ALGORITHMS FOR ROUTING LOOKUPS AND PACKET CLASSIFICATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Pankaj Gupta

December 2000

© Copyright by Pankaj Gupta 2001
All Rights Reserved

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Nicholas W. McKeown
(Principal Adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Balaji Prabhakar
(Co-adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Mark Horowitz

Approved for the University Committee on Graduate Studies:

To my mother&land

Abstract

The work presented in this thesis is motivated by the twin goals of increasing the capacity and the flexibility of the Internet. The Internet is comprised of packet-processing nodes, called routers, that route packets towards their destinations, and physical links that transport packets from one router to another. Owing to advances in optical technologies, such as Wavelength Division Multiplexing, the data rates of links have increased rapidly over the years. However, routers have failed to keep up with this pace because they must perform expensive per-packet processing operations.

Every router is required to perform a forwarding decision on an incoming packet to determine the packet's next-hop router. This is achieved by looking up the destination address of the incoming packet in a forwarding table. Besides increased packet arrival rates because of higher speed links, the complexity of the forwarding lookup mechanism and the large size of forwarding tables have made routing lookups a bottleneck in the routers that form the core of the Internet. The first part of this thesis describes fast and efficient routing lookup algorithms that attempt to overcome this bottleneck.

The second part of this thesis concerns itself with increasing the flexibility and functionality of the Internet. Traditionally, the Internet provides only a "best-effort" service, treating all packets going to the same destination identically, and servicing them in a first-come-first-served manner. However, Internet Service Providers are seeking ways to provide differentiated services (on the same network infrastructure) to different users based on their different requirements and expectations of quality from the Internet. For this, routers need to have the capability to distinguish and isolate traffic belonging to different flows. The ability to classify each incoming packet to determine the flow it belongs to is called packet classification, and could be based on an arbitrary number of fields in the packet header. The second part of this thesis highlights some of the issues in designing efficient packet classification algorithms, and describes novel algorithms that enable routers to perform fast packet classification on multiple fields.

Acknowledgments

First and foremost, I would like to thank my adviser, Nick McKeown, for his continued support and guidance throughout my Ph.D. Nick, I have learned a lot from you — whether in doing research, writing papers or giving presentations, yours has been an inspiring influence.

I am extremely grateful to my co-adviser, Balaji Prabhakar, for being a constant source of encouragement and advice. Balaji, I truly appreciate the time and respect you have given me over the years.

Several people have helped greatly in the preparation of this thesis. I am especially grateful to Mark Ross (Cisco Systems), Youngmi Joo and Midge Eisele for reading a draft of my entire thesis and giving me invaluable feedback. I also wish to acknowledge the immensely helpful feedback from Daniel Awduche (UUNET/Worldcom) and Greg Watson (PMC-Sierra) on Chapter 1, Srinivasan Venkatachary (Microsoft Research) on Chapter 2 and Lili Qiu (Cornell University) on Chapter 4. Darren Kerr (Cisco) kindly gave me

access to the real-life classifier datasets that enabled me to evaluate the performance of two of the algorithms mentioned in this thesis.

I have had the pleasure of sharing office space at Gates 342 with the following people: Youngmi, Pablo, Jason, Amr, Rolf, Da, Steve and Adisak. Besides them, I appreciate the chance of being able to interact with some of the smartest people in Balaji's and Nick's research groups: Devavrat, Paul, Kostas, Rong, Sundar, Ramana, Elif, Mayank, Chandra. To all of you, my Stanford experience has been greatly enriched by your company.

There are several other amazing people whom I met while at Stanford, became friends with, and whom I will remember for a long long time. Aparna, Suraj, Shankar, Mohan, Krista, Noelle, Prateek, Sanjay, John, Sunay, Pankaj, Manish and Rohit — I thank you for your companionship, best wishes and support.

I feel privileged to have been a part of ASHA Stanford (an action group for basic education in India). To witness the quality of dedication, commitment and dynamism displayed by ASHA's members is a treat without a parallel.

Last, and certainly the most, I thank my family: my father T.C. Gupta, my mother Raj-Kumari, my sister Renu and the relatively new and welcome additions (my brother-in-law Ashish, and my cute niece Aashna) for their unfailing love, encouragement and support.

Contents

Abstract vii
Acknowledgments ix
Contents xi
List of Tables xv
List of Figures xvii

CHAPTER 1

Introduction 1

1 Packet-by-packet IP router and route lookups ... 2
1.1 Architecture of a packet-by-packet router ... 5
1.2 Background and definition of the route lookup problem ... 6
1.2.1 Internet addressing architecture and route lookups ... 7
2 Flow-aware IP router and packet classification ... 17
2.1 Motivation ... 17
2.2 Architecture of a flow-aware router ... 19
2.3 Background and definition of the packet classification problem ... 19
3 Goals and metrics for lookup and classification algorithms ... 24
4 Outline of the thesis ... 28

CHAPTER 2

An Algorithm for Performing Routing Lookups in Hardware 31

1 Introduction ... 31
1.1 Organization of the chapter ... 32
2 Background and previous work on route lookup algorithms ... 32

2.1 Background: basic data structures and algorithms 32
2.1.1 Linear search 33
2.1.2 Caching of recently seen destination addresses 33
2.1.3 Radix trie 35
2.1.4 PATRICIA 37
2.1.5 Path-compressed trie 39
2.2 Previous work on route lookups 40
2.2.1 Early lookup schemes 40
2.2.2 Multi-ary trie and controlled prefix expansion 41
2.2.3 Level-compressed trie (LC-trie) 43
2.2.4 The Lulea algorithm 44
2.2.5 Binary search on prefix lengths 47
2.2.6 Binary search on intervals represented by prefixes 48
2.2.7 Summary of previous algorithms 51
2.2.8 Previous work on lookups in hardware: CAMs 51
3 Proposed algorithm 55
3.1 Assumptions 55
3.2 Observations 56
3.3 Basic scheme 57
3.3.1 Examples 59
4 Variations of the basic scheme 61
4.1 Scheme DIR-24-8-INT: adding an intermediate “length” table 62
4.2 Multiple table scheme 64
5 Routing table updates 67
5.1 Dual memory banks 68
5.2 Single memory bank 69
5.2.1 Update mechanism 1: Row-update 69
5.2.2 Update mechanism 2: Subrange-update 69
5.2.3 Update mechanism 3: One-instruction-update 70
5.2.4 Update mechanism 4: Optimized One-instruction-update 71
5.3 Simulation results 75
6 Conclusions and summary of contributions 76

CHAPTER 3

Minimum average and bounded worst-case routing lookup time on binary search trees **79**

1 Introduction 79
1.1 Organization of the chapter 81

2 Problem statement 81
3 Algorithm MINDPQ 88
3.1 The minimization problem 90
4 Depth-constrained weight balanced tree (DCWBT) 93
5 Load balancing 96
6 Experimental results 97
6.1 Tree reconfigurability 99
7 Related work 102
8 Conclusions and summary of contributions 103

CHAPTER 4

Recursive Flow Classification: An Algorithm for Packet Classification on Multiple Fields 105

1 Introduction 105
1.1 Organization of the chapter 106
2 Previous work on classification algorithms 106
2.1 Range lookups 107
2.2 Bounds from Computational Geometry 109
2.3 Linear search 110
2.4 Ternary CAMs 110
2.5 Hierarchical tries 112
2.6 Set-pruning tries 113
2.7 Grid-of-tries 115
2.8 Crossproducting 117
2.9 Bitmap-intersection 119
2.10 Tuple space search 120
2.11 A 2-dimensional classification scheme from Lakshman and Stiliadis. 122
2.12 Area-based quadtree 123
2.13 Fat Inverted Segment Tree (FIS-tree) 125
2.14 Summary of previous work 128
3 Proposed algorithm RFC (Recursive Flow Classification) 129
3.1 Background 129
3.2 Characteristics of real-life classifiers 129
3.3 Observations about the structure of the classifiers 131
3.4 The RFC algorithm 133

3.5 A simple complete example of RFC 139
4 Performance of RFC 140
4.1 RFC preprocessing 142
4.2 RFC lookup performance 146
4.2.1 Lookups in hardware 146
4.2.2 Lookups in software 150
4.3 Larger classifiers 152
5 Variations 152
5.1 Adjacency groups 153
6 Comparison with related work 158
7 Conclusions and summary of contributions 158

CHAPTER 5

Hierarchical Intelligent Cuttings: A Dynamic Multi-dimensional Packet Classification Algorithm 161

1 Introduction 161
1.1 Organization of the chapter 162
2 The Hierarchical Intelligent Cuttings (HiCuts) algorithm 163
2.1 Data structure 163
2.2 Heuristics for decision tree computation 166
3 Performance of HiCuts 168
3.1 Variation with parameters binth and spfac 172
3.2 Discussion of implementation of HiCuts 173
4 Conclusions and summary of contributions 174

CHAPTER 6

Future Directions 177

1 Ideal routing lookup solution 177
2 Ideal packet classification solution 179
3 Final words 180

Appendix A 181

Appendix B 183

Bibliography 185

List of Tables

TABLE 1.1. Class-based addressing 8
TABLE 1.2. The forwarding table of router R1 in Figure 1.10 16
TABLE 1.3. Some examples of value-added services 20
TABLE 1.4. Given the rules in Table 1.3, the router at interface X must classify an incoming packet into the following categories 21
TABLE 1.5. An example classifier 23
TABLE 1.6. Examples of packet classification on some incoming packets using the classifier of Table 1.5 23
TABLE 1.7. Lookup performance required as a function of line-rate and packet size 25
TABLE 2.1. An example forwarding table with four prefixes. The prefixes are written in binary with a '*' denoting one or more trailing wildcard bits — for instance, 10* is a 2-bit prefix 33
TABLE 2.2. Complexity comparison of the different lookup algorithms. A '-' in the update column denotes that incremental updates are not supported. A '-' in the row corresponding to the Lulea scheme denotes that it is not possible to analyze the complexity of this algorithm because it is dependent on the structure of the forwarding table 50
TABLE 2.3. Performance comparison of different lookup algorithms 51
TABLE 2.4. Memory required as a function of the number of levels 66
TABLE 2.5. Simulation results of different routing table update techniques 75
TABLE 3.1. An example forwarding table 83
TABLE 3.2. Routing tables considered in experiments. Unopt_srch is the number of memory accesses required in a naive, unoptimized binary search tree 98
TABLE 3.3. Statistics for the MINDPQ tree constructed at the end of every 0.5 million packets in the 2.14 million packet trace for the MAE_EAST routing table. All times/lengths are specified in terms of the number of memory accesses to reach the leaf of the tree storing the interval. The worst-case lookup time is denoted by luWorst, the average look up time by luAvg, the standard deviation by luSd. and the average weighted depth of the tree by WtDepth 101
TABLE 4.1. An example classifier 107
TABLE 4.2. Examples of range to prefix conversions for 4-bit fields 108
TABLE 4.3. Comparison of the complexities of previously proposed multi-dimensional classification algorithms on a classifier with N rules and d W-bit wide dimensions. The results assume that each rule is stored in $O(1)$ space and takes $O(1)$ time to determine whether it matches a packet. This table ignores the multiplicative factor of $(2W-2)^d$ in the storage complexity caused by splitting of ranges to prefixes 128
TABLE 4.4. An example 4-dimensional classifier 136

TABLE 4.5. The 4-dimensional classifier used in Figure 4.22 140
TABLE 4.6. Packet header fields corresponding to chunks for RFC Phase 0 142
TABLE 4.7. Average time to classify a packet using a software implementation of RFC 151
TABLE 4.8. An example classifier in four dimensions. The column headings indicate the names of the corresponding fields in the packet header. “gt N” in a field specification specifies a value strictly greater than N 154
TABLE 4.9. A qualitative comparison of some multi-dimensional classification algorithms 158
TABLE 5.1. An example 2-dimensional classifier 164

List of Figures

Figure 1.1 The growth in bandwidth per installed fiber between 1980 and 2005. (Source: Lucent Technologies.) 3

Figure 1.2 The growth in maximum bandwidth of a wide-area-network (WAN) router port between 1997 and 2001. Also shown is the average bandwidth per router port, taken over DS3, ATM OC3, ATM OC12, POS OC3, POS OC12, POS OC48, and POS OC192 ports in the WAN. (Data courtesy Dell'Oro Group, Portola Valley, CA)..... 4

Figure 1.3 The architecture of a typical high-speed router..... 5

Figure 1.4 Datapath of a packet through a packet-by-packet router..... 5

Figure 1.5 The IP number line and the original class-based addressing scheme. (The intervals represented by the classes are not drawn to scale.)..... 8

Figure 1.6 Typical implementation of the lookup operation in a class-based addressing scheme..... 9

Figure 1.7 Forwarding tables in backbone routers were growing exponentially between 1988 and 1992 (i.e., under the class-based addressing scheme). (Source: RFC1519 [26])..... 10

Figure 1.8 Showing how allocation of addresses consistent with the topology of the Internet helps keep the routing table size small. The prefixes are shown on the IP number line for clarity..... 12

Figure 1.9 This graph shows the weekly average size of a backbone forwarding table (source [136]). The dip in early 1994 shows the immediate effect of widespread deployment of CIDR..13

Figure 1.10 Showing the need for a routing lookup to find the most specific route in a CIDR environment..... 14

Figure 1.11 Showing how multi-homing creates special cases and hinders aggregation of prefixes... 15

Figure 1.12 This figure shows some of the header fields (and their widths) that might be used for classifying a packet. Although not shown in this figure, higher layer (e.g., application-layer) fields may also be used for packet classification. 18

Figure 1.13 Datapath of a packet through a flow-aware router. Note that in some applications, a packet may need to be classified both before and after route lookup..... 19

Figure 1.14 Example network of an ISP (ISP_1) connected to two enterprise networks (E_1 and E_2) and to two other ISP networks across a network access point (NAP)..... 20

Figure 2.1 A binary trie storing the prefixes of Table 2.1. The gray nodes store pointers to next-hops. Note that the actual prefix values are never stored since they are implicit from their position in the trie and can be recovered by the search algorithm. Nodes have been named A, B, ..., H in this figure for ease of reference..... 35

Figure 2.2 A leaf-pushed binary trie storing the prefixes of Table 2.1..... 37

Figure 2.3 The Patricia tree for the example routing table in Table 2.1. The numbers inside the internal nodes denote bit-positions (the most significant bit position is numbered 1). The leaves store the complete key values. 38
Figure 2.4 The path-compressed trie for the example routing table in Table 2.1. Each node is represented by (bitstring,next-hop,bit-position). 40
Figure 2.5 A 4-ary trie storing the prefixes of Table 2.1. The gray nodes store pointers to next-hops. . 42
Figure 2.6 An example of an LC-trie. The binary trie is first path-compressed (compressed nodes are circled). Resulting nodes rooted at complete subtries are then expanded. The end result is a trie which has nodes of different degrees. 45
Figure 2.7 (not drawn to scale) (a) shows the intervals represented by prefixes of Table 2.1. Prefix P0 is the "default" prefix. The figure shows that finding the longest matching prefix is equivalent to finding the narrowest enclosing interval. (b) shows the partitioning of the number line into disjoint intervals created from (a). This partition can be represented by a sorted list of end-points. 49
Figure 2.8 Showing the lookup operation using a ternary-CAM. P_i denotes the set of prefixes of length *i*. 53
Figure 2.9 The distribution of prefix lengths in the PAIX routing table on April 11, 2000. (Source: [124]). The number of prefixes longer than 24 bits is less than 0.07%. 57
Figure 2.10 Proposed DIR-24-8-BASIC architecture. The next-hop result comes from either *TBL24* or *TBLlong*. 57
Figure 2.11 *TBL24* entry format 58
Figure 2.12 Example with three prefixes. 60
Figure 2.13 Scheme DIR-24-8-INT 63
Figure 2.14 TBLint entry format. 63
Figure 2.15 Three table scheme in the worst case, where the prefix is longer than (*n*+*m*) bits long. In this case, all three levels must be used, as shown. 65
Figure 2.16 Holes created by longer prefixes require the update algorithm to be careful to avoid them while updating a shorter prefix. 68
Figure 2.17 Example of the balanced parentheses property of prefixes. 71
Figure 2.18 This figure shows five prefixes, one each at nesting depths 1,2 and 4; and two prefixes at depth 3. The dotted lines show those portions of ranges represented by prefixes that are also occupied by ranges of longer prefixes. Prefixes at depths 2, 3 and 4 start at the same memory entry A, and the corresponding parenthesis markers are moved appropriately. 72
Figure 2.19 Definition of the prefix and memory start and end of prefixes. Underlined PS (PE) indicates that this prefix-start (prefix-end) is also the memory-start (memory-end) marker.. 73
Figure 2.20 The optimized One-instruction-update algorithm executing Update(m,Y,Z). 74
Figure 3.1 The binary search tree corresponding to the forwarding table in Table 3.1. The bit-strings in bold are the binary codes of the leaves. 82
Figure 3.2 The optimal binary search tree (i.e., one with the minimum average weighted depth) corresponding to the tree in Figure 3.1 when leaf probabilities are as shown. The binary codewords are shown in bold. 84
Figure 3.3 The optimal binary search tree with a depth-constraint of 4, corresponding to the tree in Figure 3.1. 86
Figure 3.4 Showing the position of μ and k_μ 92
Figure 3.5 Weight balanced tree for Figure 3.1 with a depth-constraint of 4. The DCWBT heuristic is applied in this example at node v (labeled 1100). 96
Figure 3.6 Showing 8-way parallelism achievable in an alphabetic tree constructed using algorithm MINDPQ or DCWBT. 97
Figure 3.7 Showing how the average lookup time decreases when the worst-case depth-constraint is relaxed: (a) for the "uniform" probability distribution, (b) for the probability distribution

derived by the 2.14 million packet trace available from NLANR. X_Y in the legend means that the plot relates to algorithm Y when applied to routing table X. 99
Figure 3.8 Showing the probability distribution on the MAE_EAST routing table: (a) "Uniform" probability distribution, i.e., the probability of accessing an interval is proportional to its length, (b) As derived from the packet trace. Note that the "Equal" Distribution corresponds to a horizontal line at y=1.5e-5. 101
Figure 4.1 Geometric representation of the two-dimensional classifier of Table 4.1. An incoming packet represents a point in the two dimensional space, for instance, P(011,110). Note that R4 is completely hidden by R1 and R2. 110
Figure 4.2 The hierarchical trie data structure built on the rules of the example classifier of Table 4.1. The gray pointers are the "next-trie" pointers. The path traversed by the query algorithm on an incoming packet (000, 010) is also shown. 112
Figure 4.3 The set-pruning trie data structure built on the rules of example classifier of Table 4.1. The gray pointers are the "next-trie" pointers. The path traversed by the query algorithm on an incoming packet (000, 010) is also shown. 114
Figure 4.4 Showing the conditions under which a switch pointer is drawn from node w to node x. The pointers out of nodes s and r to tries Tx and Tw respectively are next-trie pointers. 115
Figure 4.5 The grid-of-tries data structure built on the rules of example classifier in Table 4.1. The gray pointers are the "next-trie" pointers, and the dashed pointers are the switch pointers. The path traversed by the query algorithm on an incoming packet (000, 010) is also shown. 116
Figure 4.6 The table produced by the crossproducting algorithm and its geometric representation of the two-dimensional classifier of Table 4.1. 118
Figure 4.7 The bitmap tables used in the "bitmap-intersection" classification scheme for the example classifier of Table 4.1. See Figure 4.6 for a description of the ranges. Also shown is classification query on an example packet P(011, 110). 120
Figure 4.8 The tuples and associated hash tables in the tuple space search scheme for the example classifier of Table 4.1. 121
Figure 4.9 The data structure of Section 2.11 for the example classifier of Table 4.1 The search path for example packet P(011, 110) resulting in R5 is also shown. 122
Figure 4.10 An example quadtree constructed by spatial decomposition of two-dimensional space. Each decomposition results in four quadrants. 124
Figure 4.11 The AQT data structure for the classifier of Table 4.1. The label of each node denotes {1-CFS, 2-CFS}. Also shown is the path traversed by the query algorithm for an incoming packet P(001, 010), yielding R1 as the best matching rule. 125
Figure 4.12 The segment tree and the 2-level FIS-tree for the classifier of Table 4.1. 126
Figure 4.13 The distribution of the total number of rules per classifier. Note the logarithmic scale on both axes. 130
Figure 4.14 Some possible arrangements of three rectangles (2-dimensional rules). Each differently shaded rectangle comprises one region. The total number of regions indicated includes the white background region. 132
Figure 4.15 A worst case arrangement of N rectangles. N/2 rectangles span the first dimension, and the remaining N/2 rectangles span the second dimension. Each of the black squares is a distinct region. The total number of distinct regions is therefore $N^2/4 + N + 1 = O(N^2)$ 133
Figure 4.16 Showing the basic idea of Recursive Flow Classification. The reduction is carried out in multiple phases, with a reduction in phase I being carried out recursively on the image of the phase I-1. The example shows the mapping of 2^S bits to 2^T bits in 4 phases. 134
Figure 4.17 Packet flow in RFC. 135
Figure 4.18 Example chopping of the packet header into chunks for the first RFC phase. L3 and L4 refer to the network-layer and transport-layer fields respectively. 135

Figure 4.19 An example of computing the four equivalence classes E0...E3 for chunk #6 (corresponding to the 16-bit transport-layer destination port number) in the classifier of Table 4.4.138
Figure 4.20 Pseudocode for RFC preprocessing for chunk *j* of Phase 0. 139
Figure 4.21 Pseudocode for RFC preprocessing for chunk *i* of Phase *j*. 140
Figure 4.22 This figure shows the contents of RFC tables for the example classifier of Table 4.5. The sequence of accesses made by the example packet have also been shown using big gray arrows. The memory locations accessed in this sequence have been marked in bold.... 141
Figure 4.23 Two example reduction trees for three phases in RFC. 143
Figure 4.24 Two example reduction trees for four phases in RFC. 143
Figure 4.25 The RFC storage requirement in Megabytes for two phases using the dataset. This special case of RFC with two phases is identical to the Crossproducting method of [95]. 144
Figure 4.26 The RFC storage requirement in Kilobytes for three phases using the dataset. The reduction tree used is tree_B in Figure 4.23. 145
Figure 4.27 The RFC storage requirement in Kilobytes for four phases using the dataset. The reduction tree used is tree_A in Figure 4.24. 146
Figure 4.28 This graph shows the average amount of time taken by the incremental delete algorithm in milliseconds on the classifiers available to us. Rules deleted were chosen randomly from the classifier. The average is taken over 10,000 delete operations, and although not shown, variance was found to be less than 1% for all experiments. This data is taken on a 333 MHz Pentium-II PC running the Linux operating system. 147
Figure 4.29 The preprocessing times for three and four phases in seconds, using the set of classifiers available to us. This data is taken by running the RFC preprocessing code on a 333 MHz Pentium-II PC running the Linux operating system. 148
Figure 4.30 An example hardware design for RFC with three phases. The registers for holding data in the pipeline and the on-chip control logic are not shown. This design achieves OC192c rates in the worst case for 40 byte packets. The phases are pipelined with 4 clock cycles (at 125 MHz clock rate) per pipeline stage. 148
Figure 4.31 Pseudocode for the RFC lookup operation. 150
Figure 4.32 The memory consumed by RFC for three and four phases on classifiers created by merging all the classifiers of one network. 153
Figure 4.33 This example shows how adjacency groups are formed on a classifier. Each rule is denoted symbolically by RuleName(value-of-field1, value-of-field2,...). All rules shown are assumed to have the same action. '+' denotes a logical OR. 155
Figure 4.34 The memory consumed by RFC for three phases with the adjacency group optimization enabled on classifiers created by merging all the classifiers of one network. The memory consumed by the basic RFC scheme for the same set of classifiers is plotted in Figure 4.35. 156
Figure 4.35 The memory consumed with four phases with the adjacency group optimization enabled on the large classifiers created by concatenating all the classifiers of a few different networks. Also shown is the memory consumed when the optimization is not enabled (i.e. the basic RFC scheme). Notice the absence of some points in the "Basic RFC" curve. For those classifiers, the basic RFC scheme takes too much memory/preprocessing time. .. 157
Figure 5.1 This figure shows the tree data structure used by HiCuts. The leaf nodes store a maximum of *binth* classification rules. 163
Figure 5.2 An example classifier in two dimensions with seven 8-bit wide rules. 164
Figure 5.3 A possible HiCuts tree with binth = 2 for the example classifier in Figure 5.2. Each ellipse denotes an internal node v with a tuple $\langle B(v), dim(C(v)), np(C(v))\rangle$. Each square is a leaf node which contains the actual classifier rules. 165
Figure 5.4 Pseudocode for algorithm to choose the number of cuts to be made at node 167
Figure 5.5 An example of the heuristic maximizing the reuse of child nodes. The gray regions correspond to children with distinct *colliding rule sets.* 168

Figure 5.6 Storage requirements for four dimensional classifiers for binth=8 and spfac=4. 169
Figure 5.7 Average and worst case tree depth for binth=8 and spfac=4. 170
Figure 5.8 Time to preprocess the classifier to build the decision tree. The measurements were taken using the time() linux system call in user level 'C' code on a 333 MHz Pentium-II PC with 96 Mbytes of memory and 512 Kbytes of L2 cache. 171
Figure 5.9 The average and maximum update times (averaged over 10,000 inserts and deletes of randomly chosen rules for a classifier). The measurements were taken using the time() linux system call in user level 'C' code on a 333 MHz Pentium-II PC with 96 Mbytes of memory and 512 Kbytes of L2 cache. 171
Figure 5.10 Variation of tree depth with parameters binth and spfac for a classifier with 1733 rules.172
Figure 5.11 Variation of storage requirements with parameters binth and spfac for a classifier with 1733 rules. 173
Figure 5.12 Variation of preprocessing times with binth and spfac for a classifier with 1733 rules. The measurements were taken using the time() linux system call in user level 'C' code on a 333 MHz Pentium-II PC with 96 Mbytes of memory and 512 Kbytes of L2 cache. 174

कर्मण्येवाधिकारस्ते मा फलेषु कदाचन ।
मा कर्मफलहेतुर्भूर् मा ते सङ्गोऽस्त्वकर्मणि ॥

You have the right to perform your prescribed
duty (Karma) but not to the fruits of action.
Never consider yourself the cause of the results
of your activities and never get attached to inaction.

Lord Krishna to Arjuna in Bhagvad-gita