Sequential Optimization of Binary Search Trees for Multiple Cost Functions

Maram Alnafie Igor Chikalov Shahid Hussain Mikhail Moshkov

Mathematical and Computer Sciences & Engineering Division,
King Abdullah University of Science and Technology,
Thuwal 23955-6900, Saudi Arabia
Email: {maram.nafie, igor.chikalov, shahid.hussain, mikhail.moshkov}@kaust.edu.sa

Abstract

In this paper, we present a methodology to model the optimal binary search tree problem as a directed acyclic graph to extract all possible optimal solutions. We provide a mechanism to find optimal binary search trees relative to different types of cost functions, sequentially. We prove that for a set of n keys our optimization procedure makes $O(n^3)$ arithmetic operations per cost function such as weighted depth or average weighted depth.

Keywords: Binary search trees, dynamic programming, cost functions, sequential optimization

1 Introduction

The *binary search tree* is an important data structure (see Mehlhorn (1984) or Cormen et al. (2001) for more discussion about binary search trees), which can be used for a variety of applications including a computerized translation of words from one language to another (Cormen et al. 2001) or a lookup table for words with known frequencies of occurrences (see Knuth (1971)), etc.

A typical computerized word translator application uses keys as well as dummy keys. The keys mean the words in a language to be translated and dummy keys represent the words for which the system cannot translate. Here, the usual problem is, for the given probabilities of keys and dummy keys, to find a binary search tree with minimum average depth.

We further generalize this problem: consider weights of keys (which correspond to complexity of key "computation") and try to construct a binary search tree with minimum average weighted depth. We also study binary search trees with minimum weighted depth to minimize the worst-case time complexity of the trees.

The novelty of this work is connected with the following: we represent all possible binary search trees by a directed acyclic graph G and apply to this graph a procedure of optimization relative to average weighted depth ψ. As a result we obtain a subgraph G_ψ of G which describes the whole set $\mathcal{T}$ of binary search trees with minimum average weighted depth. We further apply to this graph the procedure of optimization relative to weighted depth ϕ. As a result we obtain a subgraph $(G_\psi)_\phi$ of G_ψ which describes a subset of the set of all binary search trees from $\mathcal{T}$ with minimum weighted depth. The second step of optimization can improve the worst-case time complexity of the constructed trees.

The approach presented in this paper is an extension of dynamic programming. Similar techniques have been used extensively for optimization of decision trees (Moshkov & Chikalov 2000, 2003, Chikalov et al. 2005, Chikalov 2001, Alkhalid et al. 2010) for a wide range of cost functions.

This paper consists of nine sections. In Sect. 2, we provide preliminary concepts to understand and model binary search trees. In Sect. 3, we discuss the cost functions in detail. We discuss the representation of sets of binary search trees in Sect. 4. In Sect. 5, we present the procedure of optimization. Section 6 shows an example and depicts how to carry the procedure of optimization for a single cost function. We extend the optimization procedure for application of different cost functions sequentially in Sect. 7. In Sect. 8, we provide the computational complexity of our procedure and we conclude the paper in Sect. 9.

2 Preliminaries

Let we have a sequence of n distinct keys $k_1, k_2, \ldots, k_n$ such that $k_1 < k_2 < \cdots < k_n$ with probabilities $p_1, p_2, \ldots, p_n$ and $n+1$ dummy keys $d_0, d_1, \ldots, d_n$ with probabilities $q_0, q_1, \ldots, q_n$ such that for $i = 1, 2, \ldots, n-1$, the dummy key d_i corresponds to all values between k_i and k_{i+1} with d_0 representing all values less than k_1 and similarly d_n representing all values greater than k_n. The sum of probabilities is one, i.e.,

$$\sum_{i=0}^{n} p_i + \sum_{i=0}^{n} q_i = 1.$$

A binary search tree consists of keys as the internal nodes and dummy keys as the leaves. A search for some x is considered *successful* if x is one of the keys k_i and *unsuccessful* otherwise. Each key k_i, $1 \le i \le n$, has an associated weight w_i that is a positive number which can be interpreted as the complexity of comparison of k_i with x.

We consider the problem of optimizing *binary search trees* and denote it by $S(0, n)$ for $k_1, \ldots, k_n$ and $d_0, d_1, \ldots, d_n$. Furthermore, we consider the subproblems of initial problem $S(i, j)$ for $0 \le i \le j \le n$. For $i = j$, the subproblem $S(i, i)$ is the optimization problem for the binary search trees for only one dummy key d_i.

For $i < j$, the subproblem $S(i, j)$ is the optimization problem for binary search trees for the keys $k_{i+1}, \ldots, k_j$ and dummy keys $d_i, d_{i+1}, \ldots, d_j$.

Figure 1: The binary search tree from $T(i,i)$.

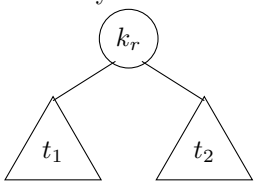

Figure 2: The binary search tree $\text{TREE}(k_r, t_1, t_2)$.

We define inductively the set $T(i,j)$ of all binary search trees for $S(i,j)$. For $i = j$, the set $T(i,i)$ contains exactly one binary search tree as depicted in Figure 1.

For $i < j$ we have

$$T(i,j) = \bigcup_{r=i+1}^{j} \{\text{TREE}(k_r, t_1, t_2) : t_1 \in T(i, r-1), t_2 \in T(r,j)\},$$

where $\text{TREE}(k_r, t_1, t_2)$ is depicted in Figure 2.

3 Cost Functions

We consider the cost functions which are given in the following way: values of considered cost function ψ (nonnegative function), are defined by induction on pair $(S(i,j), t)$, where $S(i,j)$ is a subproblem, $0 \le i \le j \le n$, and $t \in T(i,j)$. Let $i = j$ and t be a tree represented as in Figure 1 then $\psi(S(i,i), t) = 0$. Let $i < j$ and t be a tree depicted as in Figure 2 where $r \in \{i+1, \ldots, j\}$, $t_1 \in T(i, r-1)$, and $t_2 \in T(r,j)$. Then

$$\psi(S(i,j), t) = F(\pi(i,j), w_r, \psi(S(i, r-1), t_1), \psi(S(r,j), t_2)),$$

where $\pi(i,j)$ is the sum of probabilities of keys and dummy keys from $S(i,j)$, i.e.,

$$\pi(i,j) = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l,$$

w_r is the weight of the key k_r, and $F(\pi, w, \psi_1, \psi_2)$ is an operator which transforms the considered tuple of nonnegative numbers into a nonnegative number.

The considered cost function is called *monotonic* if for arbitrary nonnegative numbers a, b, c_1, c_2, d_1 and d_2 from inequalities $c_1 \le d_1$ and $c_2 \le d_2$ the inequality $F(a, b, c_1, c_2) \le F(a, b, d_1, d_2)$ follows.

The considered cost function is called *strictly monotonic* if it is monotonic and for arbitrary nonnegative numbers a, b, c_1, c_2, d_1 and d_2 from inequalities $c_1 \le d_1$, $c_2 \le d_2$, and inequality $c_i < d_i$ (which is true for some $i \in \{1, 2\}$), the inequality $F(a, b, c_1, c_2) < F(a, b, d_1, d_2)$ follows.

Now we take a closer view of some cost functions.

Weighted depth. We attach a weight to each path from the root to each node of the tree t which is equal to the sum of weights of the keys in this path. Then $\psi(S(i,j), t)$ is the maximum weight of a path from the root to a node of t. For this cost function $F(\pi, w, \psi_1, \psi_2) = w + \max\{\psi_1, \psi_2\}$. The cost function is monotonic.

Average weighted depth. For an arbitrary key or a dummy key h from $S(i,j)$ we denote $\Pr(h)$ its probability and by $w(h)$ we denote the weight of the path from the root of t to h. Then $\psi(S(i,j), t) = \sum_h w(h) \cdot \Pr(h)$, where we take the sum on all keys and dummy keys h from $S(i,j)$. For this cost function $F(\pi, w, \psi_1, \psi_2) = w \cdot \pi + \psi_1 + \psi_2$. The cost function is strictly monotonic.

4 Representing All Possible Binary Search Trees

Now we describe a *directed acyclic graph* (DAG) G_0, which allows us to represent all binary search trees from $T(i,j)$ for each subproblem $S(i,j)$, $0 \le i \le j \le n$. The set of vertices of this graph coincides with the set

$$\{S(i,j) : 0 \le i \le j \le n\}.$$

If $i = j$ then $S(i,i)$ has no outgoing edges. For $i < j$, $S(i,j)$ has exactly $2(j-i)$ outgoing edges. For $r = i+1, \ldots, j$ exactly two edges start from $S(i,j)$ and finish in $S(i, r-1)$ and $S(r,j)$, respectively. These edges are labeled with the index r, we call these edges a *rigid pair* of edges with index r.

Let G be a subgraph of G_0 which is obtained from G_0 by removal of some rigid pairs of edges such that for each vertex $S(i,j)$ with $i < j$ at least one rigid pair outgoing from $S(i,j)$ remains intact.

Now for each vertex $S(i,j)$ we define, by induction, the set $T_G(i,j)$ of binary search trees corresponding to $S(i,j)$ in G. For $i = j$, we have $T_G(i,i)$ containing only the trivial binary search tree as depicted in Figure 1. For $i < j$, let $R_G(i,j)$ be the set of indexes of remaining rigid pairs outgoing from $S(i,j)$, then

$$T_G(i,j) = \bigcup_{r \in R_G(i,j)} \{\text{TREE}(k_r, t_1, t_2) : t_1 \in T_G(i, r-1), t_2 \in T_G(r,j)\},$$

where $\text{TREE}(k_r, t_1, t_2)$ is as depicted in Figure 2.

One can show that $T_{G_0}(i,j) = T(i,j)$ for $0 \le i \le j \le n$.

5 Procedure of Optimization

Let ψ be a cost function, we consider the procedure of optimization of binary search trees corresponding to vertices in G relative to ψ.

For each vertex $S(i,j)$ of the graph G we mark this vertex by the number $\psi^*_{(i,j)}$, which can be interpreted as the minimum value of ψ on $T_G(i,j)$. We can compute the value $\psi^*_{(i,j)}$ for $i < j$ as:

$$\psi^*_{(i,j)} = \min_{r \in R_G(i,j)} F(\pi(i,j), w_r, \psi^*_{(i,r-1)}, \psi^*_{(r,j)}),$$

where $\psi^*_{(i,i)} = 0$.

Let $\Psi_{(i,r,j)}$ be defined as following:

$$\Psi_{(i,r,j)} = F(\pi(i,j), w_r, \psi^*_{(i,r-1)}, \psi^*_{(r,j)}),$$

now we remove from G each rigid pair with the index r starting in $S(i,j)$ such that

$$\Psi_{(i,r,j)} > \psi^*_{(i,j)}.$$

Let G_ψ be the resulting subgraph obtained from G. It is clear that for each vertex $S(i,j)$, $i < j$, at least one rigid pair outgoing from $S(i,j)$ is left intact.

We denote $T^*_{G,\psi}(i,j)$ the set of all trees from $T_G(i,j)$ that have minimum cost relative to ψ.

Following theorem summarizes the procedure of optimization for monotonic cost function.

Theorem 1 *Let ψ be a monotonic cost function. Then for every i and j, $0 \le i \le j \le n$, $\psi^*_{(i,j)}$ is the minimum cost of binary search trees in $T_G(i,j)$ and $T_{G_\psi}(i,j) \subseteq T^*_{G,\psi}(i,j)$.*

Proof is by induction on $j-i$. If $i = j$ then $T_G(i,j)$, $T^*_{G,\psi}(i,j)$, and $T_{G_\psi}(i,j)$ contain only one tree as depicted in Figure 1. Furthermore, the cost of this tree is zero as well as $\psi^*_{(i,j)} = 0$ by definition. So if $j-i = 0$ the considered statement holds. Let for some $m \ge 0$ and for each pair (i,j) such that $j - i \le m$ this statement hold.

Let us consider a pair (i,j) such that $j-i = m+1$. Then

$$\psi^*_{(i,j)} = \min_{r \in R_G(i,j)} \Psi_{(i,r,j)},$$

since $i < r \le j$, $r - 1 - i \le m$ and $j - r \le m$. By the induction hypothesis the considered statement holds for the pairs $(i, r-1)$ and (r,j) for each $r \in R_G(i,j)$. Furthermore, for $r \in R_G(i,j)$, let

$$T^r_G(i,j) = \{\text{TREE}(k_r, t_1, t_2) : t_1 \in T_G(i, r-1), t_2 \in T_G(r,j)\}.$$

It is clear that

$$T_G(i,j) = \bigcup_{r \in R_G(i,j)} T^r_G(i,j). \quad (1)$$

Using monotonicity of the cost function we know that $\Psi_{(i,r,j)}$ is the minimum cost of a tree from $T^r_G(i,j)$. From here and (1) it follows that $\psi^*_{(i,j)}$ is the minimum cost of a tree in $T_G(i,j)$. In G_ψ the only rigid pairs outgoing from $S(i,j)$ are those for which $\Psi_{(i,r,j)} = \psi^*_{(i,j)}$, where r is the index of the considered pair. Let $R_{G_\psi}(i,j)$ be the set of indexes r of rigid pairs outgoing from $S(i,j)$ in G_ψ.

It is clear that the cost of each element in the set

$$T^r_{G_\psi}(i,j) = \{\text{TREE}(k_r, t_1, t_2) : t_1 \in T_{G_\psi}(i, r-1), t_2 \in T_{G_\psi}(r,j)\}$$

is equal to $\Psi_{(i,r,j)} = F(\pi(i,j), w_r, \psi^*_{(i,r-1)}, \psi^*_{(r,j)})$, since the cost of each tree in $T_{G_\psi}(i, r-1)$ is equal to $\psi^*_{(i,r-1)}$ and the cost of each tree in $T_{G_\psi}(r,j)$ is equal to $\psi^*_{(r,j)}$. Furthermore,

$$T_{G_\psi}(i,j) = \bigcup_{r \in R_{G_\psi}(i,j)} T^r_{G_\psi}(i,j),$$

we have that each binary search tree in $T_{G_\psi}(i,j)$ has the cost equal to $\psi^*_{(i,j)}$. ∎

Furthermore, in case of a strictly monotonic cost function we have stronger result, summarized in terms of the following theorem.

Theorem 2 *Let ψ be a strictly monotonic cost function. Then for every i and j, $0 \le i \le j \le n$, $\psi^*_{(i,j)}$ is the minimum cost of binary search trees in $T_G(i,j)$ and $T_{G_\psi}(i,j) = T^*_{G,\psi}(i,j)$.*

Since ψ is a strictly monotonic cost function, ψ is also a monotonic cost function. By Theorem 1, $\psi^*_{(i,j)}$ is the minimum cost of binary search trees in $T_G(i,j)$ and $T_{G_\psi}(i,j) \subseteq T^*_{G,\psi}(i,j)$. We only need to prove that $T^*_{G,\psi}(i,j) \subseteq T_{G_\psi}(i,j)$. We prove this statement by induction on $j-i$. If $j - i = 0$ then both sets $T^*_{G,\psi}(i,j)$ and $T_{G_\psi}(i,j)$ contain the only tree as depicted in Figure 1. So $T^*_{G,\psi}(i,j) = T_{G_\psi}(i,j)$ and the considered statement holds for $j-i = 0$. Let for some $m \ge 0$ and for each pair (i,j) such that $j - i \le m$ this statement hold.

Let us consider a pair (i,j) such that $j-i = m+1$. Then (we use here the notation from the proof of Theorem 1)

$$T_G(i,j) = \bigcup_{r \in R_G(i,j)} T^r_G(i,j).$$

Let $t \in T^*_{G,\psi}(i,j)$, then there exists $r \in R_G(i,j)$ such that $t \in T^r_G(i,j)$ and t can be represented in the form $\text{TREE}(k_r, t_1, t_2)$ (see Figure 2) where $t_1 \in T_G(i, r-1)$ and $t_2 \in T_G(r,j)$. Therefore, the minimum cost $\Psi_{(i,r,j)}$ of trees from $T^r_G(i,j)$ is equal to $\psi^*_{(i,j)}$. Therefore, it follows that $r \in R_{G_\psi}(i,j)$.

Let us assume that the cost of t_1 is greater than $\psi^*_{(i,r-1)}$ or the cost of t_2 is greater than $\psi^*_{(r,j)}$. Since ψ is strictly monotonic function we have that the cost of t is greater than $\psi^*_{(i,j)}$, which is impossible. Thus $t_1 \in T^*_{G,\psi}(i, r-1)$ and $t_2 \in T^*_{G,\psi}(r,j)$. It is clear that $j - r \le m$ and $r - 1 - i \le m$. Therefore, by induction hypothesis $T_{G_\psi}(i, r-1) \supseteq T^*_{G,\psi}(i, r-1)$ and $T_{G_\psi}(r,j) \supseteq T^*_{G,\psi}(r,j)$. From here it follows that $t_1 \in T_{G_\psi}(i, r-1)$, $t_2 \in T_{G_\psi}(r,j)$ and $t \in T^r_{G_\psi}(i,j)$. Since $r \in R_{G_\psi}(i,j)$ we have $t \in T_{G_\psi}(i,j)$ and $T^*_{G,\psi}(i,j) \subseteq T_{G,\psi}(i,j)$. ∎

6 Example

We consider an example of a simple optimization problem with two keys k_1, k_2 and three dummy keys d_0, d_1 and d_2 with probabilities as $p_1 = 0.4, p_2 = 0.3$ and $q_0 = q_1 = q_2 = 0.1$. We also associate weights with keys: $w_1 = 6$ and $w_2 = 4$. We get the DAG $G = G_0$ (see Figure 3).

We apply the procedure of optimization to the graph G and use the average weighted depth ψ as the cost function. As a result we obtain the subgraph G_ψ of the graph G (see Figure 4) The binary search tree as depicted in Figure 5 corresponds to the vertex $S(0,2)$ of the graph G_ψ. This is the only optimal binary search tree relative to ψ for the considered problem.

7 Sequential Optimization

The optimization procedure presented in this paper allows sequential optimization relative to different cost functions. Let us consider one of the possible scenario.

Initially we get a DAG $G = G_0$. We apply the procedure of optimization to the graph G relative to the average weighted depth ψ. As a result we obtain the subgraph G_ψ of the graph G. By Theorem 2, the set of binary search trees corresponding to the vertex $S(0,n)$ of G_ψ coincides with the set of all binary search trees which have minimum cost relative to ψ. We denote this set by $\mathcal{T}$.

For each vertex $S(i,j)$, $i < j$, in G_ψ at least one rigid pair outgoing from $S(i,j)$ is left intact. So we can apply now the procedure of optimization to the graph G_ψ relative to the weighted depth ϕ. As a

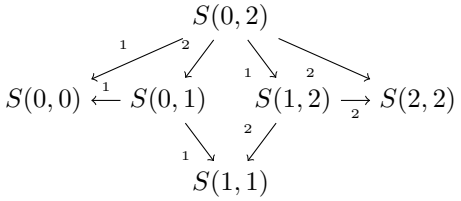

Figure 3: Initial DAG $G = G_0$.

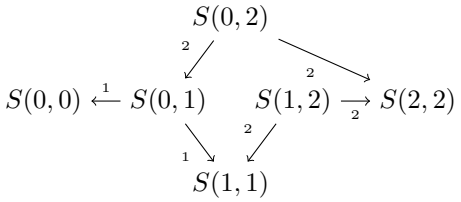

Figure 4: The graph G_ψ.

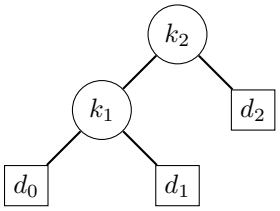

Figure 5: The binary search tree corresponding to the vertex $S(0, 2)$ in G_ψ.

result we obtain the subgraph $(G_\psi)_\phi$. By Theorem 1, the set of binary search trees corresponding to the vertex $S(0, n)$ of $(G_\psi)_\phi$ is a subset of the set of all trees from $\mathcal{T}$ which have minimum cost relative to ϕ.

8 Computational Complexity

Initially, we get a DAG G_0. After optimization relative to a sequence of cost functions we get *subgraph* G of G_0, which is obtained from G_0 by removal of some edges (rigid pairs). It is clear that the number of subproblems $S(i, j)$ for a problem with n keys is $O(n^2)$. We study two types of cost functions, i.e., weighted depth and average weighted depth.

Let us consider the procedure of optimization of G relative to one of these cost functions ψ. This procedure (as outlined in this paper) requires to compute $\psi^*_{(i,j)}$ for each subproblem $S(i, j)$. It is enough to make $O(n)$ arithmetic operations to compute the value $\psi^*_{(i,j)}$ and to remove all rigid pairs for which $\Psi_{(i,r,j)} > \psi^*_{(i,j)}$, if the values $\pi(i, j)$, $\psi^*_{(i,r-1)}$, and $\psi^*_{(r,j)}$ are already known for each $r \in R_G(i, j)$. It is clear that $\pi(i, i) = q_i$ and $\pi(i, j) = \pi(i, i) + p_{i+1} + \pi(i + 1, j)$, if $j > i$. So to compute all values $\pi(i, j)$, $0 \le i \le j \le n$, it is enough to make $O(n^2)$ operations of additions. Therefore the total number of arithmetic operations required to obtain G_ψ from G is $O(n^3)$.

9 Conclusion

We have discussed in detail a procedure to sequentially optimize binary search trees relative to two different types of cost functions. Each step of optimization can be carried out in polynomial-time with respect to the total number of keys. We also have provided a simple example in detail.

References

Alkhalid, A., Chikalov, I. & Moshkov, M. (2010), On algorithm for building of optimal α-decision trees, *in* M. Szczuka et al., ed., 'RSCTS 2010', LNAI 6086, Springer, Heidelberg, pp. 438–445.

Chikalov, I. (2001), On algorithm for constructing of decision trees with minimal number of nodes, *in* 'RSCTC '00: Revised Papers from the Second International Conference on Rough Sets and Current Trends in Computing', Springer, London, UK, pp. 139–143.

Chikalov, I., Moshkov, M. & Zelentsova, M. (2005), On optimization of decision trees, *in* 'Transactions on Rough Sets IV, Lecture Notes in Computer Science 3700', Springer, Heidelberg, pp. 18–36.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2001), *Introduction to Algorithms*, second edn, MIT Press, Cambridge, MA.

Knuth, D. E. (1971), 'Optimum binary search trees', *Acta Informatica* **1**(1), 14–25.

Mehlhorn, K. (1984), *Data Structures and Algorithms 1: Sorting and Searching*, Vol. 1 of *Monographs in Theoretical Computer Science. An EATCS Series*, Springer.

Moshkov, M. & Chikalov, I. (2000), 'On algorithm for constructing of decision trees with minimal depth', *Fundam. Inf.* **41**(3), 295–299.

Moshkov, M. & Chikalov, I. (2003), 'Consecutive optimization of decision trees concerning various complexity measures', *Fundam. Inf.* **61**(2), 87–96.