7
Preliminaries

Every query strategy for a tree T can be represented by a binary decision tree D such that a path of D indicates what queries should be made at each step. Then D is a binary tree, where each internal node corresponds to a query for a different arc of T and each leaf of D corresponds to a different node of T (these correspondences shall become clear later). In addition, each internal node u of D satisfies a *search property* that can be described as follows. If u corresponds to a query for the arc (i, j), then: (i) all nodes of the right subtree of u correspond to either a query for an arc in T_j or to a node in T_j; (ii) all nodes of the left subtree of u correspond to either a query for an arc in $T - T_j$ or to a node in $T - T_j$.

For any decision tree D, we use $u_{(i,j)}$ to denote the internal node of D which corresponds to a query for the arc (i, j) of T and u_v to denote the leaf of D which corresponds to the node v of T.

From the example presented in the introduction, we can infer an important property of the decision trees. Consider a tree T and a search strategy given by a decision tree D for T. If v is the marked node of T, the number of queries posed to find the marked node is the distance (in arcs) from the root of D to u_v.

For any decision tree D, we define $d(u, v, D)$ as the distance between nodes u and v in D (when the decision tree is clear from the context, we omit the last parameter of this function). Thus, the expected (with respect to w) number of queries it takes to find the marked node using the strategy given by the decision tree D, or simply the cost of D, is given by:

$$\text{cost}(D, w) = \sum_{v \in T} d(r(D), u_v, D)w(v)$$

Therefore, the problem of computing a search strategy for (T, w) which minimizes the expected number of queries can be recast as the problem of finding a decision tree for T with minimum cost, that is, that minimizes $\text{cost}(D, w)$ among all decision trees D for T. The cost of such minimum cost decision tree is denoted by $\mathrm{OPT}(T, w)$.

Now we present properties of decision trees which are crucial for the analysis of the proposed algorithm. Consider a subtree T' of T; we say that a node u is a *representative* of T' in a decision tree D if the following conditions hold: (i) u is a node of D that corresponds to either an arc or a node of T' (ii) u is an ancestor of all other nodes of D which correspond to arcs or nodes of T'. The next lemma asserts the existence of a representative for each subtree of T.

Lemma 12 *Consider a tree T and a decision tree D for T. For each subtree T' of T, there is a unique node $u \in D$ which is the representative of T' in D.*

We denote the representative of T' (with respect to some decision tree) by $u(T')$.

The second property is given by the following lemma:

Lemma 13 *Consider a tree T, a weight function w and a decision tree D for T. Then for every subtree T' of T, $\sum_{v \in T'} d(u(T'), u_v, D)w(v) \geq OPT(T', w)$.*

The idea of the proof is to construct a decision tree D' for T' based on D in the following way: the nodes of D' are the nodes of D which correspond to the arcs and nodes of T'; there is an arc from u to v in D' iff u is the closest ancestor of v in D, among the nodes of D' (Figure 7.1).

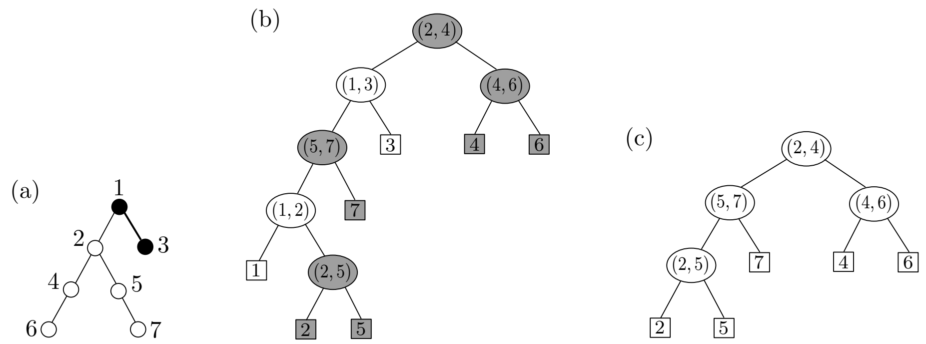

Figure 7.1: (a) Tree T. (b) A decision tree D for T, with nodes corresponding to nodes and arcs of T_2 in gray. (c) Decision tree D' for T_2 constructed by connecting the nodes of D corresponding to nodes and arcs of T_2.

By construction, the distance between two nodes u and v in D' is not greater than their distance in D. In addition, $u(T')$ is the root of D', so we have:

$$\mathrm{cost}(D', w) = \sum_{v \in T'} d(u(T'), u_v, D')w(v) \leq \sum_{v \in T'} d(u(T'), u_v, D)w(v)$$

As one can prove that D' is a valid decision tree for T', we have that $\mathrm{OPT}(T', w) \leq \mathrm{cost}(D', w)$ and consequently $\sum_{v \in T'} d(u(T'), u_v, D)w(v) \geq \mathrm{OPT}(T', w)$.