International Journal of Database Theory and Application
Vol.9, No.7 (2016), pp.179-194
http://dx.doi.org/10.14257/ijdta.2016.9.7.16

# A Round Trip Pattern for Building Decision Trees

Liang Jia[1] and Liuhong Yan[2 *]

*[1]School of Information Science & Engineering, Chang Zhou University, China*
*[2]Zhou Youguang School of Languages and Cultures, Chang Zhou University, China*
*[*] E-mail: Sanctifier@vip.qq.com*

***Abstract***

*This paper introduces a recursive procedure named round trip pattern for generating decision trees based on alterable decision tables. The pattern is mathematically described by two components representing building tree based on table and altering table based on data from application adapting tree respectively. This cyclic pattern starts at creating a decision tree from initial table altered according to Bayes theorem combined with Pareto values of objective functions. The functions reflect time and spacial complexities of decision trees. After the initial tree is implemented by application, relevant system data is collected and analyzed to alter table for building new tree. This cycle continues until system performance is satisfactory. A small business project is designed and maintained based on round trip pattern. The system statues are retrieved from its database and analyzed in details. The analyses of data indicate system performance stays satisfactorily stable.*

***Keywords****: decision tree; Bayes theorem; Pareto frontier*

## 1. Introduction

As an efficient and viable means to represent decision logic embedded in decision table, decision tree is critical for multifarious applications involving automatic decision makings, *e.g.*, digital image processing[1-2], biomedicine [3], firm power capacity scheduling [4], stock analysis [5], earnings management[6], data mining [7], *etc.* The structure of decision tree is crucial for efficiency of system depending on it. How to design algorithms for constructing decision trees of minimal complexities is the focus shared by many researches [8-13]. However generating decision tree of all minimized complexities is commonly intangible and impracticable for real-world applications. Pursuit of all minimized complexities usually ends with compromising minimization of some but not all complexities due to the requirements of application. Most literatures about generating decision tree are focusing the generating procedure depending on information provided by decision table, *i.e.*, a procedure involving a trip or investigation of decision table once for all. Few noticed that structure of decision tree which finally determines complexities may be shaped in some degree based on round trips between generating procedure and table alteration, *i.e.*, a trip from table to tree to initially generate tree, then a trip from tree to table to alter table based on information provided by application implementing tree, then generating tree again based on altered table and *etc.* The round trips can be as many as desired until the structure or complexities of generated tree meet application requirements. The basic idea of round trip is that application requirements should dominate the tree generating procedure in case that minimization of all complexities is intangible.

This paper proposes mathematical descriptions and corresponding algorithms for round trip pattern of constructing decision tree based on decision tables altered either by data retrieved from application adapting former-generated decision tree or by Bayes theorem involving Pareto optimal values of objective functions. The objective functions are designed to reflect both time and spacial complexities. Round trip pattern comprises table-to-tree trip and tree-to-table trip. The initial table-to-tree trip yields decision trees built on table altered by Bayes theorem on account of no application data is available. Then application is implemented based on tree built in initial trip. After a period of running application, statistical data is calculated based on application database. The data is then employed for altering table and new tree is generated. Application is thus updated based on new tree. This cyclic procedure continues until application performance is satisfactorily stable. A small business project adapting round trip pattern is developed and maintained. The statistical data computed from its database show its performance is improved and stays reasonably stable after several months of running application.

The rest of paper is organized as followings. Section 2 provides a brief review about Pareto set and definitions associated of decision tree. Section 3 introduces round trip pattern and its two components mathematically in details. Section 4 focuses on algorithms implementing round trip pattern. Section 5 analyzes the statistical data calculated from real-world application database and shows application performance stays stable. Section 6 draws conclusions.

## 2. Related Works and Basic Concepts

Algorithms of generating decision trees based on given decision tables are constantly evolving. There are plenty of algorithms for constructing decision trees with respect to terminal vertices and height of the tree. Different strategies are employed for decision tree generating of different kinds of decision tables, *e.g.*, single-valued decision tables [8-11] and multi-valued tables [12-13]. Construction algorithms roughly match some design pattern like dynamic programming [13], incremental algorithm [10] and greedy algorithm [11-12], *etc.* Normally, proposed algorithms attempt to construct decision trees of minimal height and minimal leaf number which are referred as time and spacial complexities. Commonly, such attempts end with compromise of optimizing one complexity because minimization of two is not always possible. This essentially resembles the case of multiobjective optimization in which several objective functions need to be minimized, but the ideal case of minimizing all objective functions simultaneously is not possible. Hence, suboptimal solutions which minimize objective functions in some degree are derived as Pareto set. Unlike typical multiobjective optimization whose objective space can be explicitly explored [14-15], objective space for complexities of decision trees can only be implicitly reflected by relationships of subtables and associated objective functions due to the great number of possible trees. Procedure of generating decision tree is commonly a one-way trip, *i.e.*, a tree is built on a given table and the table will usually never be visited again. Information required for building decision tree is retrieved from table once for all; nevertheless, the structure of decision tree will be quite different if underlying table is altered without loss of logic originally embedded. In data mining, decision table is employed as training data set which is altered with permission of reducing noisy data, like in [7]. This kind of alteration causes information lost and logic changings which are undesirable for retaining original logic of initial table, and more important, few mentioned round trip pattern which alters table based on generated tree and create new tree in turn.

Formally, $\mathrm{k}$ constrains of multiobjective optimization are formalized as functions $g_1, g_2, \dots, g_k$, there are $n$ objective functions $F_1, F_2, \dots, F_n$ and their values form objective space $\mathcal{Z} \subseteq \mathbb{R}^n$. Vectors compatible with $F_i$ where $i = 1,2, \dots, n$ form decision space $\mathcal{D} \subseteq \mathbb{R}^m$ and the set of vectors in $\mathcal{D}$ satisfying $g_j$ where $j = 1,2, \dots, k$ is called feasible space and denoted by $\mathcal{D}^* \subseteq \mathcal{D}$. Hence, there is a map $\mathcal{F}: \mathbb{R}^n \to \mathbb{R}^m$

from $\mathcal{D}^*$ to a subset $\mathcal{Z}^* \subseteq \mathcal{Z}$, *i.e.*, image of $\mathcal{D}^*$ under $\mathcal{F}$. For a given vector $\mathbb{x} \in \mathcal{D}^*$, the definition of multiobjective optimization is given below.

$$\min_{\mathbb{x}\in\mathcal{D}^*} \mathcal{F}(\mathbb{x}) = \min_{\mathbb{x}\in\mathcal{D}^*}(F_1(\mathbb{x}), F_2(\mathbb{x}), \dots, F_n(\mathbb{x}))$$

$\mathbb{x}^* \in \mathcal{D}^*$ is Pareto optimal for $\min_{\mathbb{x}\in\mathcal{D}^*}\mathcal{F}(\mathbb{x})$ iff $\nexists \mathbb{x}'$ such that $F_i(\mathbb{x}') \leq F_i(\mathbb{x}^*)$ for $i = 1,2,\dots,n$ and $\exists j$, $1 \leq j \leq n$, $F_j(\mathbb{x}') < F_j(\mathbb{x}^*)$. Minimum of $F_i$ in $\mathcal{Z}$ is denoted by $F_i^*$, a common assumption of $\min_{\mathbb{x}\in\mathcal{D}^*}\mathcal{F}(\mathbb{x})$ is that there is no $\mathbb{x}' \in \mathcal{D}^*$ satisfying $(F_1(\mathbb{x}'), F_2(\mathbb{x}'), \dots, F_n(\mathbb{x}')) = (F_1^*, F_2^*, \dots, F_n^*,)$. To understand how Pareto set is related with decision trees, relative definitions of decision tree have to be introduced.

Let $\{\cdot\}_{\neq}$ denotes a set containing distinct elements, $\{\cdot\}$ denotes a set of no requirement about distinctness of elements. $|\{\cdot\}|_{\neq}$ and $|\{\cdot\}|$ respectively denote number of elements in $\{\cdot\}$ with and without honoring distinctness. A decision tree is a concise representation of decision table. A decision table $T$ is a two dimensional table of condition attributes $\{f_1, f_2, \dots, f_m\}_{\neq} = \mathrm{E}(T)$ which graphically are column headers of $T$. Row of index $i$ in $T$ contains an array of values as $\{c_{i1}, c_{i2}, \dots, c_{im}\}$ corresponding to $\mathrm{E}(T)$ and these values lead to a decision value $d_i$. Values of $n$ rows in $T$ form a set $\{\{c_{11}, c_{12}, \dots, c_{1m}\}, \{c_{21}, c_{22}, \dots, c_{2m}\} \dots \{c_{n1}, c_{n2}, \dots, c_{nm}\}\} = \mathrm{C}(T)$ and the matched $n$ decisions form a set $\{d_1, d_2, \dots, d_n\} = \mathrm{D}(T)$ ($|\mathrm{D}(T)|_{\neq} = n$ when $d_i \neq d_j$, $i \neq j$; $|\mathrm{D}(T)|_{\neq} < n$, otherwise). Generally, a decision table $T$ has the following structure.

$$T = \begin{bmatrix} f_1 & \cdots & f_m & d \\ c_{11} & \cdots & c_{1m} & d_1 \\ \vdots & \ddots & \vdots & \vdots \\ c_{n1} & \cdots & c_{nm} & d_n \end{bmatrix}$$

For any $d_k \in \mathrm{D}(T)$, $k = 1,2 \dots n$, there is a $\{c_{k1}, c_{k2} \dots c_{km}\} = r_k \in \mathrm{C}(T)$. For two $r_{k_1}, r_{k_2} \in \mathrm{C}(T)$ and $k_1 \neq k_2$, inequalities $c_{k_11} \neq c_{k_21}, c_{k_12} \neq c_{k_22} \dots c_{k_1m} \neq c_{k_2m}$ hold for $r_{k_1}, r_{k_2}$. For a given subset of $\mathrm{E}(T)$, *i.e.*, $f_{i_1}, f_{i_2}, \dots, f_{i_t}$ and their values $a_1, a_2, \dots, a_t$, a subtable $\Theta$ can be constructed from $T$. $\Theta$ has same condition attributes as $T$, but its rows is selectively chosen from $T$, *i.e.*, $\mathrm{E}(\Theta) = \mathrm{E}(T)$ and for any $r_j = \{c_{j1}, c_{j2} \dots c_{jm}\} \in \mathrm{C}(\Theta)$ where $j = 1,2 \dots |\mathrm{C}(\Theta)|$, $c_{ji_1} = a_1, c_{ji_2} = a_2 \dots c_{ji_t} = a_t$ hold. $\Theta$ is also denoted by $T(f_{i_1}, a_1)(f_{i_2}, a_2) \dots (f_{i_t}, a_t)$.

For any vertex $v$ of a decision tree $\Gamma$ generated based on a given decision table $T$, $v$ denotes one condition attribute from $\mathrm{E}(T)$, *i.e.*, $v \in \mathrm{E}(T)$. The edge connecting just two vertices is marked by condition values from $\mathrm{C}(T)$. Notice, any $f_i \in \mathrm{E}(T)$ where $i = 1, \dots, |\mathrm{E}(T)|$ can be root vertex of $\Gamma$ under the assumption $c_{ji} \in r_j \in \mathrm{C}(T)$ where $j = 1, \dots, |\mathrm{C}(T)|$ is processed despite of order of $f_1, f_2, \dots, f_{|\mathrm{E}(T)|}$, and ending vertex of $\Gamma$ can only be decision values from $\mathrm{D}(T)$. A path of $t$ steps from root vertex $f_i$ to arbitrary $v \in \mathrm{E}(T) \cup \mathrm{D}(T)$ begins from $f_i$, passing vertices $f_{i_1}, f_{i_2}, \dots, f_{i_t}$ through their connected edges marked by values $a_1, a_2, \dots, a_t$ and ends at $v$. According the vertices and values associated with $t$-step path, a subtable $T(v)$ can be constructed as following.

$$T(v) = \begin{cases} T & v = f_i \\ T(f_{i_1}, a_1)(f_{i_2}, a_2) \dots (f_{i_t}, a_t) & v \neq f_i \end{cases}$$

For an arbitrary row $r_k \in \mathrm{C}(T)$, $k = 1,2 \ldots |\mathrm{C}(T)|$, $r_k$ is determined by some decision value $d_k \in \mathrm{D}(T)$. In a decision tree $\Gamma_{T(d_k)}$ built on $T(d_k)$, there is a path presents $r_k \cup \{d_k\}$, the length of $r_k$, $\ell(r_k)$, is defined by the sum of values marked on edges of path from root of $T(d_k)$ to $d_k$. If $r_k = \{c_{k1}, c_{k2} \ldots c_{km}\}$, then $\ell(r_k) \leq \sum_{j=1}^{m} c_{kj}$. For a decision tree $\Gamma_{T(v)}$ built on $T(v)$, the total length of $T(v)$ is defined as $\mathcal{L}(\Gamma_{T(v)}) = \sum_{k=1}^{n} \ell(r_k)$. The time complexity of $\Gamma_{T(v)}$ is defined as followings.

$$\mathcal{H}(\Gamma_{T(v)}) = \mathcal{L}(\Gamma_{T(v)}) / |\mathrm{C}(T(v))|$$

Spacial complexity of $\Gamma_{T(v)}$ is defined as number of terminal vertices of $\Gamma_{T(v)}$, $\mathcal{T}(\Gamma_{T(v)})$. The famous Bayes theorem for decision table $T$ of $d_i \in D(T)$ where $i = 1, \ldots, |\mathrm{D}(T)|_{\neq}$ and $r_j \in C(T)$ where $j = 1, \ldots, |C(T)|$ is defined as $\mathrm{P}(d_i|r_j) = \mathrm{P}(r_j|d_i)\mathrm{P}(d_i)/\mathrm{P}(r_j)$ with the assumption probabilities of $r_{j_1}$ attached to $d_{i_1}$ and $r_{j_2}$ attached to $d_{i_2}$ where $i_1 \neq i_2$ and $j_1 \neq j_2$ are independent. Probabilities involved in $\mathrm{P}(d_i|r_j)$ can be computed differently as required, *e.g.*, $\mathrm{P}(d_i)$ can be assumed as $1/|\mathrm{D}(T)|$, $1/|\mathrm{D}(T)|_{\neq}$ or $|\mathrm{C}(d_i)|/|\mathrm{C}(T)| = |\mathrm{C}(d_i)|/|\mathrm{D}(T)|$ where $\mathrm{C}(d_i)$ denotes set of $r_j$ attached by $d_i$, and $\mathrm{P}(r_j|d_i) = \prod_{k=1}^{|\mathrm{E}(T)|} \mathrm{P}(c_{jk}|d_i)$. Since $\mathrm{P}(r_j)$ is always assumed to be constant, $\mathrm{P}(d_i|r_j) = \mathrm{P}(r_j|d_i)\mathrm{P}(d_i)/\mathrm{P}(r_j) \propto \mathrm{P}(r_j|d_i)\mathrm{P}(d_i)$, *i.e.*, we only need to compute $\mathrm{P}(r_j|d_i)\mathrm{P}(d_i)$ when comparing probabilities of $r_j$ and $d_i$ when $j$ is fixed. Based on these definitions, mathematical descriptions for generating Pareto optimal point and altering decision table are given in following sections.

## 3. Round Trip Pattern

The general scheme of round trip pattern is shown in Figure1. There are four steps in scheme, *i.e.*, Step 1: calculate Pareto optimal values, Step 2: alter decision table, Step 3: construct decision tree and Step 4: apply decision tree. Step 1 to Step 3 is mathematically introduced in this section. Step 4, however, cannot be precisely described because of the diversity of applications implementing decision trees and corresponding multifarious means of retrieving and analyzing application data. But Step 4 is mandatory for completing round trip and hence is conceptually introduced. Step 1 computes Pareto values of objective functions explored in Subsection 3.1. Values computed in Step 1 are adapted in Step 2 for altering initial table. Then the flow starts with Step 2, continues to Step 3, then Step 4 and returns to Step 2 which completes a round trip. If the termination condition in Step 2 is not met, then another trip starts. This cyclic procedure is described mathematically in Subsection 3.2. Step 3 is introduced in Subsection 3.3.

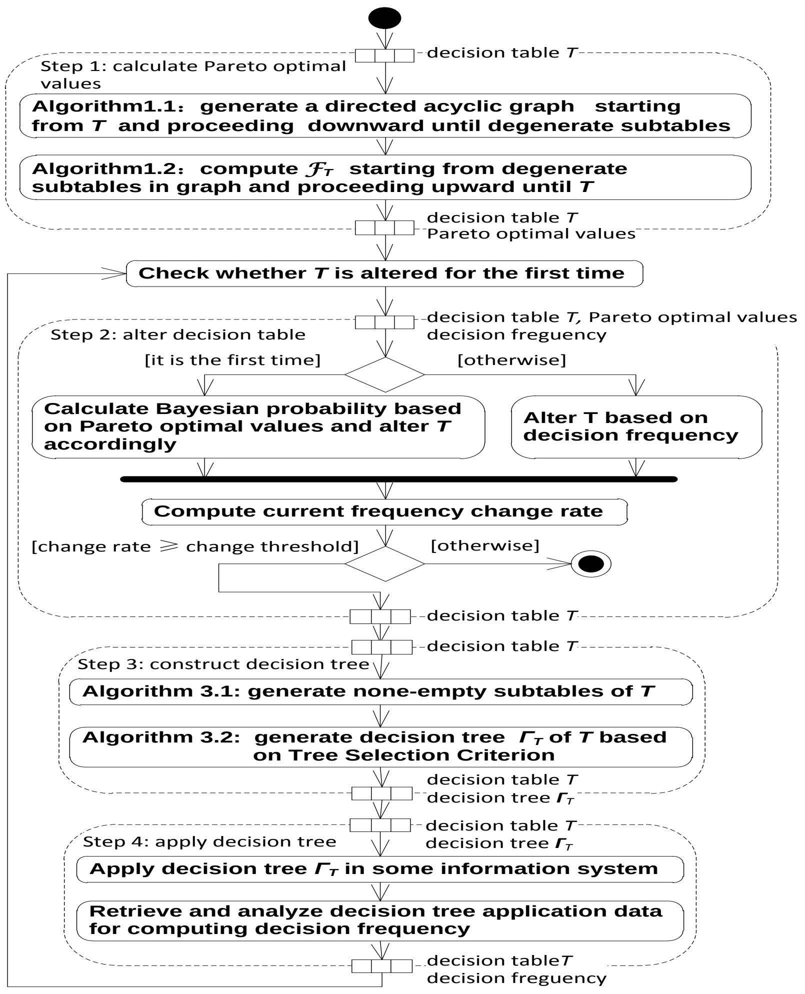
decision table $T$
Step 1: calculate Pareto optimal values
Algorithm1.1: generate a directed acyclic graph starting from $T$ and proceeding downward until degenerate subtables
Algorithm1.2: compute $\mathcal{F}_T$ starting from degenerate subtables in graph and proceeding upward until $T$
decision table $T$
Pareto optimal values
Check whether $T$ is altered for the first time
decision table $T$, Pareto optimal values
decision freguency
Step 2: alter decision table
[it is the first time]
[otherwise]
Calculate Bayesian probability based on Pareto optimal values and alter $T$ accordingly
Alter T based on decision frequency
Compute current frequency change rate
[change rate ⩾ change threshold]
[otherwise]
decision table $T$
decision table $T$
Step 3: construct decision tree
Algorithm 3.1: generate none-empty subtables of $T$
Algorithm 3.2: generate decision tree $\Gamma_T$ of $T$ based on Tree Selection Criterion
decision table $T$
decision tree $\Gamma_T$
decision table $T$
decision tree $\Gamma_T$
Step 4: apply decision tree
Apply decision tree $\Gamma_T$ in some information system
Retrieve and analyze decision tree application data for computing decision frequency
decision table$T$
decision freguency

**Figure 1. General Scheme of Round Trip Pattern**

### 3.1. Pareto Optimal Value of Objective Function

This section introduces the concepts and definitions about Pareto optimal values of objective functions reflected by Step 1 in Figure 1. Pareto optimal values are derived only for alteration about initial decision table and will not be touched by subsequent trips. For a decision table $T'$, if all condition values associated with $f_i \in \mathrm{E}(T')$ in $\mathrm{C}(T')$ is denoted by $\mathrm{C}(T',f_i)$, *i.e.*, all values in column $f_i$ of $T'$ where $i \in \mathbb{Z}^+$, $1\leq i \leq |\mathrm{E}(T')|$. If $\beta = \min_{\Gamma_{T'} \in \mathfrak{T}(T')} \mathcal{L}(\Gamma_{T'})$, $\alpha = \min_{\Gamma_{T'} \in \mathfrak{T}(T')} \mathcal{T}(\Gamma_{T'})$, $R_{max} = \max_{r_j} \sum_{k_1=1}^{|\mathrm{E}(T')|} c_{jk_1}$, $k_1 \in \mathbb{Z}^+$, then $\mathcal{R}_{\mathcal{L},T'} = \{\beta\ ,\ \beta + 1 \dots |\mathrm{C}(T')| \cdot R_{max}\}$, $\mathcal{R}_{\mathcal{T},T'} = \{\alpha, \alpha + 1 \dots |\mathrm{C}(T')|\}$ respectively denote ranges of $\mathcal{L}$ and $\mathcal{T}$. Map $\mathcal{F}_{T'}: \mathcal{T} \rightarrow \mathcal{L}$ is defined as followings.

$$\mathcal{F}_{T'}(t') = \left\{\min_{\Gamma_{T'}} \mathcal{L}(T', \Gamma_{T'}) \in \mathcal{R}_{\mathcal{L},T} \,\middle|\, \Gamma_{T'} \in \mathfrak{T}(T') \text{ and } \mathcal{T}(T', \Gamma_{T'}) \leq t'\right\}$$

Where $t' \in \mathcal{R}_{\mathcal{T},T'}$. Let data structure graph be employed to represent relationships between $T'$ and its subtables, then the nodes of graph are subtables $T'(v) = T'_v$, edges emitting from a node associated with $f_i \in \mathrm{E}(T'_v)$ are marked by pairs of values $(f_i, a_k)$, $a_k \in \mathrm{C}(T', f_i)$, $k \in \mathbb{Z}^+$, $k = 1, \ldots, |\mathrm{C}(T'_v, f_i)|$ and pointing to nodes $T'_v(f_i, a_k)$. Hence, edges in the graph are directed and connect two tables of parent-child relationship. The graph is thus a directed acyclic graph ($\mathrm{DAG}$). For a node $T'_v \in \mathrm{DAG}$, it has two possible types determined by number of decision values.

When $|\mathrm{D}(T'_v)|_{\neq} = 1$ for all rows $r_j \in \mathrm{C}(T'_v)$, $r_j$ share a common decision value and $T'_v$ is leaf node of $\mathrm{DAG}$ of $\mathcal{F}_{T'_v}(1) = 0$.

When $|\mathrm{D}(T'_v)|_{\neq} > 1$, $r_j \in \mathrm{C}(T'_v)$ can be categorized based on their different decision values. Starting from node $T'_v$, for any $f_i \in \mathrm{E}(T'_v)$ and $a_1, a_2, \ldots, a_{|\mathrm{C}(T'_v, f_i)|} \in \mathrm{C}(T'_v, f_i)$, there are $|\mathrm{C}(T'_v, f_i)|$ edges marked by $(f_i, a_1)$, $(f_i, a_2)$, $\ldots$, $\left(f_i, a_{|\mathrm{C}(T'_v, f_i)|}\right)$ and leading to nodes $T'_v(f_i, a_1)$, $T'_v(f_i, a_2), \ldots, T'_v\left(f_i, a_{|\mathrm{C}(T'_v, f_i)|}\right)$. For each $T'_v(f_i, a_k)$, there is a corresponding $\mathcal{R}_{\mathcal{T}, T'_v(f_i, a_k)}$. For $f_i \in \mathrm{E}(T_v)$, an ordered set of all possible compositions of values of $\mathcal{T}\left(\Gamma_{T'_v(f_i, a_k)}\right)$ is defined as followings.

$$\Re_{\mathcal{T}, T'_v, f_i} = \left\{ \mathbb{v}_i * \mathbb{v}, \mathbb{v}_{i+1} * \mathbb{v}, \ldots, \mathbb{v}_{I + |\mathrm{C}(T'_v)|} * \mathbb{v} \right\}$$

Where $\mathbb{v} = [1 \quad \ldots \quad 1]$, $*$ denotes inner product and $\mathbb{v}_{k'} * \mathbb{v} < \mathbb{v}_{k'+1} * \mathbb{v}$, $k' = i, i+1, \ldots, i + |\mathrm{C}(T_v)| - 1$. Let $k_{\max} = |\mathrm{C}(T'_v, f_i)|$, $\mathbb{v}_i$ is defined as followings.

$$\mathbb{v}_i = \begin{bmatrix} \min\limits_{\Gamma_1 \in \mathfrak{T}\left(T'_v(f_i, a_1)\right)} \mathcal{T}(\Gamma_1) \\ \min\limits_{\Gamma_2 \in \mathfrak{T}\left(T'_v(f_i, a_2)\right)} \mathcal{T}(\Gamma_2) \\ \vdots \\ \min\limits_{\Gamma_{k_{max}} \in \mathfrak{T}\left(T'_v(f_i, a_{k_{\max}})\right)} \mathcal{T}\left(\Gamma_{k_{max}}\right) \end{bmatrix}$$

For $\mathbb{v}_i \neq \mathbb{v}_{k''}$, $i + 1 \leq k'' \leq |\mathrm{C}(T_v)|$, $k'' \in \mathbb{Z}^+$, $\mathbb{v}_{k''}$ is defined as followings.

$$\mathbb{v}_{k''} = \begin{bmatrix} \mathcal{T}\left(\Gamma_{T'_v(f_i, a_1)}\right) \\ \mathcal{T}\left(\Gamma_{T'_v(f_i, a_2)}\right) \\ \vdots \\ \mathcal{T}\left(\Gamma_{T'_v(f_i, a_{k_{\max}})}\right) \end{bmatrix}$$

For $T'_v(f_i, a_k)$, value of $\mathcal{F}_{T'_v(f_i, a_k)}$ can be derived. For $\mathbb{v}_{k'} * \mathbb{v} \in \Re_{\mathcal{T}, T'_v, f_i}$, objective function $\mathcal{F}^{f_i}_{T'_v}$ associated with complexities of decision trees generated based on $f_i \in \mathrm{E}(T'_v)$ is defined as followings.

$$\mathcal{F}^{f_i}_{T'_v}\left(\Re_{\mathcal{T}, T'_v, f_i}\right) = \min_{\mathbb{v}_{k'} * \mathbb{v} \in \Re_{\mathcal{T}, T'_v, f_i}} \sum_{k=1}^{|\mathrm{C}(T'_v, f_i)|} \mathcal{F}_{T'_v(f_i, a_k)}\left(\mathbb{v}_{k'}(k)\right) + |\mathrm{C}(T'_v, f_i)|$$

Where $\mathbb{v}_{k'}(k)$ denotes the $k$th component of vector $\mathbb{v}_{k'}$. According to $\mathcal{F}_{T_v'}^{f_i}(\mathbb{v}_k)$, objective function $\mathcal{F}_{T_v'}(\mathfrak{R}_{\mathcal{T},T_v'})$ of $\mathrm{E}(T_v')$ for describing complexities of subtable $T_v'$ is given as followings.

$$\mathcal{F}_{T_v'}(\mathfrak{R}_{\mathcal{T},T_v'}) = \left\{ \mathcal{F}_{T_v'}^{f_i} \middle| \min_{\mathfrak{R}_{\mathcal{T},T_v',f_i} \in \mathfrak{R}_{\mathcal{T},T_v'}} \mathcal{F}_{T_v'}^{f_i}(\mathfrak{R}_{\mathcal{T},T_v',f_i}) \right\}$$

$$\mathfrak{R}_{\mathcal{T},T_v'} = \left\{ \mathfrak{R}_{\mathcal{T},T_v',f_1}, \mathfrak{R}_{\mathcal{T},T_v',f_2}, \dots, \mathfrak{R}_{\mathcal{T},T_v',f_{|\mathrm{E}(T_v')|}} \right\}$$

Generally, if $T_v'$ is terminal node of graph, then $\mathcal{F}_{T_v'}(1) = 0$ and Pareto optimal point is $\left(1, \mathcal{F}_{T_v'}(1)\right)$; if $T_v'$ is not a terminal node, we first compute $\mathcal{F}_{T_v}^{f_i}(\mathfrak{R}_{\mathcal{T},T_v',f_i})$ for each $f_i \in E(T_v')$, then choose the minimal value as $\mathcal{F}_{T_v'}(\mathfrak{R}_{\mathcal{T},T_v'})$ and Pareto optimal point is $\left(\mathfrak{R}_{\mathcal{T},T_v'}^{-1}, \mathcal{F}_{T_v}(\mathfrak{R}_{\mathcal{T},T_v'})\right)$ where $\mathfrak{R}_{\mathcal{T},T_v'}^{-1}$ is objective function for retrieving $\mathbb{v}_k * \mathbb{v} \in \mathfrak{R}_{\mathcal{T},T_v',f_i}$ which yields $\mathcal{F}_{T_v'}$. For multiobjective optimization of decision tree complexities, objective space is a two-dimensional space whose objective functions are $\mathfrak{R}_{\mathcal{T},T_v'}^{-1}$ and $\mathcal{F}_{T_v'}(\mathfrak{R}_{\mathcal{T},T_v'})$ which graphically are two axis of space.

### 3.2 Decision Table Alteration

This section introduces concepts and definitions about altering decision tables according to either Bayesian probability based on Pareto values yielded by Step 1 or decision frequency given by Step 4. This alteration is reflected in Step 2. Hence, table alteration made in Step 2 involves two different phrases: Phrase 1: before round trip starts and Phrase 2: during the activated round trip.

For Phrase 1, Bayesian probability is computed for altering initial table which is represented by left branch in Step 2 of Figure 1. If $\mathrm{P}(d_m) = |\mathrm{C}(d_m)|/|\mathrm{C}(T')|$, $\mathcal{F}_T^{\max} = \max_{f_i} \mathcal{F}_T^{f_i}$ and $\mathcal{F}_T^{f_i}/\mathcal{F}_T^{\max} = 1$ when $\mathcal{F}_T^{\max} = 0$, $k_2 \in \mathbb{Z}^+$, definition of Bayesian probability involving $r_j$ is given below.

$$\mathrm{P}(r_j|d_m) = \prod_{k_2=1}^{|\mathrm{E}(T')|} \left( \frac{\sum_{r_{j'} \in \mathrm{C}(d_m)} |r_{j'}(c_{jk_2})|}{|\mathrm{C}(T')| \cdot |\mathrm{E}(T')|} \right)^{\mathcal{F}_T^{f_{k_2}}/\mathcal{F}_T^{\max}}$$

For Phrase 2, decision frequency given by Step 4 is computed for altering non-initial table which is represented by right branch in Step 2. In Step 4, after the application implementing decision tree was running for a period, data of automatically-made decisions is recorded and retrieved for computing decision frequency corresponding to each row in decision table. For $r_j$, decision frequency can be defined as ratio of decision number associated with $r_j$ and number of all decisions made during running period.

Based on the above introduction, we can mathematically describe table alteration in Step 2 as followings. If $T_1$ denotes the first table obtained by altering $T_0$ based on Bayesian probability, $T_2$ denotes the second table obtained by altering $T_1$ based on decision frequency, … , $T_{t+1}$ denotes the $(t+1)$th table obtained by altering $T_t$ based on decision frequency and $r_j^{t+1} \in \mathrm{C}(T_{t+1})$, $t \in \mathbb{Z}^+ \cup \{0\}$ denotes the row of index $j$ in table $T_{t+1}$, then table alteration operation $\mathfrak{L}_\wedge$ is defined as followings.

$$T_{t+1} = \mathfrak{L}_\wedge(T_t) = \begin{bmatrix} f_1 & \cdots & f_{|\mathrm{E}(T_t)|} & \\ c_{11} & \cdots & c_{1|\mathrm{E}(T_t)|} & \mathfrak{L}(r_1^t) \\ \vdots & \ddots & \vdots & \vdots \\ c_{|\mathrm{C}(T_t)|1} & \cdots & c_{|\mathrm{C}(T_t)|\cdot|\mathrm{E}(T_t)|} & \mathfrak{L}(r_{|\mathrm{C}(T_t)|}^t) \end{bmatrix}$$

If $\mathrm{P}(r_j^t)$ denotes decision frequency of $r_j^t$ computed from application data, definition of $\mathfrak{L}(r_j^t)$ is given below.

$$\mathfrak{L}(r_j^t) = \begin{cases} \left(d_j, \mathrm{P}(r_j^0|d_j)\mathrm{P}(d_j)\right) & t = 0 \\ \left(d_j, \mathrm{P}(r_j^t)\right) & t > 0 \end{cases}$$

Let $\mathfrak{L}_1(r_j^t) = d_j$, $\mathfrak{L}_2(r_j^t) = \mathrm{P}(r_j^0|d_j)\mathrm{P}(d_j)$ when $t = 0$, $\mathfrak{L}_2(r_j^t) = \mathrm{P}(r_j^t)$ when $t > 0$。 After table is altered, its decision tree can be constructed by algorithms reflected in Step 3. Specifically, for $T_{t+1}$ from the $(t+1)$th table alteration, decision tree $\Gamma_{T_{t+1}}$ of $T_{t+1}$ is constructed by algorithm $\mathcal{A}_{\mathfrak{S}}$ with respect to Tree Selection Criterion denoted by $\mathfrak{S}$, *i.e.*, $\Gamma_{T_{t+1}} = \mathcal{A}_{\mathfrak{S}}(T_{t+1})$. Then $\Gamma_{T_{t+1}}$ is implemented in application and relevant data is collected during application life time in Step 4. Decision frequency is computed from application data and round trip returns to Step 2. Round trip continues until frequency change rate for current trip does not exceed a fixed threshold called change threshold and denoted by $\mathscr{t}_\wedge \in \mathbb{R}^+$, $0 \le \mathscr{t}_\wedge \le 1$. Change rate $\mathscr{v}_{t+1}$ is defined as followings.

$$\mathscr{v}_{t+1} = \begin{cases} \mathscr{t}_\wedge & t = 0 \\ \dfrac{\left|\left\{r_j^{t+1} \in \mathrm{C}(T_{t+1}) \,\middle|\, \left|\mathfrak{L}_2(r_j^t) - \mathfrak{L}_2(r_j^{t-1})\right| \ge \mathscr{t}_r\right\}\right|}{|\mathrm{C}(T_{t+1})|} & t > 0 \end{cases}$$

Where $\mathscr{t}_r \in \mathbb{R}^+$, $0 \le \mathscr{t}_r \le 1$ is another given threshold for rows. Once $\mathscr{v}_{t+1}$ is found smaller than $\mathscr{t}_\wedge$ which means there is no need to continue round trip based on given parameters $\mathscr{t}_r$ and $\mathscr{t}_\wedge$.

## 3.3. Generating Decision Tree

This section introduces concepts and definitions about Tree Selection Criterion $\mathfrak{S}$ and $\mathcal{A}_{\mathfrak{S}}$ reflected by Step 3 in Figure 1. $\mathcal{A}_{\mathfrak{S}}$ generates a decision tree based on decision table altered in Step 2. How decision tree is generated is determined by $\mathfrak{S}$ whose definition is given below.

$$\mathfrak{S} = \sum_{j=1}^{|\mathrm{D}(T_t)|} \ell(r_j^{t+1}) \cdot \left(1 - \mathfrak{L}_2(r_j^t)\right)$$

Assuming the procedure of running $\mathcal{A}_{\mathfrak{S}}$ can be conceptually divided by generating trees for subtables of $T_t$, *i.e.*, each phrase of $\mathcal{A}_{\mathfrak{S}}$ generates exactly one decision tree and current phrase depends on some or all trees generated in previous phrases. Notice generating single-vertex trees is not a phrase because it does not depend on any previously-generated trees.

Assuming the $t'$th phrase $\mathcal{A}_{\mathfrak{S},t'}$ just completed and the $(t'+1)$th phrase $\mathcal{A}_{\mathfrak{S},t'+1}$ is about to begin, then for subtable $T_t$ targeted at current phrase, we first categorize its subtables $T_t(f_i, a_k) = T_{t,\sqsubseteq}$ based on their decision values set $\{\mathrm{D}(T_{t,\sqsubseteq})\}_{\neq} = \mathrm{D}_{\neq}(T_{t,\sqsubseteq})$. If $D_{\sqsubseteq}$ denotes an arbitrary subset of $\mathrm{D}_{\neq}(T_t)$, then category $\mathcal{C}(T_t)$ is defined as $\mathcal{C}(D_{\sqsubseteq}) = \{T_{t,\sqsubseteq} | \mathrm{D}_{\neq}(T_{t,\sqsubseteq}) = D_{\sqsubseteq}\}$, *i.e.*, $D_{\sqsubseteq}$ functions as a key to group subtables whose decision values coincide with elements of $D_{\sqsubseteq}$. The tree most suitable for $\mathfrak{S}$ among trees corresponding to subtables in $\mathcal{C}(D_{\sqsubseteq})$ is found and is denoted as $\Gamma_{\mathfrak{S},T_{t,\sqsubseteq}}$ whose definition is given in followings.

$$\Gamma_{\mathfrak{S},T_{t,\sqsubseteq}} = \left\{\Gamma_{T_{t,\sqsubseteq}} \in \mathcal{A}_{\mathfrak{S},t'}\left(\mathcal{C}(D_{\sqsubseteq})\right) \middle| \min_{\Gamma_{T_{t,\sqsubseteq}}} \mathfrak{S}\right\}$$

Where $\mathcal{A}_{\mathfrak{S},t'}\left(\mathcal{C}(D_{\sqsubseteq})\right)$ denotes the set containing trees of subtables in $\mathcal{C}(D_{\sqsubseteq})$ generated in $\mathcal{A}_{\mathfrak{S},t'}$. After we find $\Gamma_{\mathfrak{S},T_{t,\sqsubseteq}}$ for each $\mathcal{C}(D_{\sqsubseteq})$, any $\Gamma_{T_{t,\sqsubseteq}} \in \mathcal{A}_{\mathfrak{S},t'}\left(\mathcal{C}(D_{\sqsubseteq})\right)$ will be replaced by $\Gamma_{\mathfrak{S},T_{t,\sqsubseteq}}$, *i.e.*, once a $\mathfrak{S}$-optimal tree for a $\mathcal{C}(D_{\sqsubseteq})$ is found, all trees related with tables in $\mathcal{C}(D_{\sqsubseteq})$ are replaced by the new-found optimal tree. Finally, we attempt to find $\mathfrak{S}$-optimal tree for all $\mathcal{C}(D_{\sqsubseteq})$ which is the result of $\mathcal{A}_{\mathfrak{S}}(T_t)$.

$$\mathcal{A}_{\mathfrak{S}}(T_t) = \left\{\Gamma_{T_{t,\sqsubseteq}} \in \bigcup_{D_{\sqsubseteq}} \mathcal{A}_{\mathfrak{S},t'}\left(\mathcal{C}(D_{\sqsubseteq})\right) \middle| \min_{\Gamma_{T_{t,\sqsubseteq}}} \mathfrak{S}\right\}$$

## 4. Algorithm Design

In this section, algorithms associated with Step 1 and Step 3 are discussed, namely, Algorithm 1.1 and Algorithm 1.2 in Step 1; Algorithm 3.1 and Algorithm 3.2 in Step 3. Algorithms associated with Step 1 are described in Subsection 4.1 and algorithms of Step 3 are explored in Subsection 4.2.

### 4.1. Algorithms for Pareto Optimal Values

As depicted in Figure 2, Algorithm 1.1 attempts to split a given decision table $T$ in a recursive manner. After adding $\mathrm{T}$ to $\mathrm{DAG}$, it tries to find an table $T'$ not split yet from $\mathrm{DAG}$ and insert subtables of $T'(f_i, a_j)$ where $f_i \in \mathrm{E}(T')$ and $a_j \in \mathrm{C}(T, f_i)$ as nodes to $\mathrm{DAG}$. Node $T'$ and all $T'(f_i, a_j)$ are connected by directed edges starting from $T'$ and ending at $T'(f_i, a_j)$. Edges are marked by corresponding condition values $(f_i, a_j)$. Node insertion only occurs when subtable is not found in $\mathrm{DAG}$ and if there is an edge connecting $T'$ and existing $T'(f_i, a_j)$, the mark of edge will be added by $(f_i, a_j)$ rather than adding a new edge. This node insertion and edge modification or addition continues until there is no tables found in $\mathrm{DAG}$ which are not split.

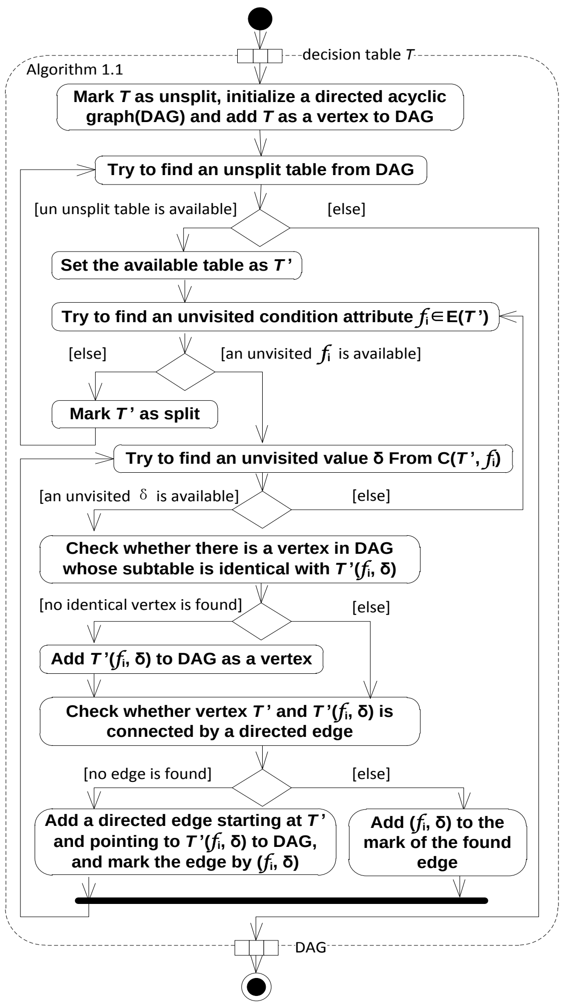
decision table T
Algorithm 1.1
Mark T as unsplit, initialize a directed acyclic graph(DAG) and add T as a vertex to DAG
Try to find an unsplit table from DAG
[un unsplit table is available]
[else]
Set the available table as T'
Try to find an unvisited condition attribute fi∈E(T')
[else]
[an unvisited fi is available]
Mark T' as split
Try to find an unvisited value δ From C(T', fi)
[an unvisited δ is available]
[else]
Check whether there is a vertex in DAG whose subtable is identical with T'(fi, δ)
[no identical vertex is found]
[else]
Add T'(fi, δ) to DAG as a vertex
Check whether vertex T' and T'(fi, δ) is connected by a directed edge
[no edge is found]
[else]
Add a directed edge starting at T' and pointing to T'(fi, δ) to DAG, and mark the edge by (fi, δ)
Add (fi, δ) to the mark of the found edge
DAG

**Figure 2. Algorithm 1.1 for Generating DAG Based on Decision Table**

Algorithm 1.2 provides core functionality for generating Pareto points based on mathematical descriptions made in Section 3. The strategy is starting generating from terminal nodes of $\mathrm{DAG}$ whose Pareto points are of form $(1,0)$, then moving upward to subtables $T'$ containing only terminal nodes providing values for $\mathcal{F}_{T'}^{f_i}$ and compute $\mathcal{F}_{T'}$ by adding the smallest $\mathcal{F}_{T'}^{f_i}$ with $|\mathrm{C}(T')|$. Before $T$ is reached, it continues to find the next subtables $T'$ whose subtables have completed their Pareto point computation. For $T$, each Pareto point attached with $f_i \in \mathrm{E}(T)$ is considered as an optimal Pareto point as output of proposed algorithm. The details are illustrated in Figure 3.

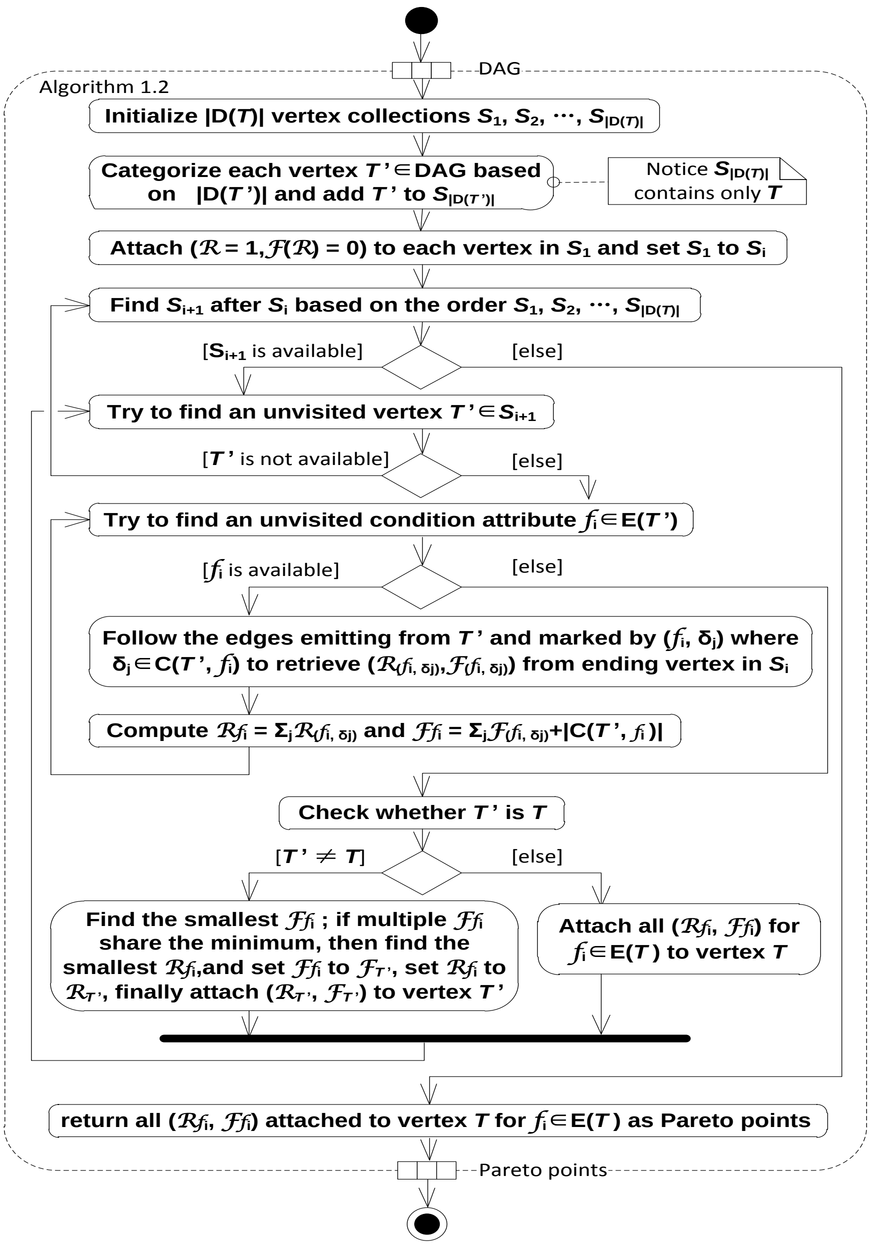
DAG
Algorithm 1.2
Initialize |D(T)| vertex collections S1, S2, ···, S|D(T)|
Categorize each vertex T'∈DAG based on |D(T')| and add T' to S|D(T')|
Notice S|D(T)| contains only T
Attach (R = 1, F(R) = 0) to each vertex in S1 and set S1 to Si
Find Si+1 after Si based on the order S1, S2, ···, S|D(T)|
[Si+1 is available]
[else]
Try to find an unvisited vertex T'∈Si+1
[T' is not available]
[else]
Try to find an unvisited condition attribute fi∈E(T')
[fi is available]
[else]
Follow the edges emitting from T' and marked by (fi, δj) where δj∈C(T', fi) to retrieve (R(fi, δj), F(fi, δj)) from ending vertex in Si
Compute Rfi = ΣjR(fi, δj) and Ffi = ΣjF(fi, δj)+|C(T', fi)|
Check whether T' is T
[T' ≠ T]
[else]
Find the smallest Ffi ; if multiple Ffi share the minimum, then find the smallest Rfi,and set Ffi to FT', set Rfi to RT', finally attach (RT', FT') to vertex T'
Attach all (Rfi, Ffi) for fi∈E(T) to vertex T
return all (Rfi, Ffi) attached to vertex T for fi∈E(T) as Pareto points
Pareto points

**Figure 3. Algorithm 1.2 for Generating Pareto Optimal Values**

## 4.2. Algorithms for Generating Decision Trees

This section introduces algorithms implementing $\mathcal{A}_{\mathfrak{S}}$. There are two subroutines, *i.e.*, Algorithm 3.1 for generating none-empty subtables of $T$ and Algorithm 3.2 for building decision tree $\Gamma_T$ of $T$ based on Tree Selection Criterion. Tree Selection Criterion refers to $\mathfrak{S}$ in our case, but it can also be other logic like minimizing entropy measure $\left(-\sum_{k=1}^{|D(T)|_{\neq}}(p_k\log_2(p_k)+(1-p_k)\log_2(1-p_k))\right)$ where $p_k = |C(d_k)|/|D(T)|$.

Algorithm 3.1 finds all subtables of $T$. All found subtables are preserved in a collection denoted by $S$ and Algorithm 3.2 attempts to generate $\Gamma_T$ of $T$ based on criterion defined by $\mathfrak{S}$. The implementation details of Algorithm 3.2 are shown in Figure 5.

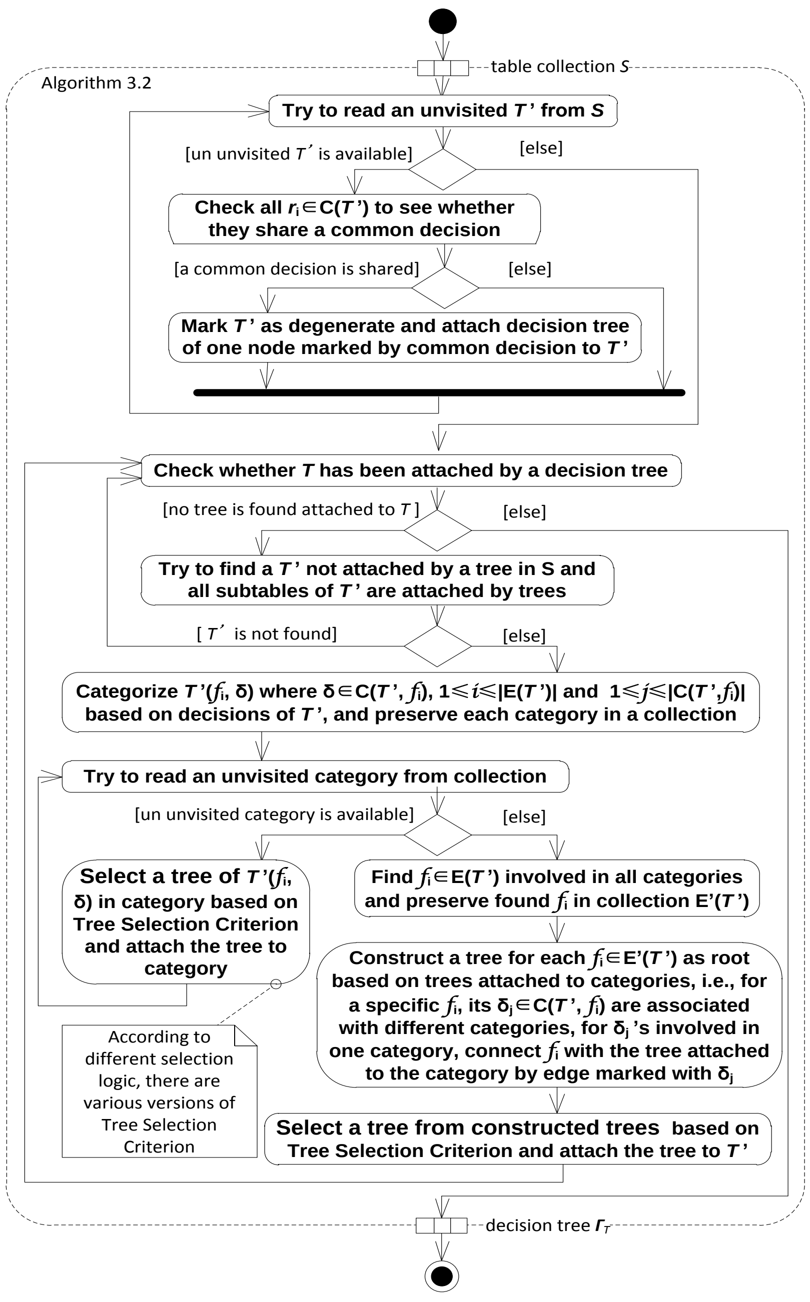
table collection S
Algorithm 3.2
Try to read an unvisited T' from S
[un unvisited T´ is available]
[else]
Check all rᵢ∈C(T') to see whether they share a common decision
[a common decision is shared]
[else]
Mark T' as degenerate and attach decision tree of one node marked by common decision to T'
Check whether T has been attached by a decision tree
[no tree is found attached to T]
[else]
Try to find a T' not attached by a tree in S and all subtables of T' are attached by trees
[T´ is not found]
[else]
Categorize T'(fᵢ, δ) where δ∈C(T', fᵢ), 1≤i≤|E(T')| and 1≤j≤|C(T',fᵢ)| based on decisions of T', and preserve each category in a collection
Try to read an unvisited category from collection
[un unvisited category is available]
[else]
Select a tree of T'(fᵢ, δ) in category based on Tree Selection Criterion and attach the tree to category
Find fᵢ∈E(T') involved in all categories and preserve found fᵢ in collection E'(T')
Construct a tree for each fᵢ∈E'(T') as root based on trees attached to categories, i.e., for a specific fᵢ, its δⱼ∈C(T', fᵢ) are associated with different categories, for δⱼ's involved in one category, connect fᵢ with the tree attached to the category by edge marked with δⱼ
According to different selection logic, there are various versions of Tree Selection Criterion
Select a tree from constructed trees based on Tree Selection Criterion and attach the tree to T'
decision tree Γ_T

**Figure 5. Algorithm 3.2 for Generating Decision Tree $\Gamma_T$ of $T$**

## 5. Experimental Results

We employed the proposed algorithms to generate decision trees for a small business project which automatically classifies narrowband LED pack by scanning two-dimensional barcode printed on each pack. The barcode consists of two main parts of digital information, *i.e.*, attributes shared by narrowband LED packed inside and unique identifier composed by three fields of barcode for pack itself. Partial attributes and unique identifier are crucial for classifying and chosen as $f_i \in \mathrm{E}(T)$ for a decision table $T$ to represent classifying rules. The classifying logic of system is implemented according to decision trees generated by using proposed algorithms based on $T$, and round trip is made through analyzing database of system and adjust $\mathfrak{L}(r_j^t)$ for each rule represented as a row in decision table. The system has been running over 6 months and its statues at different time are reflected through Figure 6 to Figure 8.

The overall number of processed packs is shown in Figure 6. The packs are cyclically classified at start of each month and the number is increasing as time elapsed. During the inactivated period of each month, system data is collected and analyzed for generating new trees adapted by system in turn. The number starts at 0 and ceases at about 15 thousands. Since the unique identifier is checked for each processing, processing involves more and more data preserved in system as time elapsed.

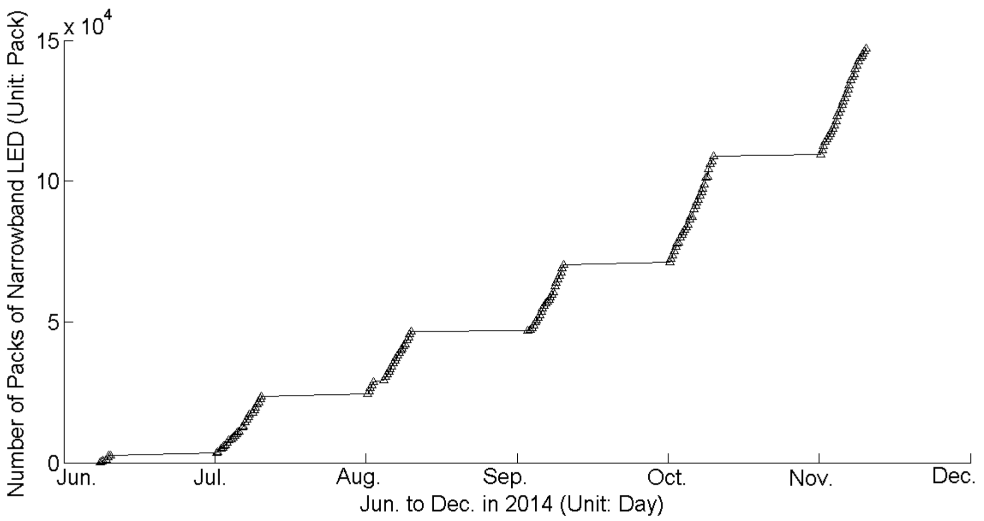

**Figure 6. Number of LED Packs Accumulated in System**

At beginning of each month, packs are classified by system and number of processed packs per day is depicted in Figure 7. From Figure 7, the fact that number of packs per day coarsely increases as time elapsed. This reflects system users gained increasing confidence about system and they became more and more dependent on automatic classifying provided by system.

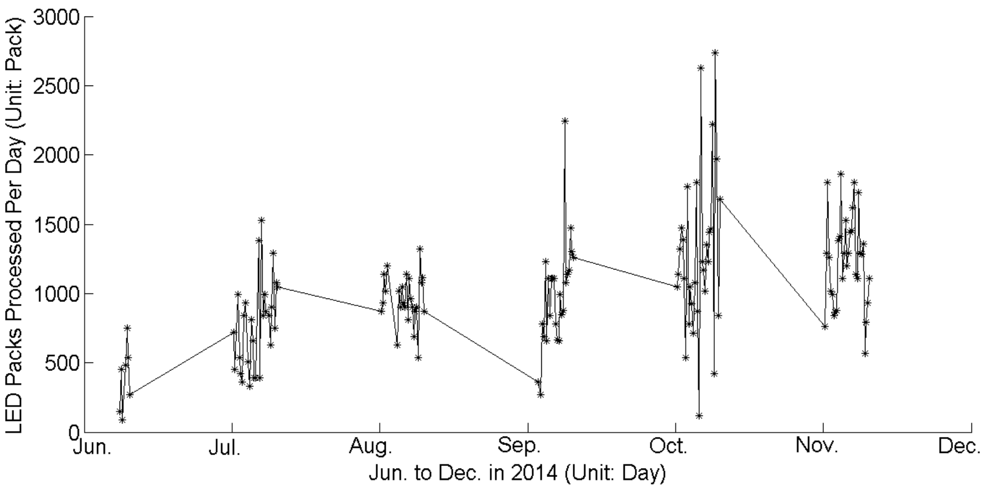

**Figure 7. Number of LED Packs Processed per Day in System**

The reason answers for the increasing confidence of system can be investigated in

Figure 8. Figure 8 shows the average time of processing packs during a day. Since the units of all abscissas of Figure 6 to Figure 8 are day, we can clearly compare data shown in three figures uniformly. Although number of processed packs increases in Figure 6, the average processing time oscillates fiercely during Jun. and Jul. in which accumulated number remains less than 3 thousand. Contrarily, as accumulated number increases from 3 thousand in Aug. to 15 thousand in Nov. shown in Figure 6, processing time stays stably. Surprisingly, most of processing time during Nov. shown in Figure 8 is lower than its predecessors during Oct. and Sep. This explains why confidence about system is increasing as shown in Figure 7.

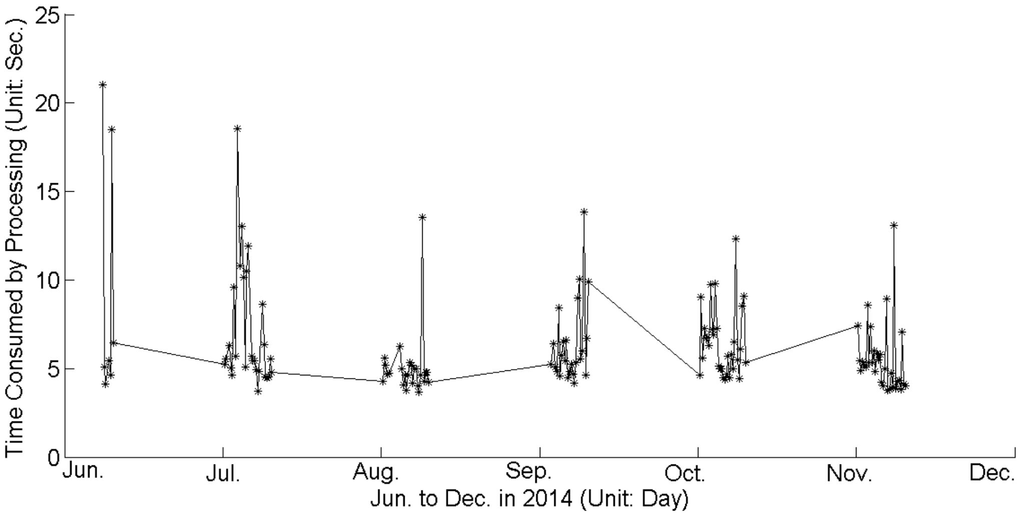

**Figure 8. Average Time of Processing Barcode per Day**

## 6. Conclusion

This paper mathematically describes round trip pattern for generating decision tree based on given decision table. The round trip consists of tree-to-table trip and table-to-tree trip. For the initial tree-to-table trip, Bayes theorem combined with Pareto optimal values of objective function is adapted for calculating probability-like factor to shape decision tree for generating. Table-to-tree trip is reflected by generating decision tree based on table altered in initial tree-to-table trip. The structure of generated tree is determined by a criterion primarily depending on factors embedded in altered table. After the first round trip, a system adapting decision tree generated by proposed algorithm is running and data collected from system can be employed for deriving factors to alter table, and new tree is generated based on table altered by system data. The round trip continues until some performance indicator of system stays stable. This procedure is applied in a small business project. In this real-world application, we follow the initial round trip to implement decision tree in system. Then subsequent round trip depends on system data, and system is altered according to new-generated decision trees. The statistical data retrieved from database of system is analyzed and the results showed the round trip pattern maintains system performance at a reasonable level.

## References

[1] H. A. Elsalamony, “Detecting distorted and benign blood cells using the Hough transform based on neural networks and decision trees”, in Emerging Trends in Image Process., Computer Vision and Pattern Recognition, 1st ed. USA: Morgan Kaufmann, vol. 30, **(2014)**, pp. 457-473.
[2] C. Grana, M. Montangero and D. Borghesani, “Optimal decision trees for local image processing algorithms”, Pattern Recognition Letters, vol. 33, no. 16, **(2012)**, pp. 2302-2310.
[3] M. Czajkowski, M. Grześ and M. Kretowski, “Multi-test decision tree and its application to microarray data classification”, Artificial Intell. in Medicine, vol. 61, no. 1, **(2014)**, pp. 35-44.
[4] P. Moutis and N. D. Hatziargyriou, “Decision trees aided scheduling for firm power capacity provision by virtual power plants”, International Journal of Elect. Power & Energy Syst., vol. 63, **(2014)**, pp. 730-739.
[5] E. Bastı, C. Kuzey and D. Delen, “Analyzing initial public offerings' short-term performance using decision trees and SVMs”, Decision Support Syst., vol. 73, **(2015)**, pp. 15-27.
[6] F. Chen, D. Chi and Y. Wang, “Detecting biotechnology industry's earnings management using Bayesian network, principal component analysis, back propagation neural network, and decision tree”, Econ. Modelling, vol. 46, **(2015)**, pp. 1-10.
[7] D. M. Farid, L. Zhang, C. M. Rahman, M. A. Hossain and R. Strachan, “Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks”, Expert Syst. with Application, vol. 41, no. 4, **(2014)**, pp. 1937-1946.
[8] S. Hussain, “Total path length and number of terminal nodes for decision trees”, Proceeding Computer Science, vol. 35, **(2014)**, pp. 514-521.
[9] I. Chikalov, S. Hussain and M. Moshkov, “Totally optimal decision trees for monotone Boolean functions with at most five variables”, Proceeding Computer Science, vol. 22, **(2013)**, pp. 359-365.
[10] C. Luo, T. Li, H. Chen and L. Lu, “Fast algorithms for computing rough approximations in set-valued decision systems while updating criteria values”, Information Science, vol. 299, **(2015)**, pp. 221-242.
[11] T. Amin, I. Chikalov, M. Moshkov and B. Zielosko, “Dynamic programming approach to optimization of approximate decision rules”, Information Science, vol. 221, **(2013)**, pp. 403-418.
[12] M. Azad, B. Zielosko, M. Moshkov and I. Chikalov, “Decision rules, trees and tests for tables with many-valued decisions–comparative study”, Proceeding Computer Science, vol. 22, **(2013)**, pp. 87-94.
[13] M. Azad and M. Moshkov, “Minimization of decision tree average depth for decision tables with many-valued decisions”, Proceeding Computer Science, vol. 35, **(2014)**, pp. 368-377.
[14] H. Attouch, G. Garrigos and X. Goudou, “A dynamic gradient approach to Pareto optimization with nonsmooth convex objective functions”, Journal of Math. Anal. and Application, vol. 422, no. 1, **(2015)**, pp. 741-771.
[15] F. J. Pulido, L. Mandow and J. L. P. Cruz, “Multi-objective shortest path problems with lexicographic goal-based preferences”, European Journal of Operational Research, vol. 239, no. 1, **(2014)**, pp. 89-101.