980) provide bounds for the minimum cost of the search, and a linear time approximation algorithm. **Choy and Wong (1983)** derive a linear time algorithm for the problem with costs $c_i = a + (i-1)b$, where $a, b > 0$. Choi and Golin (2001) characterize the combinatorial structure of the optimal tree emphasizing the asymptotic analysis when N grows.

Shivakumar and Venkatasubramanian (1996) describe an application of alphabetic trees to broadcasting in wireless systems. The *tuning time* of a user is the time spent before it starts downloading the desired information. Tuning time can be reduced by indexing frequently used keys and organizing it in a t-ary alphabetic tree (also called an *index tree*). Further details on the use of index trees in multiple wireless broadcast channels are described by **Lo and Chen (2000); Jung, Lee, and Pramanik (2005); Gao, Yang, Chen, Lu, and Zhong (2016)**. Shivakumar and Venkatasubramanian (1996) and Gao et al (2016) also offer heuristics based on the Hu-Tucker algorithm.

A generalization of the Hu-Tucker algorithm to optimal ternary trees is presented in Morgenthaler and Hu (2014).

5.6 Search for rationals

How many queries of the form "is $x \le \frac{p}{q}$" (for $p, q \in \mathbb{N}$) are needed to determine a rational number x with denominator and numerator that are integers bounded by an integer M? One can list and sort all possible rational numbers in an array and perform a binary search, but the complexity of the preprocessing phase is already $\Omega(M^2)$. Efficient $O(\log M)$-time algorithms that do not require a preprocessing phase are proposed by **Papadimitriou (1979)** and **Reiss (1979)**.

Zemel (1981) discusses applications that require searching for a rational number in a bounded set of rationals. Consider the problem (PR):

$$s^* = \max \frac{c_0 + cx}{d_0 + dx} \quad \text{subject to } x \in F,$$

where $c_0, d_0 \in \mathbb{Z}$, $c, d \in \mathbb{Z}^n$ and F is a set of 0/1 vectors in $\mathbb{R}^n$. Consider the linear version (PL): $s^* = \max\{cx \ : \ x \in F.\}$. Megiddo (1979) proves that if problem (PL) can be solved within $O(p(n))$ comparisons and $O(q(n))$ additions, then problem (PR) can be solved in time $O(p(n)[q(n)+p(n)])$. The author generalizes this result proving that (PR) is solved in polynomial time iff (PL) is, thus removing the limitation on the type of operations allowed by the algorithm for (PL).

A second application applies search for rationals for efficiently solving the weighted p-center problem on a tree.

Kwek and Mehlhorn (2003) present an algorithm that requires $2\log M + O(1)$ queries, which matches the lower bound for this problem. The algorithm expresses x as $\lfloor x \rfloor + \frac{a}{b}$ where a and b are relatively prime and $a < b$. Searching for the integer part combines exponential search with binary search:compare x with 2^k for $k = 0, 1, 2 \ldots$ until $x \le 2^k$ and then apply binary search to locate x in the interval $[2^{k-1}, 2^k]$. The number of comparisons required so far is $2\log\lfloor x \rfloor + O(1)$. The fraction $\frac{a}{b}$ is determined efficiently by computing in $2\log M - 2\log\lfloor x \rfloor + O(1)$ queries an interval of form $\left[\frac{\mu}{2T^2}, \frac{\mu+1}{2T^2}\right]$ for $T := \left\lfloor \frac{M}{\lfloor x \rfloor} \right\rfloor$. This fraction is unique in the computed interval.

6. Search for a state-transition point

This section refers to a two-state process. Initially, the state is *in-control* and at a certain time it changes to be *out-of-control* and stays there. In the basic model, *conforming* items are produced when the state is in-control, and defective *nonconforming* items are produced when it is out-of-control.

Suppose that an item produced by a machine is found to be defective (non-conforming). It is the N-th item produced since the machine was last inspected and found to be operating properly. All items produced after the first defective item are also flawed. If a query at an item reveals that it is not defective then the first defective item was produced later. Otherwise, the first defective item has already been produced.

The producer's objective is to minimize the expected number of inspections required to find the transition point.

In many applications the failure rate is assumed to be constant, i.e., the location of the first nonconforming object is truncated geometric. Formally, there is a 2-state binary stochastic process $\{I_j, j = 0, 1, \dots\}$, with $I_0 = 0$ and $I_N = 1$. Once in state 1 the system remains there. If in state 0 the system stays there to the next period with probability $p < 1$. The objective is to find t such that $I_0 = \cdots = I_{t-1} = 0$ and $I_t = I_{t+1} = \cdots = I_N = 1$, with minimum expected number of inspections. This time period will also be referred to as *first nonconforming unit* (FNU).

A closely related subject of research deals with optimal *on-line* inspection of systems which are *stochastically deteriorating*. The main goal there is to minimize the sum of inspection costs and cost associated with the time between the failure of the system and its discovery by an inspection. These problems typically refer to a cyclic environment. Once a failure is detected the equipment is repaired and there is often no further interest in backtracking the exact time the failure occurred, as in the *off-line* search models which are the subject of this survey. Moreover, the time to failure is often unbounded (corresponding to $N = \infty$ in our notation). There is an extensive literature on such systems and we do not cover it, but rather refer the reader to existing surveys McCall (1965); Sherif and Smith (1981); Chelbi and Ait-Kadi (2009); Sarkar and Saren (2016).

6.1 The basic problem

The first research on this problem is by **Hassin (1984)** who illustrates the problem through an example of a communication system consisting of $N - 1$ transmitting stations. A message is sent from the source to the first station, then to the second and so forth, until it is sent to the final destination. The number of messages a station transmits until it fails is geometrically distributed. Given that a message has not arrived at the destination, the goal is to locate the defective transmitter using minimal expected number of queries. A query at a transmitting station reveals whether the message has arrived to it.

The probability that we observe state 0 after j transitions is $\frac{p^j(1-p^{N-j})}{1-p^N}$. Let $f(n)$ denote the expected search cost under the optimal strategy in a problem of length n. Then $f(1) = 0$ and

$$f(n) = 1 + \min_{x=1,\dots,n-1} \left\{ \frac{p^x(1-p^{n-x})}{1-p^n} f(n-x) + \frac{1-p^x}{1-p^n} f(x) \right\}. \tag{6.1}$$

Let x_n^* denote the minimizer of $f(n)$. The author proves that $x_{n+1}^* \in \{x_n^*, x_n^* + 1\}$ for $n \in \{1, \dots, N\}$, and, s a corollary, (6.1) can be modified and solved in $O(N)$ time: Let $F(n) = (1 - p^n)f(n)$. Then

$$F(n) = (1 - p^n) + \min_{x=x_{n-1}^*, x_{n-1}^*+1} \{p^x F(n-x) + F(x)\}.$$

The problem can be reformulated via the alphabetic tree formulation: Find a binary tree minimizing $\sum_{k=1}^N l(k)p^{k-1}$ (the weighted path length), where $l(t)$ is the level of node t.

A tree satisfying $l(1) \le l(2) \le \cdots \le l(N)$ is *nondecreasing*. There is a one-to-one correspondence between nondecreasing sequences of integers satisfying $\sum_{k=1}^N (1/2)^{l(k)} = 1$ and nondecreasing alphabetic binary trees. It is shown that the optimal tree is nondecreasing and the levels $l(j)$ of its terminal nodes solve the following problem:

$$\min \left\{ \sum_{k=1}^N l(k)p^{k-1} \;:\; \sum_{k=1}^N \left(\frac{1}{2}\right)^{l(k)} = 1,\ l(1), \dots, l(N) \text{ are integers.} \right\} \tag{6.2}$$

The author suggests an approximate solution applying Lagrange approximation based on relaxation of (6.2), treating $l(k)$ as a continuous variable. The approximate value of $F(N)$ is given as a closed formula:

$$\widehat{F}(N) = (1 - p^N)\log_2\left(\frac{p - p^{N+1}}{1-p}\right) - \left(\frac{1-p^{N+1}}{1-p} - (N+1)p^N\right)\log_2 p.$$

If $x_N^* = \log_p\left(\frac{1+p^N}{2}\right)$ is between $k - \frac{1}{2}$ and $k + \frac{1}{2}$ for an integer k, the approximate strategy is k. The difference between the approximate and optimal values are numerically compared and found not to exceed 0.5%.

The author also mentions a version of the model where I_N is not a priori assumed to be 1. A simple approximation for this version checks the last state - if it is 0, then $I_j = 0$ for all $j = 0, \ldots, N$. Otherwise the problem is reduced to (6.1). The complexity of the search is $O(N)$, and the expected number of queries differs from the optimal solution by at most one query.

A simple modification of the dynamic program solves this version: Let $g(n)$ be the expected number of inspections needed if I_n is unknown, and let $f(n)$ be the expected number of inspections when $I_n = 1$. Note that $g(1) = 1$ and $f(1) = 0$. Then, $f(n)$ is given by (6.1), and

$$g(n) = \min_{1\le k\le n} \left\{1 + p^k g(n-k) + (1-p^k) f(k)\right\}.$$

He, Gerchak and Grosfeld-Nir (1996) further investigate this problem. They observe a crucial difference when I_N is a priori known to be 1 and the version they treat when it is unknown: In their case the unit one should inspect first is not monotone in N. Therefore the complexity reduction to $O(N)$ is not possible. Another qualitatively different result is that if $p^N + p^{N-1} > 1$ then the optimal first query is at N, whereas it is proved in Hassin (1984) that when $I_N = 1$ is a priori known, the optimal first query is always placed before $N/2$.

The optimal first inspection has a limiting value of $\log_p(0.5)$ when $N \to \infty$. Moreover, it converges to the same limiting value if it is known that $I_N = 1$. This suggests the following heuristic:

(1) If $N \ge \log_p(0.5)$, inspect unit $\lfloor \log_p(0.5) \rfloor$.

(2) If $N < \log_p(0.5)$ and I_N is unknown, inspect unit N.

(3) If $I_N = 1$, inspect unit $N/2$.

Herer and Raz (2000) consider general failure rates. Let p_i be the probability that i is the FNU, and denote $P := (p_1, \ldots, p_N)$. The uncertainty associated with the FNU's location is measured by the entropy $U_0(N, P) = -\sum_{i=1}^N p_i \log p_i$. The uncertainty after inspecting unit k in the batch becomes

$$(1 - a(k,N,P))U_0\left(N-k, \frac{(p_{k+1}, \ldots, p_N)}{\sum_{i=k+1}^N p_i}\right) + a(k,N,P)U_0\left(k, \frac{(p_1, \ldots, p_k)}{\sum_{i=1}^k p_i}\right),$$

where $a(k, N, P)$ denotes the probability of shifting to state 1 no later than unit k when it is known that unit N is non-conforming and the probability vector is P.

If only one inspection is available then the remaining uncertainty is minimized by approximately dividing the batch into two segments with equal probability of containing the FNU, i.e., at

$$\bar{k} = \arg\min_{k\in 1,\ldots,N} |0.5 - a(k,N,P)|. \tag{6.3}$$

For the geometric case with parameter p, $p_i = \frac{p^{i-1}(1-p)}{1-p^N}$. Thus $\bar{k} = \left\lfloor \max\left(1, \log_p\left(\frac{1+p^N}{2}\right)\right) + 0.5 \right\rfloor$, exactly as obtained in Hassin (1984), where it was reached by a different method of continuous relaxation of an integer program. Thus, equation (6.3) generalizes this heuristic for an arbitrary distribution of the FNU.

Numerical comparisons confirm that for the geometric case this lower bound is very tight. Moreover, the proposed heuristic yields better results than the heuristic proposed in He, Gerchak and Grosfeld-Nir (1996).

The authors extend the analysis allowing of t simultaneous inspections: Let $K = (k_1, \ldots, k_t)$ denote the units inspected during that round, $k_0 = 0$, and $k_{t+1} = N$. The uncertainty after inspecting K is

$$\sum_{i=1}^{t+1} (a(k_i, N, P) - a(k_{i-1}, N, P)) \cdot U_0 \left(k_i - k_{i-1}, \frac{p_{k_{i-1}+1}, \ldots, p_{k_i}}{\sum_{i=k_{i-1}+1}^{k_i} p_i} \right).$$

The main results state that:

(1) The remaining uncertainty is minimized by the inspection vector that approximately forms $t+1$ equal-probability segments:
$K = \arg\min \sum_{i=1}^{t+1} \left| a(k_i, N, P) - a(k_{i-1}, N, P) - \frac{1}{t+1} \right|$.

(2) The maximum reduction in uncertainty is $\log(t+1)$. A lower bound on the expected number of inspection rounds can be computed by dividing the initial uncertainty by $\log(t+1)$.

6.2 Economic models

Raz, Herer and Grosfeld-Nir (2000) consider the geometric case where C_I is the inspection cost per unit, C_P is the penalty of incorrect acceptance, and C_S the penalty of incorrect rejection. They first find the optimal solution if no inspections are performed at all:

(1) If the quality of the last unit is unknown, accept the first

$$j^* = \left\lfloor \frac{\log(C_P/(C_S + C_P))}{\log p} \right\rfloor$$

units, and reject the rest. Note that j^* is independent of batch size, N. $j^* = 0$ means that all units should be rejected. (When $j^* > N$ we set $j^* = N$.) The expected cost is

$$V^0(N) = C_p \left[j^* - p \frac{1 - p^{j^*}}{1 - p} \right] + C_S \frac{p^{j^*+1} - p^{N+1}}{1 - p}.$$

(2) If the last unit is known to be non-conforming, accept the first

$$j' = \left\lfloor \frac{\log((C_P + p^N C_S)/(C_S + C_P t))}{\log p} \right\rfloor$$

units, and reject the rest. As expected, $j' < j^*$. The expected cost is[10]

$$G^0(N) = C_p \frac{j' - p\left(\frac{1-p^{j'}}{1-p}\right)}{1 - p^N} + C_S \left[\frac{p^{j'+1} - p^{N+1}}{(1-p)(1-p^N)} - \frac{(N - j')p^N}{(1 - p^N)} \right].$$

The following equations compute the optimal inspection/disposition policy:

$$V(k) = \min \left\{ \min_{1 \le j \le k} \left\{ C_I + \Pr[I_j = 1] G(j) + \Pr[I_j = 0] V(k - j) \right\}, V^0(k) \right\}$$

$$G(k) = \min \left\{ \min_{1 \le j \le k-1} \left\{ C_I + \Pr[I_j = 1 | I_k = 1] G(j) + \Pr[I_j = 0 | x_k = 1] G(k - j) \right\}, G^0(k) \right\}.$$

where:

- $I_j = 0$ if j conforms, $I_j = 1$ otherwise.
- $V(k)$ is the cost of the optimal policy for a batch of size k and the quality of the last unit is unknown.
- $G(k)$ is as $V(k)$ but TeX last unit is non-conforming.

[10] **Wang and Chuang (2011)** use these findings to compute the optimal testing policy when inspections are made at equal intervals and no further inspection is allowed once a nonconforming item is found.

Boundary conditions are $G(1) = 1$ and $V(0) = 0$, and the complexity is $O(N^2)$.

Chun (2010) assumes that the profit from an item verified as conforming is v_a whereas the (possibly negative) salvage value of any other item is $v_b < v_a$. The problem is (1) to compute the optimal size, N, of the inspected batch given the inspection cost c and the constant failure rate p, and (2) once a nonconforming item is found, how to conduct an inspection in the last batch. The resolution of the second question is similar to that in Raz, Herer and Grosfeld-Nir (2000) (where the problem is posed in terms of costs rather than profit).

The number of batches k taken until a defective item is detected is geometric with parameter p^N. Let c be the cost of one inspection. The expected profit per a produced item π is shown to be

$$\pi(N) = p^N v_a + (1 - p^N)\frac{EV^*(N)}{N} - \frac{c}{N},$$

where $EV^*(N)$ is the maximum expected profit obtained for a sequence of N items, when the N-th item is non-conforming. The optimal inspection interval minimizes $\pi(N)$. The author also suggests a methodology for estimating p.

6.3 Process recovery

Finkelshtein, Herer, Raz, and Ben-Gal (2005) allow the production process to recover after failure. The transition probability is p_c, and the reverse transition probability is p_n.

The inspection cost is C_I, the cost of incorrect acceptance is C_P, and the cost of incorrect rejection is C_S. Let S_b and S_e be the state of the system before the start of the batch and at the end of the batch respectively. Each of these variables can be conforming (c) or non-conforming (n).

Let $P_i^{S_b}$ be the probability that unit i is conforming given that the initial condition of the batch was S_b, then $P_i^c = \left[(1 - p_n - p_c)^i p_c + p_n\right]/(p_n + p_c)$, and $P_i^n = 1 - \left[(1 - p_n - p_c)^i p_n + p_c\right]/(p_n + p_c)$.

Consider a batch of size N. Let $a_i^{S_bS_e}(N)$ be the probability that unit i is conforming given the initial state S_b and final state S_e. For example $a_i^{cc} = \frac{P_i^c P_{N-i}^c}{P_N^c}$. Let $W^{S_bS_e}(N)$ be the minimal expected cost of classifying the units in the batch without inspection, given states S_b and S_e. Then:

$$W^{S_bS_e}(N) = \sum_{i=1}^{N} \min\left(a_i^{S_bS_e}(N)C_S, \left[1 - a_i^{S_bS_e}(N)\right] C_P\right).$$

The minimal cost of classifying the units in a batch, given S_b and S_e and that unit j is to be inspected is

$$\begin{aligned} G_j^{S_bS_e}(N) &= C_I + a_j^{S_bS_e}(N)\left(G^{S_bc}(j) + G^{cS_e}(N-j)\right) \\ &+ (1 - a_j^{S_bS_e}(N))\left(G^{S_bn}(j) + G^{nS_e}(N-j)\right), \end{aligned}$$

and the expected cost when inspecting optimally is

$$G^{S_bS_e}(N) = \min\left[W^{S_bS_e}(N), \min_{1\le j\le N} G_j^{S_bS_e}(N)\right].$$

The complexity of this recursion is $O(N^2)$.

W. Wang, Sheu, Chen and Horng (2009) add an option to repair nonconforming items. It is assumed that a constant proportion δ of defective units can be successfully repaired. Thus, if unit j is found to be non-conforming, the values obtained from units $j+1$ through N is $(N-j)\delta(U - C_r)$ where C_r is the repair cost and U is the sale price.[11]

[11] Equivalently, all units are repairable, but the sale price for a repaired unit is $\delta(U - C_r) < U$.

Let X denote the unknown FNU.[12] The solution is given by the following recursive formulas:

$$\begin{aligned}
ER_V(N) &= \max\left\{\max_{1\le j\le N}\left[ER_V^1(N,j)\right], ER_V^0(N)\right\},\\
ER_G(N) &= \max\left\{\max_{1\le j\le N-1}\left[ER_G^1(N,j)\right], ER_G^0(N)\right\},\\
ER_G^1(N,j) &= \Pr(X\le j\,|X\le N\,)\left\{ER_G(j)+(N-j)\delta(U-C_r)\right\}\\
&+ \Pr(X> j\,|X\le N\,)\left\{ER_G(N-j)+U\cdot j\right\}-C_I,\\
ER_V^1(N,j) &= \Pr(X\le j)\left\{ER_G(j)+(N-j)\delta(U-C_r)\right\}\\
&+ \Pr(X> j)\left\{ER_V(N-j)+U\cdot j\right\}-C_I.
\end{aligned}$$

$ER_V(j)$ is the expected profit the optimal inspection policy when the batch size is j, and $ER_G(j)$ is the same when but when unit j is known to be non-conforming.
$ER_V^1(N,j)$ is the expected profit obtained by the optimal policy given that unit j will be inspected first, and $ER_G^1(N,j)$ is the same but when the last unit is non-conforming.
$ER_V^0(N)$ is the expected profit from the optimal no-inspection policy when the batch size is N, and $ER_G^0(N)$ is the same but when the last unit is known to be non-conforming. These functions are computed as in Raz, Herer and Grosfeld-Nir (2000).

Tsai and Wang (2011) complete the results of W. Wang, Sheu, Chen and Horng (2009) to general distributions of the FNU. Moreover, when an inspection policy is explored, not only reworking the identified nonconforming units but also their rejection is considered.[13]

6.4 Unreliable answers

In real life, conforming units can be mistakenly classified as non-conforming and vice versa. Let α denote the probability of misclassifying a conforming unit; β is the probability misclassifying a nonconforming unit. A common assumption to the articles described below is that each item can be tested only once during the search. There are costs C_I per inspection, C_p per incorrect acceptance of a unit, and C_s per incorrect rejection.

Sheu, Chen, Wang and Shin (2003) define x_j to be 1 or 0 if unit j is judged to be conforming or non-conforming, respectively. The analysis is conducted as in Raz, Herer and Grosfeld-Nir (2000), only with $\Pr[x_j = 1] = p^j(1-\alpha) + (1-p^j)\beta$. The authors derive recursion equations.

Wang (2007) and **Chun (2008)** identify several flaws in Sheu, Chen, Wang and Shin (2003). The main one being that the formulas mix between the observed inspection result and the unobservable state and that they do not apply when the failure rate is not constant. Chun provides a reformulation of the equations. In a reply Sheu et.al. Sheu et.al. (2008) correct one of Chun's formulas.

Wang (2007) also suggest a heuristic for the problem of Sheu, Chen, Wang and Shin (2003) assuming that the FNU location is geometric. Consider a batch of size k consisting of unites $f, f+1, \ldots, f+k-1$. If an inspection of unit $f+j-1$ classifies it as non-conforming then it is assumed that units $f+j$ through $f+k-1$ are rejected. If it is classified as conforming then units f to $f+j-2$ are accepted. The search continues on the rest of the batch ignoring the results of past inspection taken outside the batch. In general, the current batch to be tested is such that its first (last) item has been inspected and found conforming (nonconforming).

[12] As noted in Tsai and Wang (2011), these equations only apply when the FNU distribution is geometric. For a general distribution one must add an extra index denoting the beginning of the interval.

[13] The boundary conditions given in the solution procedure of W. Wang, Sheu, Chen and Horng (2009) are also corrected; the expected payoff of a batch with one nonconforming unit can be positive since rework is possible.

The author first computes the optimal non-inspection break-even point, i.e., a point such that the expected cost of rejecting all units after it and accepting all other units is minimal. The computation is completed by defining appropriate $O(N^3)$ recursive formulas.

Tzimerman and Herer (2009) consider inspection errors with the objective of finding the transition point with a given confidence level γ using minimal number of inspections. The location of the transition point is arbitrarily distributed. X_j is 1 if unit j is conforming and -1 otherwise; I_j is 1 if the inspection result indicates that unit j is conforming and -1 otherwise.

The authors use the following notations: T_i is the information gathered before iteration i; T_{i+1}^{j+} (T_{i+1}^{j-}) is T_{i+1} when inspecting j at iteration i classifies j as conforming (non-conforming); $f(T_i, j)$ is the expected number of additional inspections if unit j is inspected next; j^* is the optimal unit to be inspected; $f^*(T_i)$ is the minimum expected number of additional inspections under the optimal inspection policy. Thus, $f^*(T_j) \equiv f(T_i, j^*)$ where:

$$f(T_i, j) = 1 + \Pr[I_j = 1 \,|T_i] \cdot f^*(T_{i+1}^{j+}) + \Pr[I_j = -1 \,|T_i] \cdot f^*(T_{i+1}^{j-}).$$

As long as we have not yet identified the transition point with required confidence, we continue the search by inspecting j^*. The complexity of the dynamic program is $O(N3^N)$.

Four heuristics are introduced and compared. The one yielding the best outcome is a weight-balanced heuristic closest to each having a 50% probability of containing the transition point (see Allen (1982)).

6.5 Unreliable processes

This section refers to situations where conforming items may also be produced when the system is out-of-control and nonconforming items may also be produced when the system is in-control. In addition to the inspection cost C_I there are penalties C_P and C_S for incorrect acceptance and rejection, respectively.

Wang and Hung (2008) assume that the FNU is strictly greater than j with probability $p^{j\alpha}$. The geometric case is obtained when $\alpha = 1$. If $0 < \alpha < 1$ then we have a decreasing failure rate, and if $\alpha > 1$ then it is a case of increasing failure rate. The authors make the simplifying restrictive assumption that when a nonconforming unit is found all subsequently produced units are rejected, and when a conforming unit is found all earlier units are accepted. The optimal policy is computed assuming penalties on incorrect acceptance and rejection.

Bendavid and Herer (2009) assume that the probability for a unit produced during the abnormal state to be non-conforming is $\alpha_O > \alpha_I$, where α_I is the probability of producing a nonconforming item during the normal state of production. The distribution of the transition point location is arbitrary. Let $S = (s_1, \ldots, s_N)$ where $s_i = u$ if unit i has not been inspected, $s_i = n$ if unit i has been inspected and found non-conforming, and $s_i = c$ if it has been inspected and found conforming.

The authors define $P_C^n(S) = (p_1^n(S), \ldots, p_N^n(S))$ and $P_C^c(S) = (p_1^c(S), \ldots, p_N^c(S))$ as the vectors of probabilities that the units are non-conforming and conforming respectively given the vector S. $f(S)$ is the minimum search cost for a given vector S, and $S \,|s_k \leftarrow c$ represents the vector S with its k-th element (presently u) replaced by c.

The dynamic program is

$$f(S) = \min \left[\min_{j|s_j=u} \left\{ C_I + p_j^c(S) f(S \,|s_j \leftarrow c) + p_j^n(S) f(S \,|s_j \leftarrow n) \right\}, W(S) \right], \tag{6.4}$$

where $W(S)$ is the cost of the optimal no-inspection policy. The algorithm's complexity is $O(N3^N)$, and therefore the authors develop heuristics for selecting the next inspection unit i:

(1) Greedy in uncertainty: The probability that a transition occurs at or before i is closest to 0.5, i.e., $i = \arg\min_{i|s_i=u} \left| 0.5 - \sum_{j=1}^{i} p_j^T(S) \right|$.

(2) Greedy in cost: i minimizes the expected no-inspection cost obtained after performing one inspection:
$$i = \arg \min_{i|s_i=u} \left\{ p_i^c(S)W(S, s_i \leftarrow c) + p_i^n(S)W(S, s_i \leftarrow n) \right\}.$$

(3) Myopic stopping rule: Inspect i if this reduces the cost assuming that no more inspections will be allowed. We inspect i if $W(S) > h^1(S) \equiv C_I p_i^c(S)W(S, s_i \leftarrow c) + p_i^n(S)W(S, s_i \leftarrow n)$.

(4) For the "look ahead" stopping rule the authors define
$$h^j(S) = C_I + p_i^c(S) \min \left\{ W(S, s_i \leftarrow c) \cdot h^{j-1}(S, s_i \leftarrow c) \right\} +$$
$$p_i^n(S) \min \left\{ W(S, s_i \leftarrow n) \cdot h^{j-1}(S, s_i \leftarrow n) \right\}.$$
$h^i(S)$ can be interpreted as the expected cost obtained after performing up to j inspections and then implementing the optimal no-inspection policy. The inspection is performed iff $W(S) > h^{\lfloor \log N \rfloor + 1}(S)$.

The combinations of a selection rule (1 or 2) with a stopping rule (3 or 4) create four heuristic policies, and the dominance of the heuristic composed of the greedy in cost selection rule and the look-ahead stopping rule over all others is demonstrated.

C-H. Wang, Shih and Tsai (2011) suggest a heuristic approach based on identifying the transition point with a given confidence level (similar to Tzimerman and Herer (2009)). The selection of item to inspect next is the one that minimizes the uncertainty of the transition point as expressed by the expected entropy. The search terminates when either all items are inspected, or the time of transition is identified with the given confidence level. In the latter case all earlier uninspected items are accepted and later uninspected items are rejected. Inspected items are accepted or rejected according to the inspection results.

Numerical examples indicate that for a batch of size < 35, the expected number of inspections to meet a confidence level of 0.95 doesn't exceed 3.5.

Chen, Pan, and Cui (2017) solve a variation of the problem with an exogenous confidence level for classifying in-control units. In their numerical analysis the authors allow variable inspection costs, such that the cost of h-th inspection is proportional to h^γ for a constant γ.

7. Dichotomous search experimentation and games

7.1 Search for unknown level of demand

Alpern and Snower (1987) assume that a product's demand D is uniformly selected from a given interval and stays constant over time. The firm searches for the unknown D as follows: At the beginning of period k the firm produces Q_k units. The inventory carried from the previous period is $(1-\delta)I_{k-1}$, where I_{k-1} is the inventory stock held at the end of that period and $\delta < 1$ is the inventory depreciation rate. At period k the firm puts up for sale a quantity $S_k = (1-\delta)I_{k-1} + Q_k$, and sells $\min(S_k, D)$ units. If $I_k = 0$, then the firm learns that $D \in [S_k, \bar{D}]$. If $I_k > 0$ then $D = S_k - I_k$ and the firm future supplies will all be equal to D.[14] If demand exceeds supply, the difference is lost.

Let p be the selling price, f the unit cost of production, h the unit cost of holding inventory per period, and α the discount factor. Given a *supply strategy* $S_1 \leq S_2 \leq \cdots$, the firm's opportunity cost of not correctly guessing D is

$$(p-f) \left[\sum_{t=1}^{N-1} \alpha^{t-1}(p-f)(D-S_t) + \alpha^{N-1} b(S_N - D) \right], \tag{7.1}$$

[14] Unless $I_k > D$. The authors simplify the presentation by ignoring this possibility.

where N is the least t with $S_t > D$ and $b = \frac{h+f\cdot(1-\alpha(1-\delta))}{p-f}$.

The firm's objective is to minimize expected opportunity cost. The main result of the paper states that if $D \sim$ U[0,1], the optimal quantity to be put up for sale in period k (provided that the previous supplies have resulted in stock-outs) is $S_k = 1-(1-\lambda)^k$, where

$$\lambda = \frac{\alpha b - b - 1 + \sqrt{(b-\alpha b+1)^2 + 4\alpha b}}{2\alpha b}.$$

Alpern and Snower (1988) solve a two-period version of Alpern and Snower (1987) with $D \sim$ U[0,1], $p-f=1$ and $h=1$. The optimal solution in this case is $S_1 = (\delta+2)/(\delta+4)$, $S_2 = (\delta+3)/(\delta+4)$.

Alpern and Snower (1988a) suggest a generalization of Alpern and Snower (1987) where the price is a decision variable and can be changed over time. The firm's goal is to find the profit-maximizing price and the associated demand. A two-dimensional search can be carried out by iteratively setting price p and offering for sale d units. If the offered amount d is fully sold the firm concludes that $f(p) \geq d$. Otherwise, the amount sold reveals $f(p)$.

Reyniers (1988) considers a variation of Alpern and Snower (1987) with *information delays*: the sale outcome at period k is not known to the supplier until period $k+2$. The firm's goal is to minimize the maximum possible cost over all possible demand values. As in Alpern and Snower (1987), it is assumed that the maximum stock level can be consumed in one period so inventory is held only one period. Reyniers (1990) solves a *newsvendor variation* where it is not possible to hold inventory.

Reyniers (1989a) assumes stockouts decrease future demand. The unique interesting feature of this model is that the location of the target is influenced by the search policy. The author solves this model both under the min-max objective and under the minimum expected cost objective when the initial demand is a uniform random variable.

Reyniers (1989) assumes that the demand in a sequence of newsvendor problems linearly increases: $D_t = D + \alpha t$, where D is known but not the slope $\alpha \in [\alpha_L, \alpha_U]$. Guessing D_t is equivalent to guessing α, but with a different opportunity-cost structure than previous models. The author obtains a closed-form solution to the min-max problem.

Alpern (1989) surveys previous literature on learning from experimentation for unknown level of demand and also offers some extensions.

7.2 Wage bargaining - optimal wage request

This section deals with dynamic models of wage bargaining. It is assumed that the worker's value to the firm is known to the firm but not to the worker. The models cover variations from both worker and firm being nonstrategic (myopic) to both rationally playing a game. The strategy for the worker consists of a first-period wage demand w_1, and second-period demands w_a and w_r depending on whether his first-period demand is accepted or rejected. Both worker and firm are interested in maximizing discounted payoff, with discount factors δ_w and δ_f, respectively.

Alpern and Snower (1988) consider a worker whose value to a firm is a random variable $Q \sim$ U[0,1]. If the worker's wage demand is lower than the worker's value to the firm, the worker is hired and this wage is used to update the interval of uncertainty for the worker's value. Being unemployed is associated with utility -1 per period. The authors solve the two-period case, showing that the worker's first-period wage demand is $w_1 = \delta_w/(2(1+\delta_w))$. If hired, the worker demands the same wage in the second period, i.e., $w_a = w_1$, and if not then he demands zero wage just to avoid the disutility of being unemployed.

Reyniers (1992) considers the two-period model of Alpern and Snower (1988) assuming that unemployment is associated with zero utility. It is shown that $w_1 = \frac{1+\delta}{2+1.5\delta}$, $w_a = \max(w_1, 0.5)$ and $w_r = 0.5w_1$. The worker's optimal expected payoff is $P_w = (1+\delta_w)^2(1+0.75\delta_w)/(2+1.5\delta_w)^2$, and the expected payoff to the firm is $P_f = \delta_f w_1^2/8 + (1+\delta_f)(1-w_1)^2/2$.

The author then considers a variation where the firm is strategic, and influences the worker's wage demands through its hiring decisions, while the worker is naive and believes that the firm is myopic. In this case, it is optimal for the firm to hire the worker at his wage demand w_1 in period 1 if the worker's value exceeds $w_1(1+0.5\delta_f)$. The firm's expected payoff is

$$P_f = \frac{\delta_f}{8} w_1^2 (1+\delta_f)^2 + \frac{1+\delta_f}{2}\left(1 - w_1\left(1+\frac{\delta_f}{2}\right)\right)\left(1 - w_1\left(1-\frac{\delta_f}{2}\right)\right),$$

and the worker's expected payoff is

$$P_w = (1+\delta_w) w_1 \left(1 - w_1\left(1+\frac{\delta_f}{2}\right)\right) + \frac{\delta_w}{4} w_1^2 (1+\delta_f).$$

When the firm is strategic, there is more first-period unemployment, the worker's utility is lower and the firm's payoff is higher relative to when the firm is myopic. These effects are higher when the respective discount factors are lower.

Reyniers (1998) considers a variation of Reyniers (1992) where workers observe the firm's first-period cutoff value. Based on the worker's wage demand and hiring history, the worker believes in period n that his value to the firm is uniformly distributed over an interval $[a, b]$. The firm then strategically sets a cutoff value for hiring the worker in the next period.

The author analyzes the two-period game and finds that there is a unique subgame perfect equilibrium, and some workers are hired at a wage above their value. A comparison with Reyniers (1992) where workers do not know the firm's cutoff reveals that workers are better off, but if $\delta_f > 0.8$ the firm's profit is smaller than when it behaves non-strategically.

Reyniers (2000) solves the two-period equilibrium when both worker and firm act strategically. The firm's strategy consists of a function $f(w_1)$ such that the worker's demand w_1 in period 1 is accepted iff his value to the firm exceeds $f(w_1)$. An interesting qualitative result of the equilibrium is that a worker hired in period 1 is always also hired in period 2.

The author also finds the equilibrium in an alternative model where the worker demands wage w_{12} for both periods. If rejected the worker makes another demand w_2 for his second-period wage. It is shown that both worker and firm prefer this mechanism iff $\delta_W \geq \delta_F$.

7.3 Dichotomous search games

In the generic dichotomous search game G_N a *hider* H hides an object at $y \in \{1, \ldots, N\}$ and a *searcher* S makes an effort to find its location. After each query x_i the searcher learns whether $y \leq x_i$ or $y > x_i$. A more difficult version of the game where the searcher also knows whether $y = x_i$ is explored by Gilbert (1962), Johnson (1964), and Fokkink and Stassen (2011). The payoff to the hider is the expected number of queries required to locate y. The *value* $v(N)$ of the game G_N is the maximal payoff that can be assured by the hider (i.e., the minimal price the searcher must pay). This game and variations of it are often referred to as *high-low search games* or *search games with directional information.*

Gal (1974) considers G_N. Let Q be a mixed strategy of S, and let q be a mixed strategy of H. $V_N(q, Q)$ is the expected number of queries used to locate y. Define $I = \lceil \log_2 N \rceil$, $J = \lceil N/2 \rceil$, $t_N = I + \frac{2N - 2^{I+1}}{N}$, and let the value of G_N be $v(N) = \sup_q \inf_Q V_N(q, Q) = \inf_Q \sup_q V_N(q, Q)$.

The author derives the optimal strategies of the hider. It is shown that for $N = 2J$, $v(N) = t_N$ and the optimal strategy of H is one of the following: (i) Choose each integer with probability $1/N$, (ii) choose each odd integer with equal probability, or (iii) choose each even integer with equal probability. For $N = 2J + 1$, $V(N) = I + \frac{2N - 2^{I+1}}{N-1}$ and the optimal strategy of H is to choose each even integer with equal probability.

An optimal strategy Q_N of S is also constructed.

Gal (1978) assumes the hider H chooses a point $e \in [a, b\ )$. In order to locate H, S can pose n sequential queries of the form "Is $x \geq t$?" If $x \leq e$ then S obtains a correct answer with probability α, and if $x > e$ then the probability of a correct answer is β, where $\alpha + \beta > 1$.

After making n observations, S chooses a set E and receives $\frac{1}{\mu(E)}$, where $\mu(E)$ is 0 if $e \notin E$ and the size of E otherwise. Thus S wishes to find a set which is small and contains e with high probability. H, on the other hand, wishes to find a distribution function of the location of e in a way that minimizes $1/\mu(E)$.

The author proves that: (1) uniformly choosing e according on $[a, b)$ is the unique optimal strategy of H; (2) Let $[a', b')$ be the interval of uncertainty at any stage of the search under the assumption that the information obtained so far is reliable. It is optimal for S to place the next query at $a' + \frac{\beta}{\alpha+\beta}(b' - a')$; the value of the search game is $\frac{(\alpha+\beta)^n}{b-a}$. Interestingly, both optimal strategies are independent of n.

Baston and Bostock (1985) consider a two-person zero-sum game with a hider choosing a point in [0,1] and a searcher wishing to locate this point by guesses $g_1, g_2, \ldots$, each time obtaining the information of whether the previous guess was high or low. The cost function is the *sum of errors* $\sum_{i=1}^{\infty} |g_i - H|$. The searcher's goal is to minimize the maximum cost of the search, while the hider's goal is to maximize the minimum payoff.

The results of Baston and Bostock (1985) are completed by **Alpern (1985)** (see also Alpern and Gal (1985) §5.2.1) where it is proved that the game has a finite value, approximately 0.624. The author also computes the optimal (pure) min-max strategy.

Ferguson (1996) considers a single-step game. A hider chooses $y \in [-1, 1]$ and a searcher chooses $x \in [-1, 1]$. Using the information of whether $x < y$ or $x > y$, the searcher estimates the value of y by z. The payoff given by the searcher to the hider is $(y - z)^2$.

This game has a value $v = \frac{1}{2e}$. The unique optimal strategy for the hider is to choose y according to a distribution $F(y)$ that has positive density over $(-1, +1)$ and probability $1/(e + 1)$ at $y = 1$ and at $y = -1$. The unique optimal search strategy choose x in a proper subinterval of $[-1, +1]$ with positive probabilities at its ends. Open problems related to this model are described by **Fokkink, Geupel, and Kikuta (2013)**.

References

Noomi Abigadol and Aharon Ben-Tal. A minmax search for the critical level of a system: The asymmetric case. *Naval Research Logistics*, 32:137–154, 1985. (Cited on page 8 and 17.)

Julia Abrahams. Parallelized Huffman and Hu-Tucker searching. *IEEE Transactions on Information Theory* 40:508–510, 1994. (Cited on page 19.)

Alekh Agarwal, Dean P. Foster, Daniel Hsu, Sham M, Kakade, and Alexander Rakhlin. Stochastic convex optimization with bandit feedback. *SIAM J. on Optimization* 23:213–240. (Cited on page 16.)

Brian Allen. On the costs of optimal and near-optimal binary search trees. *Acta Informatics* 18:255-263, 1982. (Cited on page 4, 19, and 26.)

Steve Alpern. Search for a point in interval, with high-low feedback. *Mathematical Proceedings of the Cambridge Philosophical Society*, 98:569–578, 1985. (Cited on page 30.)

Steve Alpern. Geometric search theory and demand uncertainty. *Engineering Costs and Production Economics* 17:245–251, 1989. (Cited on page 28.)

Steve Alpern and Shmuel Gal. *The theory of search games and rendezvous.* Kluwer Academic Publishers, Boston 2003. (Cited on page 30.)

Steve Alpern and Dennis J. Snower. Inventories as an information-gathering device. *World Bank Development Research, Report no. DRD267*, 1987. (Cited on page 27 and 28.)

Steve Alpern and Dennis J. Snower. "High-low search" in product and labor markets. *American Economic Review* 78:356–362, 1988. (Cited on page 28.)

Steve Alpern and Dennis J. Snower. A search model of optimal pricing and production. *Engineering Costs and Production Economics*, 15:279–284, 1988. (Cited on page 28.)

Doris Altenkamp and Kurt Mehlhorn. Codes: unequal probabilities, unequal letter costs. *J. of the Association for Computing Machinery* 27:412–427, 1980. (Cited on page 19.)

Andris Ambainis, Stephen A. Bloch, and David L. Schweizer. Delayed binary search, or playing twenty questions with a procrastinator. *Algorithmica* 32:641–650, 2002. (Cited on page 16.)

Arne Andersson. A note on searching in a binary search tree. *Software: Practice and Experience 21*, 10:1125–1128, 1991. (Cited on page 3.)

Shoshana Anily and Refael Hassin. Ranking the best binary trees. *SIAM J. on Computing*, 18:882–892, 1989. (Cited on page 6.)

Javed A. Aslam and Aditi Dhagat. Searching in the presence of linearly bounded errors. *STOC*, 1991. (Cited on page 16.)

Mordecai Avriel and Douglass Wilde. Optimality proof for the symmetric Fibonacci search technique. *The Fibonacci Quarterly*, 4:265–269, 1966. (Cited on page 7.)

Michael B. Baer. On conditional branches in optimal decision trees. *ISIT2007* 436–440, (2007). (Cited on page 10.)

Michael B. Baer. Alphabetic coding with exponential costs. *Information Processing Letters*, 110:139–142, 2010. (Cited on page 7.)

Victor J. Baston and F.A. Bostock. A high-low search game on the unit interval. *Mathematical Proceedings of the Cambridge Philosophical Society*, 97:345–348, 1985. (Cited on page 30.)

Paul Bayer. Improved bounds on the cost of optimal and balanced binary search trees. M.Sc. Thesis, MIT, Cambridge, 1975. (Cited on page 4.)

Ahmed A. Belal, Mohamed S. Selim, and Shymaa M. Arafat. Building optimal alphabetic trees recursively. *Proceedings of the Third WSEAS International Multi-conference on Mathematics, Wolin Island, Poland*, 2002. (Cited on page 4.)

Ahmed A. Belal, Mohamed S. Selim, and Shymaa M. Arafat. Towards a dynamic optimal alphabetic tree. *International J. of Cooperative Information Systems*, 4:46–74, 2004. (Cited on page 4.)

Irad Ben-Gal. An upper bound on the weight-balanced testing procedure with multiple testers. *IIE Transactions* 36:481–493, 2004. (Cited on page 19.)

Illana Bendavid and Yale Herer. Economic optimization of off-line inspection in a process that also produces non-conforming units when in control and conforming units when out of control. *European J. of Operational Research*, 195:139–155, 2009. (Cited on page 26.)

Michael Ben Or and Avinatan Hassidim. The Bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). *FOCS*, 2008. (Cited on page 15.)

Donald A. Berry and Roy F. Mensch. Discrete search with directional information. *Operations Research*, 34:470–477, 1986. (Cited on page 14.)

Biagio Bonasera, Emilio Ferrara, Giacomo Fiumara, Francesco Pagano, and Alessandro Provetti. Adaptive search over sorted sets. *J. of Discrete Algorithms* 30:-128-133, 2015. (Cited on page 7.)

Ryan S. Bergstrom and S. Rao Kosaraju. Comparison-based Search in the Presence of Errors. *STOC*, 1993. (Cited on page 16.)

Prosenjit Bose and Karim Douïeb. Efficient construction of near-optimal binary and multiway search trees. *WADS*, 2009. (Cited on page 3.)

Scott H. Cameron and S.G. Narayanamurthy. A search problem. *Operations Research*, 12:623–629, 1964. (Cited on page 8 and 9.)

Leonard Carlitz. A sorting function. *Duke Mathematical J.* 38:561–568, 1971. (Cited on page 7.)

Moses Charikar. Ronald Fagin, Venkatesan Guruswami, Jon Kleinberg, Prabhakar Raghavan, and Amit Sahai. Query strategies for priced information. *J. of Computer and System Sciences* 64:785–819, 2002. (Cited on page 13.)

Anis Chelbi and Daoud Ait-Kadi. Inspection strategies for randomly failing systems. In *Handbook of Maintenance Management and Engineering* M. Ben-Daya, S.O. Duffuaa, A. Raouf, J. Knezevic, and D. Ait-Kadi (Eds.) Springer–Verlag, London, 303–335, 2009. (Cited on page 21.)

Zhenlu Chen, Rong Pan, and Lirong Cui. An economic off-line quality control approach for unstable production processes. *Quality Engineering* published online, 2017. (Cited on page 27.)

Vicky Choi and Mordecai J. Golin. Lopsided trees I: analyses. *Algorithmica* 31:240–290, 2001. (Cited on page 20.)

David M. Choy and C.K. Wong. Bounds for optimal α-β binary trees. *BIT* 17:1–15, 1977. (Cited on page 9.)

David M. Choy and C.K. Wong. Optimal α-β trees with capacity constraints. *Acta Informatica* 10:273–296, 1978. (Cited on page 9.)

David M. Choy and C.K. Wong. Construction of optimal $\alpha - \beta$ leaf trees with applications to prefix code and information retrieval. *SIAM J. on Computing* 12:426–446, 1983. (Cited on page 19.)

Yung-Ching Chu. An extended result of Kleitman and Saks concerning binary trees. *Discrete Applied Mathematics*, 10:255–259, 1985. (Cited on page 8.)

Young H. Chun. Economic optimization of off-line inspection procedures with inspection errors. *J. of the Operational Research Society*, 59:863–865, 2008. (Cited on page 25.)

Young H. Chun. Bayesian inspection model for the production process subject to a random failure. *IIE Transactions*, 42:304–316, 2010. (Cited on page 24.)

Kuo-Liang Chung, Wen-Chin Chen, and Ferng-Ching Lin. On the complexity of search algorithms. *IEEE Transactions on Computers*, 41:1172–1176, 1992. (Cited on page 12.)

Ferdinando Cicalese, Tobias Jacobs, Eduardo Laber, and Caio Valentim. The binary identification problem for weighted trees. *Theoretical Computer Science* 459:100–112, 2012. (Cited on page 13.)

Ferdinando Cicalese and Ugo Vaccaro. Binary search with delayed and missing answers. *Information Processing Letters*, 85:239–247, 2003. (Cited on page 16.)

Don Coppersmith, Maria M. Klawe, and Nicholas J. Pippenger. Alphabetic minimax trees of degree at most t. *SIAM J. on Computing*, 15:189–192, 1986. (Cited on page 14.)

Isma Dahmani, Mhand Hifi, and Lei Wu. An exact decomposition algorithm for the generalized knapsack sharing problem. *European Journal of Operational Research* 252:761–774. (Cited on page 7.)

Peter Damaschke. A chip search problem on binary numbers. *LATIN* 1998. (Cited on page 13.)

Annalisa De Bonis, Luisa Gargano, and Ugo Vaccaro. Efficient algorithms for chemical threshold testing problems. *Theoretical Computer Science* 259:493-511, 2001. (Cited on page 13.)

Pavlos S. Efraimidis. (α, β) Fibonacci search. Unpublished manuscript, 2010. (Cited on page 10.)

David E. Ferguson. Fibonaccian searching. *Communications of the ACM*, 3:648, 1960. (Cited on page 6 and 11.)

Thomas S. Ferguson. A problem of minimax estimation with directional information. *Statistics & Probability Letters*, 26:205–211, 1996. (Cited on page 30.)

Alexander Finkelshtein, Yale Herer, Tzvi Raz, and Irad Ben-Gal. Economic optimization of off-line inspection in a process subject to failure and recovery. *IIE Transactions*, 37:995–1009, 2005. (Cited on page 24.)

Robbert Fokkink, Leonhard Geupel, and Kensaku Kikuta, Open problems on search games. *Search Theory: A Game Theoretic Perspective* Steve Alpern, Robbert Fokkink, Leszek Gasieniec, Roy Lindelauf, V.S. Subrahmanian (Eds.) Springer New York, 75–81, 2013. (Cited on page 30.)

Robbert Fokkink and Misha Stassen. An asymptotic solution of Dresher's guessing game. *GameSec* 2011. (Cited on page 29.)

Hiroshi Fujiwara and Tobias Jacobs. On the Huffman and alphabetic tree problem with general cost functions. *Algorithmica*, 69:582–604, 2014. (Cited on page 6 and 15.)

Travis Gagie. A new algorithm for building alphabetic minimax trees. *Fundamenta Informaticae*, 97:321–329, 2009. (Cited on page 14.)

Shmuel Gal. A discrete search game. *SIAM J. on Applied Mathematics*, 27:641–648, 1974. (Cited on page 29.)

Shmuel Gal. A stochastic search game. *SIAM J. on Applied Mathematics*, 34:205–210, 1978. (Cited on page 29.)

Shmuel Gal, Boris Bachelis, and Aharon Ben-Tal. On finding the maximal range of validity of a constrained system. *SIAM J. on Control and Optimization* 16:473-503, 1978. (Cited on page 16 and 17.)

Michael R. Garey and F.K. Hwang. Isolating a single defective using group testing. *J. of the American Statistical Association* 69:151–153, 1974. (Cited on page 6.)

Adriano M. Garsia and Michelle L. Wachs. A new algorithm for minimum cost binary trees. *SIAM J. on Computing*, 6:622–642, 1977. (Cited on page 2 and 4.)

Xiaofeng Gao, Yongtian Yang, Guihai Chen, Xin Lu, and Jiaofei Zhong. Global optimization for multi-channel wireless data broadcast with AH-tree indexing scheme. *IEEE Transactions on Computers* 65:2104–2117. (Cited on page 20.)

Pawel Gawrychowski. Alphabetic minimax trees in linear time. *CSR* 2013 (Cited on page 14.)

Edgar N. Gilbert. Games of identification and convergence. *SIAM Review* 4:16–24, 1962. (Cited on page 29.)

Edgar N. Gilbert and Edward F. Moore. Variable length binary encodings. *The Bell System Technical J.*, 38:933–968, 1959. (Cited on page 3, 4, and 5.)

L. Gotlieb. Optimal multi-way search trees. *SIAM J. on Computing*, 10:422–433, 1981. (Cited on page 6 and 19.)

L. Gotlieb and D. Wood. The construction of optimal multiway search trees and the monotonicity principle. *International J. of Computer Mathematics*, 9:17–24, 1981. (Cited on page 19.)

Hans W. Gottinger. On a problem of optimal search. *Zeitschrift für Operations Research* 21:223–231, 1977. (Cited on page 6.)

Pankaj Gupta, Balaji Prabhakar, and Stephen Boyd. Near-Optimal depth-constrained codes. *IEEE Transactions on Information Theory* 50:3294–3298, 2004. (Cited on page 15.)

Refael Hassin. On maximizing functions by Fibonacci search. *The Fibonacci Quarterly*, 17:347–351, 1981. (Cited on page 7.)

Refael Hassin. A dichotomous search for a geometric random variable. *Operations Research*, 32, 1984. (Cited on page 8, 21, and 22.)

Refael Hassin and Mordechai Henig. Dichotomous search for random objects on an interval. *Mathematics of Operations Research*, 9:301–308, 1984. (Cited on page 18.)

Refael Hassin and Mordechai Henig. Monotonicity and efficient computation of optimal dichotomous search. *Discrete Applied Mathematics*, 46:221–234, 1993. (Cited on page 5, 6, 11, and 15.)

Refael Hassin and Reuven Hotovely. Asymptotic analysis of dichotomous search with search and travel costs. *European J. of Operational Research*, 58:78–89, 1992. (Cited on page 11.)

Refael Hassin and Nimrod Megiddo. An optimal algorithm for finding all the jumps of a monotone step-function. *J. of Algorithms*, 6:265–274, 1985. (Cited on page 18.)

Qi-Ming He, Yigal Gerchak, and Abraham Grosfeld-Nir. Optimal inspection order when process failure rate is constant. *International J. of Reliability, Quality and Safety Engineering*, 3:25–41, 1996. (Cited on page 22.)

Yale T. Herer and Tzvi Raz. Optimal parallel inspection for finding the first nonconforming unit in a batch - an information theoretic approach. *Management Science*, 46:845–857, 2000. (Cited on page 18 and 22.)

Karl Hinderer. On dichotomous search with direction-dependent costs for a uniformly hidden object. *Optimization*, 21:215–229, 1990. (Cited on page 9.)

Karl Hinderer and Michael Stieglitz. Isotonicity of minimizers in polychotomous discrete interval search via lattice programming. *Mathematical Methods of Operations Research*, 51:139–173, 2000. (Cited on page 6 and 14.)

Micha Hofri. On searching an ordered list: are two yardsticks better than one? unpublished manuscript, 1987. (Cited on page 12.)

Yasuichi Horibe. An entropy view of Fibonacci trees. *The Fibonacci Quarterly*, 20:168–178, 1982. (Cited on page 9.)

Yasuichi Horibe. Note on Fibonacci trees and their optimality. *The Fibonacci Quarterly*, 21:118–128, 1983. (Cited on page 9.)

Scott W. Hornick, Sanjeev R. Maddila, Ernst P. Mücke, Harald Rosenberg, Steven S. Skiena, and Ioannis G. Tollis. Searching on a tape. *IEEE Transactions on Computers*, 39:1265–1271, 1990. (Cited on page 11.)

Michael Horstein. Sequential transmission using noiseless feedback. *IEEE Transactions on Information Theory* 9:136–143, 1963. (Cited on page 15.)

A.J. Hu. Selection of the optimum uniform partition search. *Computing*, 37:261–264, 1986. (Cited on page 10.)

T. C. Hu. A new proof of the T-C algorithm. *SIAM J. on Applied Mathematics* 25:83-94, 1973. (Cited on page 4.)

T.C. Hu, Daniel J. Kleitman, and J.K. Tamaki. Binary trees optimum under various criteria. *SIAM J. on Applied Mathematics*, 37:246–256, 1979. (Cited on page 7 and 14.)

T.C. Hu, L.Lawrence Larmore, and J. David Morgenthaler. Optimal integer alphabetic trees in linear time. *ESA* 2005. (Cited on page 4.)

T.C. Hu and J. David Morgenthaler. Optimum alphabetic binary trees. *Lecture Notes in Computer Science, Springer-Verlag*, 1120:234–243, 1996. (Cited on page 4.)

T.C. Hu and K.C. Tan. Path length of binary search trees. *SIAM J. on Applied Mathematics*, 22:225–234, 1972. (Cited on page 4.)

T.C. Hu and Alan C. Tucker. Optimal computer-search trees and variable-length alphabetical codes. *SIAM J. on Applied Mathematics*, 21:514–532, 1971. (Cited on page 4.)

T.C. Hu and P.A. Tucker. Optimal alphabetic trees for binary search. *Information Processing Letters*, 67:137–140, 1998. (Cited on page 3.)

T.C. Hu and Michelle L. Wachs. Binary search on a tape. *SIAM J. on Computing*, 16:573–590, 1987. (Cited on page 10.)

David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40:1098–1101, 1952. (Cited on page 4.)

Hsien-Kuei Hwang and Tsung-Hsi Tsai. An asymptotic theory for recurrence relations based on minimization and maximization. *Theoretical Computer Science*, 290:1475–1501, 2003. (Cited on page 4.)

Alon Itai. Optimal alphabetic trees. *SIAM J. on Computing*, 5:9–18, 1976. (Cited on page 6, 15, 18, and 19.)

Selmer M. Johnson. A search game. *Annals of Mathematical Studies* 52:39-48, 1964. (Cited on page 29.)

Sungwon Jung, Byungkyu Lee, and Sakti Pramanik. A tree-structured index allocation method with replication over multiple broadcast channels in wireless environments. *IEEE Transactions on Knowledge and Data Engineering* 17:311–325. (Cited on page 20.)

Sanjiv Kapoor and Edward M. Reingold. Optimum lopsided binary trees. *J. of the Association for Computing Machinery* 36:573–590, 1989. (Cited on page 9.)

Richard M. Karp A generalization of binary search. *WADS* 1993. (Cited on page 18.)

Richard M. Karp and Robert Kleinbe4rg. Noisy binary search and its applications. *SODA* 2007. (Cited on page 16.)

Richard M. Karp and Willard Miranker. Parallel minimax search for a maximum. *J. of Combinatorial Theory* 4:19–35, 1968. (Cited on page 7.)

Jack C. Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematics Society*, 4:502–506, 1953. (Cited on page 7.)

Jeffrey H. Kingston. A new proof of the Garsia-Wachs algorithm. *J. of Algorithms* 9:129-136, 1988. (Cited on page 4.)

David G. Kirkpatrick and Maria M. Klawe. Alphabetic minimax trees. *SIAM J. on Computing*, 14:514–526, 1985. (Cited on page 14.)

Maria M. Klawe and Brendan Mumey. Upper and lower bounds on constructing alphabetic binary trees. *SIAM J. on Discrete Mathematics*, 8:638–651, 1995. (Cited on page 4.)

Daniel J. Kleitman and Michael Ezra Saks. Set orderings requiring costliest alphabetic binary trees. *SIAM J. on Algorithms in Discreet Mathematics*, 2:142–146, 1981. (Cited on page 8.)

William J. Knight. Search in an ordered array having variable probe cost. *SIAM J. on Computing*, 17:1203–1214, 1988. (Cited on page 12 and 13.)

Donald E. Knuth. Optimum binary search trees. *Acta Informatica*, 1:14–25, 1971. (Cited on page 4 and 5.)

Stephen Kwek and Kurt Mehlhorn. Optimal search for rationals. *Information Processing Letters*, 86:23–26, 2003. (Cited on page 20.)

Eduardo S. Laber, Ruy L. Milidiú, and Artur A. Pessoa. A strategy for searching with different access costs. *Theoretical Computer Science* 287:571–584, 2002. (Cited on page 13.)

Eduardo S. Laber, Ruy L. Milidiú, and Artur A. Pessoa. On binary searching with nonuniform costs. *SIAM J. on Computing* 31:1022-1047, 2002. (Cited on page 13.)

Lawrence L. Larmore. A subquadratic algorithm for constructing approximately optimal binary search trees. *J. of Algorithms* 8:579-591, 1987. (Cited on page 4.)

Lawrence L. Larmore and Teresa M. Przytycka. A fast algorithm for optimal height-limited alphabetic binary trees. *SIAM J. on Computing*, 23:1283–1312, 1994. (Cited on page 15.)

Lawrence L. Larmore and Teresa M. Przytycka. A parallel algorithm for optimum height-limited alphabetic binary trees. *J. of Parallel and Distributed Computing*, 35:49–56, 1996. (Cited on page 15.)

Lawrence L. Larmore and Teresa M. Przytycka. The optimal alphabetic tree problem revisited. *J. of Algorithms*, 28:1–20, 1998. (Cited on page 4.)

Yanzhe (Murray) Lei, Stefanus Jasin, and Amitabh Sinha. Generalized bisection search for constrained optimization with noisy observations. Tech. Report, 2016. (Cited on page 7 and 16.)

Timo Leipälä. On a generalization of binary search. *Information Processing Letters*, 8:230–233, 1979. (Cited on page 4.)

Shou-Chih Lo and Arbee L.P. Chen. Optimal index and data allocation in multiple broadcast channels. *International Conference on Data Engineering*, 2000 (Cited on page 20.)

Yannis Manolopoulos, John G. Kollias, and F. Warren Burton. Batched interpolation search. *The Computer J.* 30:565–568, 1987. (Cited on page 7 and 18.)

Yannis Manolopoulos, John G. Kollias, and Michael Hatzopoulos. Sequential vs. binary batched searching. *The Computer J.* 29:368–372, 1986. (Cited on page 7 and 18.)

John J. McCall. Maintenance policies for stochastically failing equipment: a survey. *Management Science* 11:493–524, 1965. (Cited on page 21.)

Nimrod Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4:414–424, 1979. (Cited on page 20.)

Kurt Mehlhorn. A best possible bound for the weighted path length of binary search trees. *SIAM J. on Computing* 6:235-239. 1977. (Cited on page 4.)

J. David Morgenthaler and T.C. Hu. Optimal alphabetic ternary trees. Tech. report, 2014. (Cited on page 20.)

Robert Morris. Some theorems on sorting. *SIAM J. on Applied Mathematics*, 17:1–6, 1969. (Cited on page 6.)

Satoru Murakami. A dichotomous search. *J. of the Operations Research Society of Japan*, 14:127–142, 1971. (Cited on page 8.)

Satoru Murakami. A dichotomous search with travel cost. *J. of the Operations Research Society of Japan*, 19:245–254, 1976. (Cited on page 10.)

S.V. Nagaraj. Optimal binary search trees. *Theoretical Computer Science* 188:1–44, 1997. (Cited on page 2.)

Narao Nakatsu. Bounds on the redundancy of binary alphabetical codes. *IEEE Transactions on Information Theory* 37:1225-1229, 1991. (Cited on page 3.)

Gonzalo Navarro, Eduardo F. Barbosa, Ricardo Baeza-Yates, Walter Cunto, and Nívio Ziviani. Binary searching with nonuniform costs and its application to text retrieval. *Algorithmica* 27:145–169, 2000. (Cited on page 12 and 13.)

Seiichi Nishihara and Hiroji Nishino. Binary search revisited: another advantage of Fibonacci search. *IEEE Transactions on Computers*, C-36:1132–1135, 1987. (Cited on page 6, 11, and 12.)

L. T. Oliver and D. J. Wilde. Symmetric sequential minimax search for an optimum. *Fibonacci Quarterly* 2:24–41, 1964. (Cited on page 7.)

Thomas Ottmann, Arnold L. Rosenberg, Hans-Werner Six, and Derick Wood. Binary search trees with binary comparison cost. *International J. of Computer and Information Sciences* 13:77–101, 1984. (Cited on page 9.)

K.J. Overholt. Efficiency of the Fibonacci search method. *BIT Numerical Mathematics*, 13:92–96, 1973. (Cited on page 7.)

Christos H. Papadimitriou. Efficient search for rationals. *Information Processing Letters*, 8:1–4, 1979. (Cited on page 20.)

Yehoshua Perl, Michael R. Garey, and Shimon Even. Efficient generation of optimal prefix code: equiprobable words using unequal cost letters. *J. of the Association for Computing Machinery* 22:202–214, 1975. (Cited on page 19.)

Yehoshua Perl, Alon Itai, and Haim Avni. Interpolation search – a $\log \log N$ search. *Communication of the ACM* 21:550–553, 1978. (Cited on page 7.)

W. Wesley Peterson. Addressing for random-access storage. *IBM J. of Research and Development* 1:131-132, 1957. (Cited on page 7.)

Roberto De Prisco and Alfredo De Santis. On binary search trees. *Information Processing Letters* 45:249–253, 1993. (Cited on page 3.)

A.L. Rastsvetaev and Lev D. Beklemishev. On the query complexity of finding a local maximum point. *Information Processing Letters* 84:327–332, 2002. (Cited on page 7.)

Tzvi Raz, Yale T. Herer, and Avraham Grosfeld-Nir. Economic optimization of off-line inspection. *IIE Transactions*, 32:205–217, 2000. (Cited on page 23, 24, and 25.)

Steven P. Reiss. Rational search. *Information Processing Letters*, 8:89–90, 1979. (Cited on page 20.)

Diane J. Reyniers. A high-low search model of inventories with time delay. *Engineering Costs and Production Economics*, 15:417–422, 1988. (Cited on page 28.)

Diane J. Reyniers. Interactive high-low search: the case of lost sales. *J. of the Operational Research Society*, 40:769–780, 1989. (Cited on page 28.)

Diane J. Reyniers. Supply decisions for unknown linearly increasing demand. *Engineering Costs and Production Economics* 17:389–393, 1989. (Cited on page 28.)

Diane J. Reyniers. A high-low search algorithm for a newsboy problem with delayed information feedback. *Operations Research*, 38:838–846, 1990. (Cited on page 28.)

Diane J. Reyniers. Information and rationality asymmetries in a simple high-low search wage model. *Economics Letters*, 38:479–486, 1992. (Cited on page 28 and 29.)

Diane J. Reyniers. A dynamic model of collective bargaining. *Computational Economics* 11:205–220, 1998. (Cited on page 29.)

Diane J. Reyniers. Relative impatience determines preference between contract bargaining and repeated bargaining. *International J. of Game Theory* 29:165–176, 2000. (Cited on page 29.)

Jorma Rissanen. Bounds for weight balanced trees. *IBM J. of Research and Development* 17:101–105, 1973. (Cited on page 4.)

Ronald L. Rivest, Albert R. Meyer, Daniel J. Kleitman, Karl Winklmann and Joel Spencer. Coping with errors in binary search procedures. *J. of Computer and System Sciences*, 20:396–404, 1980. (Cited on page 14 and 16.)

Wojciech Rytter. Trees with minimum weighted path length. Chapter 14 in *Handbook of Data Structures and Applications* (eds. Dinesh P. Mehta, Sartaj Sahni) CRC Press, 2005. (Cited on page 2.)

Nicola Santoro and Jefferey B. Sidney. Interpolation-binary search. *Information Processing Letters* 20:179–181, 1985. (Cited on page 7.)

Biswajit Sarkar and Sharmila Saren. Product inspection policy for an imperfect production system with inspection errors and warranty cost. *European J. of Operational Research* 248:263–271, 2016. (Cited on page 21.)

Narayanan Shivakumar and Suresh Venkatasubramanian. Efficient indexing for broadcast based wireless systems. *Mobile Networks and Application* 1:443–446, 1996. (Cited on page 20.)

Frank Schulz. Trees with exponentially growing costs. *Information and Computation* 206:569–578, 2008. (Cited on page 6 and 15.)

Y.S. Sherif and L. Smith. Optimal maintenance models for systems subject to failure - a review. *Naval Research Logistics* 28:47–74, 1981. (Cited on page 21.)

Shey-Huei Sheu, Yan-Chun Chen, Weng-Ying Wang, and N.H. Shin. Economic optimization of off-line inspection with inspection errors. *J. of the Operational Research Society*, 54:888–895, 2003. (Cited on page 25.)

Dafna Sheinwald. On binary alphabetic codes. *DCC*, 1992. (Cited on page 3.)

Man-tak Shing. Optimum ordered bi-weighted binary trees. *Information Processing Letters* 17:67–70, 1983. (Cited on page 9.)

David Spuler. The optimal binary search tree for Andersson's search algorithm. *Acta Informatica* 30:405–407, 1993. (Cited on page 3.)

Jayme L. Szwarcfiter, Gonzalo Navarro, Ricardo Baeza-Yates, Joísa de S. Oliveira, Walter Cunto, and Nívio Ziviani. Optimal binary search trees with costs depending on the access paths. *Theoretical Computer Science* 290:1799-1814, 2003. (Cited on page 12.)

W.C. Tsai and Chih-Hsiung Wang. Economic optimization for an off-line inspection, disposition and rework model. *Computers and Industrial Engineering*, 61:891–896, 2011. (Cited on page 25.)

Avinoam Tzimerman and Yale Herer. Off-line inspections under inspection errors. *IIE Transactions*, 41:626–641, 2009. (Cited on page 26 and 27.)

Ben Varn. Optimal variable length codes. *Information and Control* 19:289-301, 1971. (Cited on page 9 and 19.)

Michelle L. Wachs. On an efficient dynamic programming technique of F.F. Yao. *J. of Algorithms*, 10:518–530, 1989. (Cited on page 11.)

Rolf Waeber, Peter I. Frazier, and Shane G. Henderson. Bisection search with noisy responses. *SIAM J. on Control and Optimization* 51:2261–2279, 2013. (Cited on page 15 and 16.)

Chih-Hsiung Wang. Economic off-line quality control strategy with two types of inspection errors. *European J. of Operational Research*, 179:132–147, 2007. (Cited on page 25.)

Chih-Hsiung Wang and Hsiao Ping Chuang. Integrated on-line and off-line quality control for products with destructive testing. *J. of Information & Optimization Sciences* 32:369–380, 2011. (Cited on page 23.)

Chih-Hsiung Wang and Chen-Chien Hung. An offline inspection and disposition model incorporating discrete Weibull distribution and manufacturing variation. *J. of the Operations Research Society of Japan* 51:155–165, 2008. (Cited on page 26.)

Chih-Hsiung Wang, N-H. Shih, and W.C. Tsai. Utilizing the information theory of entropy to solve an off-line inspection problem. *4OR- A Quarterly J. of Operations Research*, 9:391–401, 2011. (Cited on page 27.)

Wen-Ying Wang, Shey-Huei Sheu, Yan-Chun Chen, and Der-Juinn Horng. On a more general formulation of off-line inspection with inspection errors. *J. of the Operational Research Society* 59:865-867, 2008. (Cited on page 25.)

Wen-Ying Wang, Shey-Huei Sheu, Yan-Chun Chen, and Der-Juinn Horng. Economic optimization of off-line inspection with rework consideration. *European J. of Operational Research*, 194:807–813, 2009. (Cited on page 24 and 25.)

Russell L. Wessner. Optimal alphabetic search trees with restricted maximal height. *Information Processing Letters*, 4:90–94, 1976. (Cited on page 6 and 15.)

Eugene Wong. A linear search problem. *SIAM Review*, 6:168–174, 1964. (Cited on page 6.)

F. Frances Yao. Efficient dynamic programming using quadrangle inequalities. *STOC* 1980. (Cited on page 5.)

F. Frances Yao. Speed-up in dynamic programming. *SIAM J. on Algebraic and Discrete Methods*, 3:532–540, 1982. (Cited on page 5.)

Andrew C. Yao and F. Frances Yao. The complexity of searching an ordered random table. *FOCS* 1976. (Cited on page 7.)

Raymond W. Yeung. *Alphabetic codes revisited. IEEE Transactions on Information Theory* 37:564–572, 1991. (Cited on page 3 and 15.)

Cun-Quan Zhang. Optimal alphabetic binary tree for a nonregular cost function. *Discrete Applied Mathematics* 8:307–312, 1984. (Cited on page 14.)

Eitan Zemel. On search over rationals. *Operations Research Letters*, 1:34–38, 1981. (Cited on page 20.)