Wireless Service Attributes Classification and Matching Mechanism Based on Decision Tree

Min Peng[1], Laurence T. Yang[2], Wuqing Zhao[1], Naixue Xiong[3]
[1] Department of Computer, Wuhan University, Wuhan, Hubei, China,
hhdawn@public.wh.hb.cn
[2] Department of Computer Science, St. Francis Xavier University
Antigonish, NS, B2G 2W5, Canada, ltyang@stfx.ca
[3] Information Science, Japan Advanced Institute of Science and Technology,
JAIST, 1-309, 1-8, Asahidai, Nomi, Ishikawa, Japan, naixue@jaist.ac.jp

Abstract

In this paper, SeviceCuts, a decision tree based model is proposed, considering the special attributes of wireless service and the normal need of users, helping service decision Agent classify the wireless services adaptively. The decision tree is traversed based on some searching rule. A small number of matching rules are stored in the leaf node, which contain the most matching service strategies to users, and are linearly traversed to find the highest priority rule that matches the user's query requirement. The analysis of algorithm complication and performance shows that the efficiency of ServiceCuts decision tree model is better than traditional linear search structure and the normal binary decision tree structure.

1. Introduction

In the next-generation wireless network, wireless services will be more pervasive in the world. User profiles such as the location, the velocity (both speed and direction), the resource requirements of the mobile device, and the selection of the most appropriate services should be accurately determined. And it's important to maximize network efficiency and provide better Quality-of-Service (QoS) to different classes of users using different mobile devices in different environment [1].

In this paper, we are interested in designing an allocation mechanism to select the most appropriate service for mobile users. We identify various factors that can influence the QoS and requirements of services, which profile users' receive, and adjust the wireless service selection dynamically.

Our model is ServiceCut, it is a decision tree based agent model. At each node of decision tree, the set of rules present the services with different strategies, and it will match and decide the service type of users' requirement. Each time a service query arrives, the decision tree is traversed based on some searching rules, and at the last finds a leaf node. A small number of matching rules stored in the leaf node, which contain the most matching service strategies to users, and are linearly traversed to find the highest priority rule that matches the requirement.

2. The wireless services selection problem

The wireless service provided to users is determined by its service attributes[2], and the value of each attribute is relied on service content details on user's mobile device such as data type, limitation of wireless bandwidth, pixel, jitter, latency. The decision problem of the consumer is which service profile to select that maximizes the quality and minimizes the cost for the set of application, task and user defined constraints.

Usually, wireless service customers know what they want by the applications, but little of them are well-informed about the details of attributes in each service. So it's hard for them to make their decisions. On the other hand, at some time step later the users are in a different context, but still select running the same service for the same tasks and requirements, but now facing supply from a different set of base stations offering different service attributes. In the worst case the users may in fact be in a more hurried state than before. Or maybe they have to start another application, or terminated old ones. It's difficult for users to select some wireless services for their tasks and applications, because the complexity and dynamicity of wireless services. So there should be some allocation mechanism and an agent-user system to control for this complexity.

In this paper, we create the ServiceCut, which is a decision tree based model. Considering the special attributes of wireless service and the normal need of users, helping service decision Agent classify the

978-0-7695-3096-3/08 $25.00 © 2008 IEEE
DOI 10.1109/WAINA.2008.43

wireless services adaptively.

3. The mechanism

3.1 Definition

ServiceCut is a decision tree [3-6] based agent model, allocates the wireless services to users, based on the services attributes and the requirements of users. At each node in the decision tree, the set of current rules present the services with different strategies, and are split based on information from one or more fields in the service rules. When the ServiceCut tree is created, it can be used to match and decide the service strategies that users want. Each time a service query arrives, the decision tree is traversed based on some searching rules, if there is appropriate service rule can be find, a leaf node will be reached. A small number of matching rules stored in the leaf node, which contain the most matching service strategies to users (figure 1).

Each node in the decision tree has associated with it:

a. Service set, which is a decision tree Γ, divided into L discrete subset (region) B_1, B_2, …, B_L, $\Gamma = \bigcup_{i=1}^{L} B_i$ and for each $i \neq j$, there is $B_i \bigcap B_j = \emptyset$.

b. A region B_v, $v = 1, \ldots, L$,which is a k-tuple of ranges, intervals, discrete values and binary class labels, $B_v = \{x_1, x_2, ..., x_k\}$. All the tuple are the key attributes of wireless services, v is the vth node of decision tree, holding k dimensions, each of them corresponding to a service attribute.

c. Cut C(v). The cut C(v) is defined by a dimension i and $nc(i,v)$, $nc(i,v)$ means the number of cut times in the ith dimension of node v. The Region B_v is cut (or partitioned) in dimension i(i.e. cuts in the interval of the ith tuple of Region B_v).

d. A list of rules, R_i: $\{R_{i1}, R_{i2}, ..., R_{ik}\}$, $i = 1 \ldots n$. Rules with different parameters of ranges value present the different services strategies. The root of the tree has all the service rules associated with it.

e. A service query, $Q_A(s_j)$, $j = 1 \ldots m$. A query matches a rule $R_i(v)$: $\{R_{j1}, R_{i2}, ..., R_{ik}\}$, iff all the header fields of s_j of service query $Q_A(s_j)$ match the corresponding regions and dimensions in B_v : $\{x_1, x_2, ..., x_k\}$.

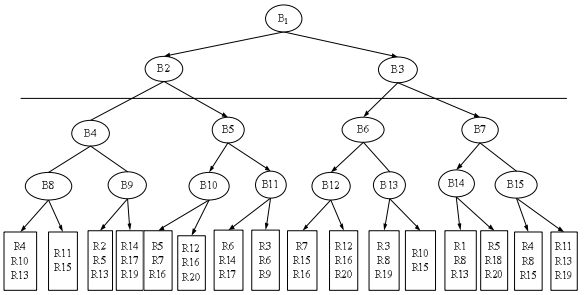

Figure 1. Service Classification by ServiceCuts

3.2 Building the serviceCuts tree

There is a set of wireless services containing N service entities, each of the service contains k dimensions, which denote the different attributes of the services. We construct decision tree in a top-down manner [7-9], starting with the root node that has all services rules associated with it. At any point we consider a node and the set of rules associated with it. The decision whether to split the node and if so, which test (cut-rule) to use, is locally optimal with respect to the optimization criteria and set of cut-rules: the cost of each cut-rule is computed under the assumption that the respective potential children are leaf nodes.

3.2.1 Unique selection domain and multi select domain. Since the tree regions are composed of k-tuples of intervals and discrete values and binary class labels match the service attributes correspondingly, we try to pick the dimensions which will lead to the most uniform distribution of the rules when the nodes are split into sub-nodes. In a given service searching, some dimensions can be matched with either an interval values or partial of discrete values contained in the region of tuples, but there are other dimensions should be matched with unique value, i.e., the result of match should be a unique select from the dimension range. So we can divide the k-tuples space into two types, one is the unique select dimensions, the match result in them must be an unique selection, in spite of how many values of the rule dimension can be considered and selected. Another are the multi-selected dimensions in which the rule dimensions matching can return more than one.

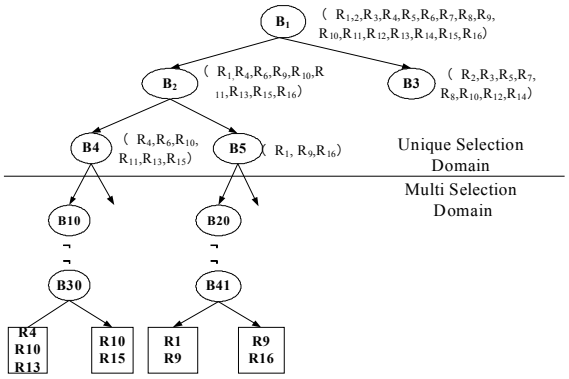

Figure 2. Attributes Splitting on Decision Tree

3.2.2 Region cutting. On each decision tree node, cutting dimension is selected, and the maximum rules number of each cut is minimized. The cut time in each dimension should be limited, which can take balance between cost of storage space and querying time,

although the repetitious cuts on one node are permitted. Frequently, the class structure depends on combinations of variables, and the standard tree with single dimension split will do poorly at uncovering the structure.

There are two combination procedures during region cutting. One is searching for a best linear combination split, another one is for Boolean combination split. Some variables do little contributes to split will be elided from combination. We will talk about it in the next section.

3.2.3 Detailed restriction in decision creation. Unique selecting dimensions will be cut firstly, and then multi select domain dimensions. Instead of cutting the unique selecting dimensions one by one at each step, our solution is to consider all of the unique dimensions firstly.

1. Unique select domain splitting

We can find that there are coordinate relationships between some unique selected attributes. During one split, the cooperated attributes $x_i, x_j, \ldots$ in field node t are combined. Based on the experience of handheld device and wireless service training set, a set of constant coefficients $a = (a_i, a_j, \ldots,)$ can be specified, such that $\|a\|^2 = a_i^2 + a_j^2 + \ldots = 1$, and search for the best split of the form $a_i x_i + a_j x_j + \ldots \le c$, c is the given threshold. In our case, for the combination split of hardware attributes, c is a threshold describing the computing capacity of handheld device.

2. Multi select domain

(1) Select the splitting node

The number of interval split executed in the *i*-th dimension should be suitable. In order to decrease the depth of the tree (which will accelerate query time) at the expense of increasing storage, we also represent a decision taken on the most representative dimensions, and combine some dimensions in one split. We given a fixed bucket size to the size of the set of rules at each node, when the current node size is larger than the acceptable bucket size, the node is split in a number (nc) of child nodes.

(2) Splitting order and interval value

Computing the priority levels of each dimension and the frequency counts associated with every cut of the dimension based on the training of service samples.

a. Every dimension in multi selection domain will be specified a priority level value $\sigma(X_i, \Gamma)$, *Xi* is the *i*-th dimension of the service tree Γ, smaller it is, higher the priority and weightiness the dimension will be.

b. Based on the knowledge and experience about the service set instances, the frequency counts of each split interval or unique element value of each dimension are accumulated, it is describe as $\varphi(v, X_i, \Gamma)$, v is a cut value of the dimension X_i in service class Γ. The probability of v in all split value of X_i is

$$\Pr(v) = \frac{\varphi(v, X_i, \Gamma)}{\sum_{e \in value\ of\ Xi} \varphi(e, X_i, \Gamma)}$$

(3) Decision tree splitting

a. In the construction of the decision tree, for each ancestor node, the dimension with the highest priority level will be considered to split first, and the cut will result several sub-ranges of the ancestor node. The split will continue until there are no intervals or unique elements values in this dimension to be equal or larger than a given threshold ω. Then the second smallest dimension in those sub-ranges will be considered to split respectively, and so on.

b. For those dimensions having same priority levels, can execute a combinative split. And the dimensions whose elements number is larger than the mean of the number of total elements will be considered to split.

c. For the i-th dimension that is considered to split, the determining of the number nc(i) of interval split executed in it should be suitable. It should decrease the depth of the tree and with the least expense of increasing storage.

d. Base on step (2), the frequency count of each interval or unique value of each dimension $\varphi(v, X_i, \Gamma)$ is known. And during the split, if the value is equal or larger than a given threshold ω, i.e.

$$\Pr(v) = \frac{\varphi(v, X_i, \Gamma)}{\sum_{e \in value\ of\ Xi} \varphi(e, X_i, \Gamma)} \ge \omega,$$

the value v will be split from others, and the dimension will be split into two types, one is the children nodes which the frequency counts are not smaller than a given threshold, another one is a child node with the remainder value. The split step recursive until there is no value equal or large than threshold ω. If there is no value's distribution probability equal or larger than the threshold in a dimension, this dimension will not be split. The threshold ω can be varied to tradeoff time against storage.

3.3 Decision tree search

When customers ask for wireless services from their mobile devices, based on their requirement and the wireless communication condition, research on the decision tree, and the appropriate service will be matched.

Some of the decision tree nodes are clustered based on cluster rules. One cluster includes some nodes in decision tree that are neighbored and continuous.

During the tree search, instead of a single node, a cluster is traversed firstly to increase the efficiency and veracity.

3.3.1 Cluster.

1. Key Dimension

In the multi select domain, each dimension is evaluated with a priority level number, denoting the importance degree of the corresponding service attribute within all the service attributes. The most important attribute is called Key Attribute, with the highest priority level number, which is described as a minimal number. Usually, the service ID is abstractive and uneasy to remember to user, so it's not practical to be named as Key Attribute. On the other side, before the using of a wireless service, user knows what they want clearly, so the service name is appropriate to be a Key Attribute.

2. Cluster

All the unique select domain nodes are aggregated as one cluster, which has the following characters:

a. Splitting in some dimensions, a ancestor node will be cut into several subnodes, a user's querying will be matched and only matched with one of those subnodes.

b. There will no overlapped ranges between all subnodes that belonged to a same ancestor.

c. The splitting will be stopped until the sizes of all unique selected attributes are smaller than the bucket threshold.

The splitting in multi select domain begins from the dimension with highest priority level. In fact, the splitting must begin after all unique select domain dimensions are stopped their splitting, therefore, the Key Attributes in unique select domain leaves will be cut firstly.

All the nodes split from the same Key Attribute are organized as a cluster.

Identify all the clusters by the sequence and ID of nodes where the Key Attributes are cut, $C_i = \{B_j\}, j \in 1, 2, ..., L$.

Match the user's querying requirement with the organized clusters in services search:

a. Firstly, match the querying with the unique select domain cluster, therefore, cluster C_1, finding the matched cluster leaf node.

b. Based on this leaf node, continue its searching path in the cluster belongs to the Key Attribute, begin another cluster search.

c. Travers the elements in this cluster, if there is no correct interval or single element of attributes can match the requirement, search is stopped. Otherwise, the search will arrive at one or multi leaves nodes in the cluster, and all the attributes in the nodes will determine one or multi wireless services correspondingly.

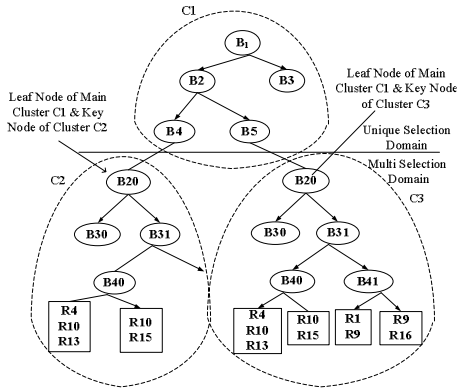

Figure 3. Nodes Cluster on Decision Tree

3.3.2 Searching algorithm.

Input: the service Tree S, the query condition (attributes value): $\{Y_1, Y_2, ..., Y_h\}$

Output: matched service rules

```
int function SearchTree(Service Y, Node B)
    if (B = Null) return NIL;
    if (B.leaf)
        return values of B.leaf
    else
        for v of Y
            if (Pr(v) ≥ σ)
                return SearchTree(Y, B.left)
            else
                return SearchTree(Y, B.right)

int function CooperatedSearch (ServiceQuery A(Ya, Yb,…Yh), Node B)
    if (B = Null) return NIL;
     if (B.leaf)
        return values of B.leaf
     else
        if ∃Yd, Yf, ..., Yh ∈ a cooperated attributes set
           (d, f, h ∈ K) do (C ← τd Yd + τf Yf + ... + τh Yh) or
                            (C ← τd Yd × τf Yf ... × τh Yh)
           in which ||τ||² = τd² + τf² + ... + τh² = 1
           if C ≥ ω
                return CooperatedSearch (A(Ya, Yb,…Yh), B.left)
           else
                return CooperatedSearch (A(Ya, Yb,…Yh),
                    B.right)

int function Search (ServiceQuery Q, Tree Γ )
Node B = root;
Y1, Y2,…Yj ← the attributes of service in query Q
    for all Ya, Yb,...Yh ∈ UniqueSelectionDomain ∧
    is a cooperated attributes set (a,b,...,h ∈ [1, j])
        return CooperatedSearch (Q(Ya, Yb,…, Yh), B)
    for other Y ∈ UniqueSelectionDomain ∧
    ¬cooperated attribute sets
        Node root = B.leaf
        return SearchTree(Y, B.leaf)
          KeyAttributesNode ← B.leaf.leaf

for each cluster C specified by one key attribute of Multi Selection
Domain (one leaf node of B.leaf in cluster C1)
    Node root = KeyAttributesNode;
    Y1, Y2,…Yj ← the attributes of service in query Q
          for all Ya, Yb,...Yh ∈ multiSelectionDomain ∧
              is a cooperated attributes set
```

($a, b, ..., h \in [1, j]$)
return *CooperatedSearch* ($Q(Y_a, Y_b, \ldots, Y_b)$, *KeyAttributesNode*)
for other $Y \in MultiSelectionDomain \wedge \neg cooperated\ attribute\ sets$
Node root = *KeyAttributesNode.leaf*
return
SearchTree(*Y*, *KeyAttributesNode.leaf*)
ServiceRules ← *KeyAttributesNode.leaf.leaf*
return ServiceRules.

for each ServiceRules list in the leaf node
linearly search

4 Performance analysis

4.1 Memory

The algorithm of building trees, like the compute complication of ID3 algorithm, is the linear function of the product such as the rule number, the attribute number and the node number. There are rule subsets stored in leaf nodes, so extra memory will be used. However, the space increased is far less than O(*Nc*) (*N* is Service number, *c* is Attribute number).

Assume that the method of accessing memory being random, it is reasonable to value the memory condition through the sum of all the rule number and the number of inside nodes. Time evaluation of the service strategy contains all the rule number and the process from the beginning to the end of deciding which service operation can be chosen. Every service strategy evaluation is the process from the root node to the leaf node of the decision tree through the routes of decision branches.

4.2 Time

To the time spent by searching service strategy, the worst condition is not to split the nodes (linear search in all the service rules), the cost is O(*Ncl*), *l* is the section region of every attribute or dispersive average number.

The decision tree search algorithm practically can be cut into two parts: search the qualified leaf node and search the linear table pointed by this leaf node. The first part is searching the binary tree, and at the same time this can be divided into the unique select section search and multi select section search. In the unique-selection section, it is just like a decision subtree. Assumed this part the number of the splitting level is *k*, has *m* nodes, the search of this section considering the worst condition of binary tree $m = 2^k - 1$, then the comparing count will be $k = \lceil \log_2(m+1) \rceil$ at most. In the multi select section, the condition of search is based on a key attribute and searches in the cluster related to it. The time complication in the cluster is not able to surpass the time complication O(2*L*) of the full binary tree with the same level. And every complication of search linear table is O(*n*(*d*-*L*)), *n* is the biggest rule number in the linear table and $n << N$, *N* is the total rule number, *d* is the attribute number split in the multi select section. According to the nature of binary tree, there are 2*L* leaf nodes. As a result the time complication in the cluster can't surpass O(2*Ln*(*d*-*L*)) and be less than time complication O(*Nd*) of the traditional linear search method. Total search time complication is O(*k*) + O(2*Ln*(*d*-*L*)).

4.3 Performance

In the ServiceCuts model, size of the storage space depends on the number of nodes in the process. One node contains a head and the pointer pointing to its children. Every head should be 2 bytes, and every pointer occupies 2 bytes, the number of branches in one level should equals to the number of children. The position of head is used to distinguish and mark the nodes including node's serial number, the related cluster number of the stored number and other information.

In the practical performance analysis, using the wireless service software, hardware and data attribute classification described in section 3, ServiceCuts will be compared with normal linear access and binary access. Part of the parameters of this wireless rule base is shown in table 1. The storage space compare is in table 2. The access time is shown in table 3. Performance analysis is in the figure 4.

Table 1. The related parameters of five kinds of rule sets with same attributes classification but different attributes intervals in the decision tree ServiceCuts

Rule Set	Attribute Classification Number	Total Attribute Number	Average Decision Tree Nodes Number	Total Service Rule Number
1	12	17	51	72
2	12	24	127	148
3	12	41	4095	8129
4	12	45	131070	524288
5	12	74	8643089	33554432

Table 2. The total storage space occupied in linear, binary and SeviceCuts. The storage space unit is word, every word is 32 bits.

Rule Set	Attribute classification number	Linear Table	Binary Tree	ServiceCuts
1	72	512	10250	510
2	148	800	11530	1270
3	8192	1148	327670	40950

Table 3. The total access time in linear, binary tree and SeviceCuts. The unit of the storage space is word, every word is 32 bits.

Rule Set	Attribute Classification Number	Linear Table	Binary Tree	ServiceCuts
1	72	16	12	4
2	148	23	14	7
3	8192	37	20	9
4	524288	44	24	12
5	33554432	56	29	19

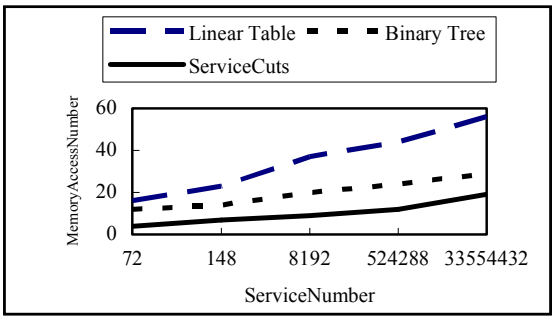

Figure 4. The time of access memory changing corresPonding to the change of services number in three different search structures: linear, binary, decision tree ServiceCuts.

According to the result of above assessment and analysis figures, we can conclude that after using the tree structure as the storage structure, the space used to store service information is larger than linear structure, but the access time is cut down during the search process. So practically, our method uses some storage space cost to exchange some benefits of speed, and expand and increase the convenience in rule based operation. At the same time, compared to traditional binary tree structure, ServiceCuts decision tree model decreases the storage space and access time both, so the performance our model is better than traditional binary tree structure.

5. Conclusion and future works

The decision tree classifier is a kind of important datamining and classification model which can be used in handheld devices. Based on decision tree classifier, we designed wireless service select model of ServiceCuts to provide fast services that maximizing network efficiency and with better Quality-of-Service (QoS) to different classes of users using different mobile devices in different environment.

SeviceCuts is an Agent adaptive decision model based on decision tree algorithm series, which considers the special attributes of wireless service and the normal need of users and helps service decision Agent classify the wireless services adaptively. Different to some mature decision tree split algorithm, our decision tree is created correspondingly to the wireless service character. According to the service match attributes difference, cut it into unique select domain and multi select domain, and split them separately. We use the method of cluster in the decision tree search, and raise the efficiency and accuracy of search greatly. The analysis of algorithm complication and performance shows that the efficiency of ServiceCuts decision tree model is much better than traditional linear search structure and the normal binary decision tree structure.

The future work will emphasize on modifying the decision tree structure to improve the search efficiency furthermore.

References

[1] G. Lee, S. Bauer, P. Faratin, J. Wroclawski. "Learning User Preferences for Wireless Services Provisioning", In Proc. of *AAMAS'04*, July 19-23, 2004, New York, USA.

[2] Kyung-Chang kim, Suk-Woo Yun. "MR-Tree: A Cache-Conscious Main Memory Spatial index Structure for Mobile GIS". In Proc. of *Web and Wireless Geographical information Systems (W2GIS 2004)*, LNCS 3428, pp. 167-180, 2005.

[3] J. Ross Quinlan. "C45, Programs for Machine Learning", *Morgan Kaufmann Publishers Press*, pp.3-33, 1993.

[4] Yasuhiko Morimoto, Hiromu Ishii, Shinichi Morishita. "Efficient Construction of Regression Trees with Range and Region Splitting". *In Machine Learning*, Vol. 45, 2001.

[5] T.V. lakshman, D. Stiliadis. "High-Speed Policy-based Packet Forwarding Using Efficient multi-dimensional Range Matching". In Proc. of *SIGCOMM'98 Vancouver*, B.C.,pp.203-214, 1998.

[6] Sumeet Singh, Florin Baboescu, George Varghese, Jia Wang. "Packet Classification using Multidimensional Cutting". In Proc. of *SIGCOMM'03*, Karlsruhe, Germany, pp.213-222, 2003.

[7] Carl-Christian kanne, Matthias Brantner, Guido moerkotte. "Cost-Sensitive Reordering of Navigational Primitives". In Proc. of *SIGMOD 2005*, Maryland, USA, pp. 742-753, 2005.

[8] Edith Cohen, Carsten Lund. "Packet Classification in Large ISPs: Design and Evaluation of Decision Tree Classifiers". In Proc. of *SIGMETRICS'05*, Banff, Alberta, Canada, pp.73-84, 2005.

[9] Domenico Cantone, Alfredo Ferro, Alfredo Pulvirenti. "Antipole Tree Indexing to Support Range Search and K-Nearest Neighbor Search in Metric Spaces". In Proc. of *IEEE Transactions on knowledge and Data Engineering*, Vol. 17, no. 4, pp.535-550, 2005.