Compressing Rectilinear Pictures and Minimizing Access Control Lists

David A. Applegate * Gruia Calinescu † David S. Johnson ‡
Howard Karloff § Katrina Ligett ¶ Jia Wang ‖

Abstract

We consider a geometric model for the problem of minimizing access control lists (ACLs) in network routers, a model that also has applications to rectilinear picture compression and figure drawing in common graphics software packages. Here the goal is to create a colored rectilinear pattern within an initially white rectangular canvas, and the basic operation is to choose a subrectangle and paint it a single color, overwriting all previous colors in the rectangle. Rectangle Rule List (RRL) minimization is the problem of finding the shortest list of rules needed to create a given pattern. ACL minimization is a restricted version of this problem where the set of allowed rectangles must correspond to pairs of IP address prefixes. Motivated by the ACL application, we study the special cases of RRL and ACL minimization in which all rectangles must be strips that extend either the full width or the full height of the canvas (*strip-rules*). We provide several equivalent characterizations of the patterns achievable using strip-rules and present polynomial-time algorithms for optimally constructing such patterns when, as in the ACL application, the only colors are black and white (permit or deny). We also show that RRL minimization is NP-hard in general and provide $O(\min(n^{1/3}, \mathrm{OPT}^{1/2}))$-approximation algorithms for general RRL and ACL minimization by exploiting our results about strip-rule patterns.

1 Background and Motivation

1.1 Rectilinear Pictures. Many of today's software packages that generate graphics, from `Xfig` to `PowerPoint`, share a common method for creating rectilinear patterns. Starting with a white rectangular canvas, the user repeatedly applies a "rectangle tool" with which one sweeps out a rectangular area and colors the interior with a specified color, overwriting any previous contents of that area. Most of the figures in this abstract were produced using such a tool. See for instance Figure 1, where the underlying canvas is a 4×4 grid, and a sequence of three rectangle operations is applied.

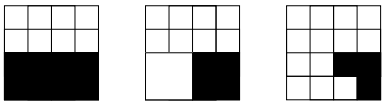

Figure 1: Creating a figure using rectangle rule applications.

The sequence depicted is inefficient since two operations would have sufficed. Given the manual work involved in producing such figures, one might well want to determine efficient plans of action in advance. Call a pair (R, c), where R is a rectangle and c is a color, a *rectangle rule*, and a sequence of such rules $(R_1, c_1), (R_2, c_2), \ldots, (R_n, c_n)$ a *rectangle rule list* or RRL. For an RRL $\mathcal{R}$, let $P_{\mathcal{R}}$ denote the pattern it produces. Given a target pattern P, our goal is to find a minimum-length RRL $\mathcal{R}$ such that $P_{\mathcal{R}} = P$. Such an RRL also provides a scheme for picture compression that is potentially more effective than the classic scheme in which a black rectilinear figure is represented by a collection of black rectangles that covers it. Moreover, a restricted version of the problem has an important application to Internet management.

1.2 Access Control Lists. Access control lists (ACLs) are used in network router line cards to determine which arriving packets should be forwarded to their destination and which should be dropped. For instance, an Internet Service Provider (ISP) might want its access routers to forward only packets that came from or are destined to customers who have officially been assigned to that router. ACLs can also be used to implement firewalls and to provide a range of levels of quality of service [Cis01]. An ACL consists of a sequence of *rules*. In an *extended ACL* a rule can be viewed as having five components: Source Range, Destination Range, Protocol, Port(s), and Action. The source and destination ranges are specified by binary strings s of length w or less, where w is the length of an IP address, currently 32 but expanding to 64 in IPv6. The string s matches all IP addresses that have it as a prefix; an empty string matches everything. Possible protocols

*`david@research.att.com`. AT&T Labs – Research, Room C224, 180 Park Avenue, Florham Park, NJ 07932.

†`calinesc@iit.edu`. Computer Science Department, Illinois Institute of Technology, Stuart Building, Room 236, 10 West 31st Street, Chicago, IL 60616. Research supported in part by NSF grant CCF-0515088.

‡`dsj@research.att.com`. AT&T Labs – Research, Room C239, 180 Park Avenue, Florham Park, NJ 07932.

§`howard@research.att.com`. AT&T Labs – Research, Room C231, 180 Park Avenue, Florham Park, NJ 07932.

¶`katrina+@cs.cmu.edu`. Department of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213. Research supported in part by AT&T.

‖`jiawang@research.att.com`. AT&T Labs – Research, Room A165, 180 Park Avenue, Florham Park, NJ 07932.

include IP, TCP, UDP, and ICMP, with IP matching all protocols and the others matching only packets labeled by that particular protocol. Ports can either be an individual port number, a range of such numbers, or "any," which matches everything. The action can either be *permit*, which allows the packet through, or *deny*, which causes the packet to be dropped. A *basic ACL* omits the protocol and port fields.

An ACL operates as follows. When a packet arrives, the router determines the first rule in the list that it matches and performs the action specified by that rule. If there is no match, the packet is dropped. In high-speed routers the classification is performed by special hardware called "Ternary CAMs" (TCAMs) that evaluate all the rules in parallel and output the lowest indexed match. These are expensive and impose limits on the size of ACLs, as do the overall memory constraints in a line card. Thus a natural optimization criterion for ACLs is to minimize their length while preserving the results of their actions. This will also reduce the maximum delay in routers that evaluate rules sequentially.

Other optimization criteria have also been studied. Many researchers have studied data structures and algorithms for quickly determining which rule in an ACL is the first match [GM99, EM01, KMT03, Tho03] or, when information about the data traffic is known in advance, trying to minimize the *average* time to find the applicable rule, either simply by reordering the ACL [Ful05] or by devising sophisticated decision tree classifiers [CL05]. None of these approaches have yet been implemented in real-world routers, however. Although such approaches can be of value when using software simulation to study router behavior, for now ACL minimization remains the most direct way to improve the access control performance of current real-world routers. AT&T's interest in ACL minimization was the inspiration for this paper.

In modeling the ACL minimization problem we can simplify matters by ignoring the protocol and port restrictions. These fields are not present in the simpler basic ACLs that are still sometimes used, and even within extended ACLs, most rules (85% or more of the rules in ACLs studied by Cohen and Lund [CL05] and similar proportions in access router ACLs we ourselves sampled) are basic in that the entries for protocol and port match everything. The more restrictive rules, even when present, are likely to have priority over the basic ones, so that optimizing over the basic rules would at least be a component of an effective heuristic for the general problem. Thus in what follows we assume that our ACLs are restricted to basic rules.

The problem of ACL minimization can be modeled in geometric terms. Consider a $2^w \times 2^w$ grid with a cell for each combination of a source and destination IP address (columns and rows indexed from 0 to $2^w - 1$). The action of an ACL can be viewed as coloring the cells of this grid, with the cell colored white if packets with that combination are denied, and black if they are permitted. Each rule applies to the rectangle in this grid whose x-coordinates are the IP addresses in the rule's source range and whose y-coordinates are those in the destination range. Thus each rule in an ACL is once again a pair (R, c), where R is a rectangle and c is a color, just as in our RRL application.

There are two distinctions between ACLs and RRLs, one minor and one major. The minor distinction is that the rules in an ACL are in the reverse order from those in an RRL. The major distinction is that the set of possible rectangles is highly constrained in ACLs since their x- and y-coordinates are determined by IP address prefixes. The widths and heights of rectangles must all be of the form 2^k for $0 \leq k \leq w$, and if a rectangle's width (height) is 2^k, then it must start in a column (row) whose coordinate is congruent to 0 mod 2^k. Note that this means that the projections of the rectangles on the axes are *laminar* in that for any two rectangles the projections are either disjoint or one is contained in the other.

1.3 Outline. The rest of this paper is organized as follows. In Section 2 we discuss previous results related to RRL and ACL minimization. In Section 3 we introduce and motivate the special case of *strip-rule* patterns. In ACL terms, this is the commonly encountered case in which no single rule restricts both the source and the destination address. We characterize the patterns that can be constructed with such rules and show that they can be efficiently recognized. In Section 4 we present polynomial-time algorithms for constructing optimal strip-rule RRLs and ACLs for 2-color strip-rule patterns. In Section 5 we show that RRL minimization for general 2-color patterns is NP-hard by exploiting insights into the strip-rule special case. In Section 6 we use subroutines for the strip-rule special case to build polynomial-time approximation algorithms for general RRL and ACL minimization with worst-case ratios of $O(\min(n^{1/3}, \mathrm{OPT}^{1/2}))$ and $O(w^2 \min(n^{1/3}, \mathrm{OPT}^{1/2}))$ respectively, where we assume the pattern is given to us by a length-n RRL/ACL that generates it. These are currently the best guarantees known to be attainable in polynomial time. We conclude in Section 7 by mentioning some additional results and open problems.

Note that all our results also hold if the rule lists must specify the color of the entire canvas/grid, i.e., when the background color does not come for free.

2 Previous Results

We know of no previous theoretical work on the RRL/ACL minimization problems as formulated here, but several special cases have been studied. The one-dimensional case where only the destination is restricted corresponds to the problem of minimizing routing tables, where under most routing protocols, the link out which a packet is sent depends only on its destination address. Here our rules correspond simply to intervals, rather than rectangles. The one-dimensional ACL problem can be solved in polynomial time by dynamic programming [DKVZ99, SSW03]. This holds even when $K > 2$ colors are allowed, as is the case with routing tables. If the pattern is specified by an ACL of length n using length-w IP addresses, the algorithm of Suri et al. [SSW03] produces an optimal equivalent ACL in time $O(Knw)$. Dynamic programming also can be used to minimize one-dimensional RRLs. We know of no formal references, but the problem was given with a solution as part of a June 2003 `TopCoder` programming contest under the name `StripePainter` [TC03] and it is not difficult to obtain a $O(Kn^3)$ running time bound.

Returning to two dimensions, the case of RRL minimization in which only black rectangle rules are allowed has been well-studied under the name RECTILINEAR PICTURE COMPRESSION. It was shown NP-Hard by Masek [Mas78] and MaxSNP-hard by Berman and DasGupta [BD97]. The best polynomial-time achievable approximation guarantee known is $O(\sqrt{\log n})$, where n is the number of black grid cells in the pattern [AR99]. In contrast, the optimal all-black ACL for a given pattern can be found in polynomial time. This was shown by Lakshmanan et al. in a database context [LNW+02]. The key insight is that when the projections of the allowed rectangles on both axes are laminar in the sense defined in Section 1.2, the ACL consisting of all the maximal black subrectangles has minimum length.

Unfortunately, the optimal all-black RRL/ACL can be a very poor approximation to the optimal black-and-white one. Consider a $(2n-1)\times(2n-1)$ grid in which the only black cells are those both of whose coordinates are odd. The optimal RRL/ACL consists of one black rule covering the whole pattern, overwritten by $n-1$ white column rules and $n-1$ white row rules, for a total of $2n-1$ rules, whereas the best all-black-rule list contains n^2 rules, yielding a ratio that is $\Theta(n) = \Theta(\mathrm{OPT}(P))$.

3 Strip-Rule Patterns: Definitions and Characterizations

In examining a sample of thousands of extended ACLs from AT&T's access routers, we observed that 95% of these lists contained no rule that restricted both source and destination IP addresses. In other words, in the vast majority of ACLs, every rule was a *strip-rule*, that is, one that colored either a set of contiguous full columns or a set of contiguous full rows of the grid. We refer to ACLs/RRLs made up entirely of strip-rules as "strip-rule ACLs/RRLs" and the patterns they generate as "strip-rule patterns." (Note that the set of patterns generated by strip-rule ACLs is identical to the set of patterns generated by strip-rule RRLs.)

Not all patterns are strip-rule patterns; for example, the simple 4-cell checkerboard pattern, with the lower left and upper right cells black and the other two white, is not. In fact, every black-and-white non-strip-rule pattern must contain such a checkerboard as a subarray. This is just one of many equivalent characterizations of strip-rule patterns, as illustrated by the following theorem.

THEOREM 3.1. *Suppose P is a 2-color pattern. The following statements are equivalent.*

1. *P is a strip-rule pattern.*
2. *P contains no checkerboard subpattern; that is, every 2×2 (not-necessarily-contiguous) subarray of P contains a monochromatic column or row.*
3. (Monotonicity Property) *For any color C the rows of P are hierarchical: For any two rows, the set of columns where C is present in the first row either contains or is contained in the set of columns where C is present in the second.*
4. *The monotonicity property holds with the roles of row and column reversed.*
5. *The following "Pick-Up-Sticks" algorithm always results in an "all-gray" grid: Let* gray *be a color not used in P. Call a column or row* pseudo-monochromatic *if it contains at most one color other than gray. Perform the following loop: While there is a pseudo-monochromatic column or row containing a cell with a non-gray color, choose such a column or row and color all its cells gray ("pick it up").*

The proof of Theorem 3.1 is relatively straightforward. Things get more complicated when we consider patterns P having more than 2 colors. For example, property (2) is no longer a characterization, as can be seen from Figure 2, which shows a 3×3 non-strip-rule pattern with three colors that has no monochromatic column or row, but each of whose 2×2 submatrices *does* have a monochromatic column or row. Surprisingly, this is the worst counterexample possible, as we have the following general characterization theorem.

1	3	1
3	3	2
1	2	2

Figure 2: A non-strip-rule pattern for which every 2×2 subarray has a monochromatic column or row.

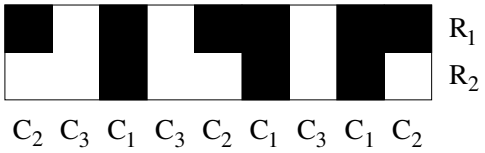

Figure 3: Pattern with columns and rows labeled by their equivalence class.

THEOREM 3.2. *Suppose P is a k-color pattern, $k > 2$. The following statements are equivalent.*

1. *P is a strip-rule pattern.*
2. *Every 2×2 and 3×3 subarray of P contains a monochromatic column or row.*
3. *The "Pick-Up-Sticks" algorithm always results in an empty grid.*

The forbidden subarray characterizations in Theorems 3.1 and 3.2 are of combinatorial interest, but are not the key to fast recognition of strip-rule patterns. For that we can simply run the Pick-Up-Sticks algorithm. By the characterization theorems, P is a strip-rule pattern if and only if the algorithm succeeds in reducing it to the empty grid, and the Pick-Up-Sticks test can be implemented to run in $O(n^2)$ time ($O(n^2 w)$ for ACLs), where n is the length of an RRL/ACL that produces P, or is the maximum dimension of the *effective grid* for the pattern, defined as follows.

For any rectilinear pattern P, call a horizontal or vertical line segment that separates and bounds two differently-colored regions a *boundary line*. Extend each boundary line in both directions so that it crosses the full canvas. We call the resulting grid the "effective grid" for the pattern, and note that all its grid cells will be monochromatic. Moreover, if the pattern is generated by an RRL/ACL with n rules, then the number of (internal) grid lines is at most $4n$ (or $2n$ if it is a strip-rule RRL/ACL).

In what follows it will be convenient to consider the variant on Pick-Up-Sticks we shall call *Maximal Pick-Up-Sticks* (MPUS), where instead of graying individual columns or rows we gray maximal contiguous sequences of columns or rows that collectively contain precisely one non-gray color. (In the case of ACLs we are further restricted to maximal contiguous sequences that yield legal strips, i.e., ones that correspond to IP prefixes.) We halt as soon as the only colors left in the pattern are gray and the background color (white). Like Pick-Up-Sticks this is not a well-defined algorithm – we don't say how to choose the next operation when more than one are available. However, even without specifying a tie-breaking rule we do get the following results. For a strip-rule pattern P, let $\mathrm{OPT}_S(P)$ denote the length of the minimum-length strip-rule RRL/ACL that generates P (it will always be clear from context whether we are talking about RRLs or ACLs).

THEOREM 3.3. *For any strip-rule pattern P,*

1. *All RRLs generable for P using MPUS are no longer than $2\mathrm{OPT}_S(P) + 1$, and.*
2. *All ACLs generable for P using MPUS are no longer than $w\mathrm{OPT}_S(P) + 3$.*

These bounds hold for any number of colors. The bound for RRLs follows from the fact that any maximal rule except possibly the last must reduce the number of effective grid lines by 1. The bounds cannot be significantly improved as we will show in the full paper.

4 2-Color Strip-Rule Patterns: Constructing Optimal RRLs and ACLs

In this section we sketch our polynomial-time algorithms for constructing optimal strip-rule RRLs and ACLs for 2-color strip-rule patterns. We shall assume that the pattern is given by its effective grid. It is not difficult to see that finding an optimal RRL/ACL is equivalent to finding an optimal ordering of operations for MPUS. In what follows, say that a column/row is "picked-up" by a rule if it is covered by the rule and was not all gray at the time the rule was applied. Hence no column/row can be picked up more than once. The "color" of the rule will be the color of the non-gray cells in the columns/rows picked up by the rule. We exploit the fact that the columns and rows of a pattern P can be divided into equivalence classes, with two columns (rows) being equivalent if their corresponding cells are identically colored in the original pattern P. For example, see Figure 3. Here the column equivalence classes are all-black columns (C_1), black-above-white columns (C_2), and all-white columns (C_3). Each of the row equivalence classes has a single member, with R_1 consisting of the top row and R_2 consisting of the bottom row. The following observations are easy to verify for any 2-color strip-rule pattern P.

1. In any MPUS rule, all the columns/rows that are picked up are in the same equivalence class.
2. If some member of an equivalence class is picked up by a rule of color c, then *all* members of that class must eventually be picked up by c-rules.

3. Every class is either picked up by a white rule, picked up by a black rule, or not picked up, and there is at most one row class and one column class in the third category.

4. For an equivalence class E, let $N_w(E)$ be the number of white cells its members contain in P and $N_b(E)$ the number of black cells. Then no two distinct column (row) equivalence classes have the same value of $N_w(E)$ or $N_b(E)$ (this follows from the monotonicity property in Theorem 3.1).

5. For any color c, no member of a class E can be picked up by a c-rule until all members of classes E' of the same type (column or row) with $N_c(E') > N_c(E)$ have also been picked up by c-rules.

6. At any point before the all-gray pattern is reached in the MPUS process, precisely two equivalence classes will be pseudo-monochromatic and hence ripe for pick-up. We will either have a "mixed state," where one class is a row class and the other a column class and both have the same color, or a "column state," where the two classes are column classes of different colors, or a "row state," where the two classes are row classes of different colors.

In light of the above, any RRL/ACL created by MPUS can be divided into a sequence of segments, each segment containing all the rules applied when a given pair of equivalence classes were pseudo-monochromatic. We can associate a 5-tuple label (E_1, c_1, E_2, c_2, U) with each such segment, whose entries are specified as follows: E_1 and E_2 are the two monochromatic classes and c_1 and c_2 are the colors of their non-gray cells just before the segment was applied. E_1 is the "newer" of the two classes in that it is the one that was not pseudo-monochromatic during the previous segment. The final parameter, U, is the set consisting of those members of E_2 that had not yet been picked up when the segment started. (Note that none of the members of E_1 can have been picked up earlier because this class was not previously pseudo-monochromatic.) The fact that some members of E_2 may already have been picked up in earlier segments, even in an optimal RRL/ACL, follows from the observation that the best way to pick up one equivalence class may sometimes involve picking up members of the other. This is illustrated by the pattern in Figure 3. Recall that an RRL must reduce the pattern to one in which all cells are either gray or (the background color) white. Here the initially available classes are C_1 (all-black columns) and C_2 (all-white columns). Any RRL that picks up all of C_1 before any member of C_2 or vice versa will require at least six rules, whereas there is an RRL of length five for the pattern that finishes C_1 first but picks up one member of C_2 in doing so.

In addition to a label, each segment has an *action* $i \in \{1, 2\}$, which is the index (from the segment's label) of the equivalence class that gets completely picked up as a result of the segment.

Our algorithms exploit several structural lemmas. First, suppose P is the target pattern, L is an RRL/ACL for P, and S is a segment of L. Define $pattern(P, L, S)$ to be the pattern that existed after all the rules in segments of L preceding S have been applied, i.e., P as modified by turning gray all columns and rows covered by rules in those segments.

LEMMA 4.1. *Suppose L and L' are two RRLs/ACLs for a given pattern P, and S and S' are segments of L and L' respectively whose associated 5-tuples are identical. Then $pattern(P, L, S) = pattern(P, L', S')$.*

This follows from property (5) above and additional arguments about the relationship between row and column equivalence classes that are picked up with different colors.

4.1 RRL-Specific Details. The next lemma specifies a normal form for segments of optimal RRLs. Call a pseudo-monochromatic column/row *active* if it contains at least one non-gray cell, and otherwise call it *gray*. Call an active column/row *color-c* if the non-gray cells it contains are color c. Call a sequence of contiguous pseudo-monochromatic columns/rows a color-c *block* if each member of the sequence is either gray or color-c, at least one member is color-c, and the sequence is maximal with respect to these properties. If c is a non-gray color (black or white), let $\bar{c}$ denote the other non-gray color. Say a color-c block is *embedded* if the columns (rows) immediately to its left and right (above and below it) are color-$\bar{c}$ columns (rows).

LEMMA 4.2. *For $i \in \{1, 2\}$, suppose a minimum-length RRL L for P contains a segment S with label (E_1, c_1, E_2, c_2, U) that picks up all active members of class E_i.*

1. *If $c_1 = c_2$, then one class consists of rows and the other of columns. Assume for specificity that E_i is a column class. (The row case is handled analogously.) Then there is a minimum-length RRL for P containing a segment S' with the same label and action as S that consists of a sequence of rules that pick up all the color-c_i column blocks in $pattern(P, L, S)$, one rule per block.*

2. *If $c_1 \neq c_2$, then both classes are column classes or both are row classes. Assume for specificity that they are column classes and $i = 1$. (The other cases are again handled analogously.)*

(a) If pattern(P, L, S) contains a non-embedded color-c_2 block, then there is a minimum-length RRL for P containing a segment S' with the same label and action as S that has the following structure. Start with rules that pick up all the embedded color-c_2 blocks in pattern(P, L, S), one rule per block. Then pick up all the (possibly newly-created) color-c_1 blocks, one rule per block.

(b) Otherwise, there is a minimum-length RRL for P containing a segment S' with the same label as S whose action is to pick up all the active c_2 columns (instead of all the active c_1 columns).

Case 1 holds because column and row rules that are adjacent in a segment and have the same color can be interchanged without affecting the resulting pattern. Case 2 is more complicated, but essentially follows from the fact that the sequence prescribed in 2(a) uses the minimum possible number of rules to pick up all of E_1, and at the same time picks up a maximum set of columns from the E_2 "for free."

We can exploit Lemmas 4.1 and 4.2 to construct a minimum-length RRL by running a Dijkstra-like shortest path algorithm on the following (implicitly constructed) *segment graph* for P. The vertices of our graph are the 5-tuples that can occur in an RRL for P, with the "source" vertex being the 5-tuple $(F_1, c_1, F_2, c_2, F_2)$, where the F_i are the two monochromatic classes in P and the c_i are their colors. Note that since both classes are new in this case, we can fix an arbitrary order for them. The "sink" vertices are those 5-tuples that correspond to patterns in which no black cells remain. Every non-sink vertex v has two out-arcs, each corresponding to the action of picking up one of the vertex's two pseudo-monochromatic classes, except for vertices to which case 2(b) of Lemma 4.2 applies. For these all the members of one of the two classes are in embedded blocks and we omit the arc for the action that picks up the other class. (Note that only one out-arc per vertex can be omitted, since if the members of one class are all contained in embedded blocks, there must be at least one member of the other class that is not.) The out-arc for action i corresponds to the Lemma 4.2 normal-form segment for v and action i and leads to the 5-tuple that results when one applies that segment to the pattern corresponding to v. The length of the arc is the number of rules in the normal-form segment. Our goal is to find the shortest path from the source vertex to a sink vertex.

Since the segment graph is layered, we can evaluate vertices in breadth-first order. The running time will thus be proportional to the number of vertices encountered multiplied by the time to construct the normal form segments for each. Unfortunately, the number of vertices is not *a priori* polynomially bounded. We might have to consider labels with all $2^{|E_2|} - 1$ possible nonempty subsets of E_2 as candidates for U. The next lemma shows that this is fortunately not the case.

LEMMA 4.3. (CONTAINMENT LEMMA) *If vertices $v = (E_1, c_1, E_2, c_2, U)$ and $v' = (E_1, c_1, E_2, c_2, U')$ are both reachable from the source vertex in the segment graph for P and $U \neq U'$, then either $U \subset U'$ or $U' \subset U$.*

Thus there can be at most $|E_2|$ reachable labels of the form (E_1, c_1, E_2, c_2, U) for given values of E_i and c_i. A careful analysis, taking into account the fact that the column classes are all disjoint, as are all the row classes, then leads to the conclusion that the total number of reachable states is $O(n^2)$ where n is the number of effective grid lines. The overall running time for the algorithm can then be shown to be $O(n^3)$.

Proof of Lemma: The proof is by induction on the depth of the vertices in the graph. The lemma clearly holds for depth 0, since the source vertex is the only vertex at this depth. Suppose it holds for depth $d \geq 0$ and that $v = (E_1, c_1, E_2, c_2, U)$ and $v' = (E_1, c_1, E_2, c_2, U')$ are vertices at depth $d + 1$. We may assume that neither U nor U' is the full class E_2, as otherwise the desired conclusion would hold trivially. There are two main cases to consider.

Case 1. E_1 and E_2 are both column classes or both row classes, in which case we must have $c_1 \neq c_2$. Assume for specificity that they are column classes. The reader may verify that the only possible arcs into v and v' come from the mixed-state vertices of the form (E_1', c_2, E_2, c_2, U) and $(E_1', c_2, E_2, c_2, U')$, respectively, where E_1' is the previously-picked-up row equivalence class that in the original pattern P had the fewest color-c_2 cells while still having color c_2 in all cells it had in common with the columns of E_1. But then, by induction either $U \subset U'$ or $U' \subset U$, as desired.

Case 2. One of E_1 and E_2 is a column class and the other is a row class, in which case $c_1 = c_2$. Assume for specificity that E_2 is the column class. Arcs into these two vertices can be from vertices of only two types, either (E_1', c, E_2, c_2, X) (*Type 1*) or (E_2, c_2, E_1', c, Y) (*Type 2*), where c is the other color from c_2 and E_1' is the previously-picked-up column class that has the fewest cells of color c while having color c in the cells it shares with the rows of E_1. Since v and v' are reachable from the source vertex, there must be at least one arc into each from a reachable vertex of depth d. Let a be such an arc into v and a' be such an arc into v' and let x and x' be the tail vertices for a and a' respectively. We break into three cases depending on the types of x and x'.

Case 2a. Both x and x' are Type-1 vertices. Denote them by $x = (E_1', c, E_2, c_2, X)$ and $x' = (E_1', c, E_2, c_2, X')$, where $U \subseteq X$ and $U' \subseteq X'$ since the actions applied to x and x' could not have enlarged the set of active columns in E_2. By our induction hypothesis we know that either $X \subseteq X'$ or $X' \subseteq X$. Assume without loss of generality that $X \subseteq X'$. We claim that this implies that $U \subseteq U'$. Let P_x and $P_{x'}$ be the patterns corresponding to x and x' respectively, and assume, again without loss of generality, that c_2 is black. Each of the normal-form segments corresponding to a and a' picks up all the embedded black blocks in its pattern. For an active black column in X to survive to be included in U, it must be the case that on at least one side of it there is no active white column between it and the boundary or between it and a non-pseudo-monochromatic column. Turning some set of all-gray columns into active black columns, as one does to $X'-X$ when one goes from x to x', will not change this fact. Thus all active columns that survive to be included in U will also be included in U' and $U \subseteq U'$ as desired.

Case 2b. Both x and x' are Type-2 vertices. Denote them by $x = (E_2, c_2, E_1', c, Y)$ and $x' = (E_2, c_2, E_1', c, Y')$. By the induction hypothesis we know that either $Y \subseteq Y'$ or $Y' \subseteq Y$. We can assume without loss of generality that $Y \subseteq Y'$. We claim that this implies that $U' \subseteq U$. Once again assume c_2 is black. For an active black column to survive to be included in U', it must again be the case that on at least one side of it there is no active white column between it and the boundary or between it and a non-pseudo-monochromatic column. Turning gray some set of active white columns, as one does to $Y'-Y$ when one goes from x' to x, will not change this fact. Thus all active columns that survive to be included in U' will also be included in U and $U' \subseteq U$ as desired.

Case 2c. One of x and x' is of Type 1, say x, and the other is of Type 2 In this case we can denote them by $x = (E_1', c, E_2, c_2, X)$ and $x' = (E_2, c_2, E_1', c, Y)$. Consider the potential vertex $x'' = (E_1', c, E_2, c_2, E_2)$, which corresponds to the same pattern $P_{x''}$ as does $(E_2, c_2, E_1', c, E_1')$, and consider the result of applying the normal-form segment that picks up all members of E_1', which we can denote by $(E_1, c_1, E_2, c_2, U'')$. Then, since $X \subseteq E_2$, we have by the argument in the first case above that $U \subseteq U''$. On the other hand, since $Y \subseteq E_1'$, we have by the argument in the second case that $U'' \subseteq U'$. Hence $U \subseteq U'$, as needed. Thus the desired conclusion holds in all three cases and by induction the lemma follows. ■

4.2 ACL-Specific Details. Our algorithm for finding optimal strip-rule ACLs is more complicated than that for RRLs, although the basic structure is the same. The key insight is to identify the normal form for a segment when the two pseudo-monochromatic classes are either both rows or both columns.

In ACLs the rectangles allowed in rules are restricted, and any legal column or row strip-rule can be specified by a color and an IP-address prefix p, representing the set of all columns/rows whose coordinate x, when written as a w-bit binary number, has p as a prefix. In what follows we shall represent such rules as pairs (p, c) where p is the prefix and c is the color. If (p, c) and (p', c') are both row or both column rules and p is a proper prefix of p' we shall say that (p, c) *contains* (p', c') – every column/row addressed by the latter rule is addressed by the former. Note that in any optimal ACL, (p', c') must precede (p, c) if it is contained in it; otherwise it would have no effect and could be deleted.

Now let us consider the segments of an ACL. Say a rule in such a segment is *undominated* if it is not contained in any other rule from that segment, and note that the set of columns picked up by the segment is completely determined by the undominated rules it contains. Moreover, in a minimum-length ACL the undominated rules in a segment can all be postponed to the end of the segment and arranged in any order without altering the resulting pattern, since they must already be preceded by any rules they dominate.

Define the *signature* for a segment to be a $(w+1)$-tuple $(x_0, x_1, x_2, \ldots, x_w)$, where x_i is the number of prefixes of length i that are undominated rules in the segment. Given two different signatures, consider the first component in which they disagree. We say that the signature with the larger value in this component is the *stronger* of the two. The *dominant signature* for a collection of segments is the strongest of their signatures.

LEMMA 4.4. *Suppose a minimum-length ACL for P contains a segment S with label (E_1, c_1, E_2, c_2, U) that picks up all active members of class E_i, $i \in \{1, 2\}$. Let S' be a segment with the same label that is a minimum-length list of rules that picks up all active members of E_i, consists only of rules of the same type (column/row) as E_i, and has the dominant signature among all such minimum-length segments. If S' does not pick up all active members of both E_i and the other class E_{3-i}, then there is a minimum-length ACL for P that contains S'. Otherwise, there is a minimum-length ACL for P that contains a segment with the same label as S that picks up class E_{3-i}.*

Note that in the latter case it can be shown that the normal-form segment that picks up all of E_{3-i} does not pick up all active members of E_i.

Proof of Lemma. Suppose the Lemma is false, and consider an ACL L for P with the maximum number of initial segments that do satisfy the Lemma. Consider the first segment S in L that fails to satisfy the Lemma.

Suppose first that $c_1 = c_2$, in which case one class is a row class and the other is a column class. Since all the rules of the optimal segment must be color-c_1 rules, they can be permuted in any way without affecting the resulting pattern. Thus we may rearrange rules so that the rules applying to E_i come first and pick up all the active members in E_i. Thus in the revised ACL the segment will end with the last of these rules. If the overall ACL is of minimum length, this segment must itself be of minimum length, and hence can have the structure claimed by the Lemma for this case. Thus the new ACL satisfies the Lemma for one additional initial segment, a contradiction.

So suppose that $c_1 \neq c_2$ and hence we are dealing either with two column classes or two row classes. Assume for specificity that they are both column classes and that i denotes the white class. As remarked above, we may assume that in S all undominated rules come at the end, with the undominated white rules preceding the undominated black rules. But note that this means that all the active white columns must have been picked up before the first undominated black rule is encountered, and so the segment must end before the first undominated black rule appears.

Thus S can be assumed to contain no undominated black rules. Suppose it nevertheless fails to have the desired properties. If S picks up the same set of active columns as S', we can replace S by S' (which by definition is no longer) and obtain an overall ACL for P that is no longer than the given one and satisfies the Lemma for one additional initial segment, again a contradiction. So S must either pick up some active black column not picked up by S' or fail to pick up some active black column that S' does pick up.

In what follows we assume that, over all minimum-length ACLs that contain segments S with the given label and action, S is one of minimum length. First suppose there is an active black column x picked up by S but not by S'. Let (p, c) be the undominated rule in S that covers column x. By the above argument, (p, c) must be a white rule. S' cannot contain (p, c) or any rule that dominates (p, c) since if it did it would pick up x. Let L_1 be the set of rules in S used to pick up columns addressed by p, and L'_1 be the corresponding set of rules for S'. Note first that we must have $|L_1| > |L'_1|$, as otherwise we could replace L'_1 by L_1 in S' and obtain a new segment that was no longer than S' and yet had a stronger signature, contradicting our assumption that S' had the dominant signature among all minimum-length segments that picked up all active members of E_i. Let L''_1 be the set of rules obtained by adding $(p, black)$ to L'_1. Then we have $|L''_1| \leq |L_1|$. Moreover, L''_1 will correctly pick up all the active columns addressed by p – all columns addressed by L'_1 must already be correctly picked up and any active column addressed by p but not picked up by L'_1 must be pseudo-monochromatic black. Thus replacing L_1 by L''_1 in S will yield an ACL that is no longer and has the same effect. However, note that the segment now has an undominated black rule, which we can assume goes at the end, and hence is no longer part of the segment. Thus the resulting segment S'' is shorter than S, contradicting our assumption that S had minimum length.

Thus we may assume that S only picks up active black columns that are also picked up by S'. Since S picks up all active members of E_i and S' has minimum length among segments that do so, we must have $|S'| \leq |S|$. Thus we can swap S' for S in our ACL without increasing its length. At the end of the segment fewer black columns may be active than before, but this will not prevent subsequent rules in the ACL from being applied. Thus the ACL will still handle the pattern P. If *no* active black columns remain after applying S', then we have constructed a minimum-length ACL in which the segment with the same label as S picks up the black class. Otherwise, this new ACL satisfies the Lemma for an additional initial segment, a final contradiction. ■

In the full paper we will show that for any label (E_1, c_1, E_2, c_2, U) and action i, every minimum-length segment with the dominant signature picks up precisely the same members of E_1 and E_2. We then present an $O(nw)$ dynamic programming algorithm that constructs such a segment. As in the RRL case, this can be incorporated into an overall dynamic programming algorithm for computing a minimum-length ACL for any given pattern P. That the state space for this algorithm is again of size $O(n^2)$ (and hence the algorithm runs in polynomial time) follows by a somewhat more complicated analogue of the Containment Lemma for RRLs.

5 NP-completeness for 2-color RRLs

In this section we provide a brief overview of our proof that the problem of constructing a minimum-length (general) RRL is NP-hard. The proof is by a transformation from RECTILINEAR PICTURE COMPRESSION. Suppose we are given a rectilinear black pattern P on a grid with a white background and asked if there is a collection of k or fewer black rectangles whose union is the pattern. We will construct a rectilinear black and white pattern P' and an integer k' such that P' can be generated by a length-k' RRL if and only if P can be represented as the union of k or fewer black rectangles.

If we could somehow restrict the RRL to use only black rules, we could simply use $P' = P$. Unfortunately, as we have seen, white rules cannot be ignored. Our solution is to embed P into a much bigger pattern which will effectively eliminate the usefulness of white rules in constructing P. To do this, we design a background for the figure that must be picked up by strip-rules if the RRL is to be sufficiently short. The background is a large grid constructed out of multiple copies of the 7×7 tile depicted in Figure 4, for which the minimum-length RRL is a strip-rule RRL with a specific structure. The construction is relatively straightforward, but the argument that it works is quite intricate due to the flexibility that 2-color rules provide.

6 Approximation Algorithms for Unrestricted Patterns

Given that the general problem of RRL minimization is NP-hard and the general ACL problem may be NP-hard as well, it is reasonable to look for good approximation algorithms. The only previously-studied algorithms that can be viewed as approximation algorithms for these problems are the ones that construct RRLs/ACLs using only black rules. Unfortunately, as observed in Section 2, even the best possible RRL/ACL consisting only of black rules can be off by a factor of $\mathrm{OPT}(P)$ for patterns where $\mathrm{OPT}(P) = \Theta(n)$.

Our new algorithms, which work for arbitrary numbers of colors and use Maximal Pick-Up-Sticks as a subroutine, represent a substantial improvement. In the RRL case, our algorithm, which we call the *Iterated Strip-Rule* algorithm (ISR), works as follows. We assume we are given the pattern by a length-n RRL that generates it. Suppose the effective grid for the pattern has N_{col} columns and N_{row} rows and assume without loss of generality that $N_{col} \geq N_{row}$. Note that $\mathrm{OPT}(P) \geq \lfloor N_{col}/2 \rfloor$ since no rule can introduce more than two new effective vertical grid lines. Our basic approach is to partition P into subrectangles that are all strip-rule patterns, each of which can be handled by MPUS. We do this for several different partitions, and take the best result, unless all are worse than the input RRL, in which case we return the latter.

More formally, for each integer q, $1 \leq q \leq N_{col}$,

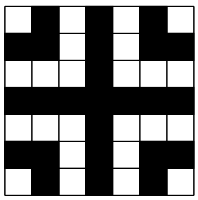

Figure 4: Tile in NP-completeness proof for RRLs.

we do the following. First we partition the grid into q vertical strips of widths as close to N_{col}/q as possible. Then we partition each strip into a sequence $B_1, B_2, \ldots$ of blocks, starting from the bottom. The odd-indexed blocks (the *good* blocks) are subrectangles that extend to the full width of the strip and as far up as they can while still constituting a strip-rule pattern. Each good block that does not extend all the way up to the top boundary of P is followed by a *bad* block that is a height-1 subrectangle immediately above the good block, extending the full width of the strip. (Note that such rectangles are trivially strip-rule patterns.) Let $\mathcal{R}_q$ be an RRL constructed by using MPUS to handle each of the blocks separately. (Strips for a block will typically be rectangles in terms of the overall pattern.) Our output is the best of the original RRL and the N_{col} constructed RRLs.

THEOREM 6.1. *The iterated strip-rule algorithm runs in polynomial time and produces an RRL whose length is* $O(\mathrm{OPT}(P) \min(n^{1/3}, \mathrm{OPT}^{1/2}))$. *Moreover, the bound is tight in that there exist patterns with arbitrarily large values of* $\mathrm{OPT}(P)$ *for which the algorithm produces RRLs whose lengths are* $\Theta\left(\mathrm{OPT}(P)^{3/2}\right)$.

Key to the proof are the observations that (1) the RRL created by MPUS for a $p \times q$ rectangle uses at most $p + q$ rules, (2) the union of each bad block with its preceding good block contains either a 2×2 or 3×3 subarray with no monochromatic columns or rows (an *obstacle subarray* in what follows), and (3) the following lower bound lemma.

Denote the grid cells by $P_{i,j}$, $1 \leq i \leq N_{col}$ and $1 \leq j \leq N_{row}$. If $1 \leq h \leq i \leq N_{col}$ and $1 \leq j \leq k \leq N_{row}$, let $R(h, i, j, k)$ be the subrectangle of the grid formed by the intersection of columns h through i with rows j through k. Such a subrectangle R is a *forbidden rectangle* if it contains an obstacle subarray that has the same leftmost and rightmost columns and the same top and bottom rows as R.

LEMMA 6.2. *Suppose there are r disjoint forbidden rectangles. Then $OPT(P) \geq r/4$.*

Proof. Let $\mathcal{R}$ be a collection of r disjoint forbidden rectangles, and let E be any rectangle in $\mathcal{R}$. Note that by definition of forbidden rectangle, all the bounding columns/rows of E (the top and bottom rows and the leftmost and rightmost columns) are non-monochromatic in P. Let A be an optimal RRL for P, which we shall view in pick-up-sticks order. The first rule in A to include a corner of E can include only one of its four corners, as otherwise a bounding column/row of E would be monochromatic in P. This means that the rectangle corresponding to the rule must have one of its corners in E. Since all the rectangles in $\mathcal{R}$ are disjoint

and nonmonochromatic, every one must be intersected by at least one rule. Since no rule has more than four corners, we conclude that $r \leq 4|A| = 4 \cdot OPT(P)$, and the claim follows. ■

The rest of the proof of Theorem 6.1, including lower bound examples, will be given in the full paper. To get an approximation algorithm for ACLs, simply replace each rectangle in the RRL solution by the minimum set of ACL rules needed to generate it, yielding an $O(w^2 \mathrm{OPT}(P) \min(n^{1/3}, \mathrm{OPT}^{1/2}))$ worst-case bound.

7 Additional Results and Open Problems

In much of this paper we have concentrated on strip-rule RRLs and ACLs for strip-rule patterns. However, in some applications one might be able to use arbitrary RRLs/ACLs for strip-rule patterns, in which case the performance penalties imposed by the restriction to strip-rule RRLs/ACLs is relevant. Fortunately the penalty is not too severe, at least in the case of RRLs.

THEOREM 7.1. *For RRLs and for any strip-rule pattern P, $\mathrm{OPT}_S(P) \leq 4 \cdot \mathrm{OPT}(P) + 1$.*

This follows from the fact that no rectangle rule can introduce more than 4 effective grid lines and every strip-rule must eliminate at least 1, with the possibility of one final rule to color any cells remaining after all effective grid lines have been eliminated. The asymptotic bound is at least 3.5, based on a sequence of patterns P_k to be presented in the full paper that have $\mathrm{OPT}(P_k) = 2k+2$ and $\mathrm{OPT}_S(P_k) = 7k+6$. It would be interesting to close this gap, as well as that for ACLs, where we get upper and lower bounds proportional to the IP address length w.

At present, however, the main open problem concerns the complexity of general ACL minimization. Our NP-completeness result for RRL minimization does not preclude the fact that ACL minimization might be in P. Also open is the question of whether either of these problems is MaxSNP-hard. (Our NP-completeness transformation for RRLs destroys the MaxSNP-hardness of the source problem.)

Another open problem is that of improving on the approximation algorithms of the previous section for the case of arbitrary patterns, both for RRLs and ACLs. We can prove that the greedy heuristic that repeatedly picks up the pseudo-monochromatic rectangle with the most non-gray cells has worst-case examples that are at least as bad as those for ISR. Are polylogarithmic guarantees possible in polynomial time, as they are for the black-rule-only case? Are additional $\Omega(w)$ factors necessary in the guarantees for ACLs?

Finally, can our strip-rule optimization algorithms be extended to more than 2 colors, where for ACLs the extra colors might correspond to different quality of service guarantees?

References

[AR99] V. S. Anil Kumar and H. Ramesh. Covering rectilinear polygons with axis-parallel rectangles. In *Proc. 31st Ann. ACM Symposium on Theory of Computing*, pages 445–454, New York, NY, 1999. ACM.

[BD97] P. Berman and B. DasGupta. Complexities of efficient solutions of rectilinear polygon cover problems. *Algorithmica*, 17:331–356, 1997.

[Cis01] Cisco Systems. Controlling network access with access control lists (Chapter 10). In *Catalyst 6500 Series Switch and Cisco 7600 Series Firewall Services Module Configuration Guide*, 2001. Available at `http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/mod_icn/fwsm/fwsm_2_2/fwsm_cfg?mngacl.pdf`.

[CL05] E. Cohen and C. Lund. Packet classification in large ISPs: Design and evaluation of decision tree classifiers. In *Proc. ACM Sigmetrics '05*, pages 73–84, 2005.

[DKVZ99] R. Daves, C. King, S. Venkatachary, and B. Zill. Constructing optimal IP routing tables. In *Proc. IEEE INFOCOM 1999*, pages 88–97, 1999.

[EM01] David Eppstein and S. Muthukrishnan. Internet packet filter management and rectangle geometry. In *Proc. 12th Ann. ACM-SIAM Symp. of Discrete Algorithms*, pages 827–835, Philadelphia, PA, 2001. SIAM.

[Ful05] E. W. Fulp. Optimization of network firewall policies using directed acyclic graphs. In *Proc. IEEE Internet Management Conf.*, 2005.

[GM99] P. Gupta and N. McKeown. Packet classification on multiple fields. In *Proc. Conf. on Applications, Technologies, Architectures, and Protocols for Comp. Communication* (SIGCOMM), pages 147–160, New York, NY, 1999. ACM.

[KMT03] H. Kaplan, E. Molad, and R. E. Tarjan. Dynamic rectangular intersection with priorities. In *Proc. 35th Ann. ACM Symposium on Theory of Computing*, pages 639–648, New York, NY, 2003. ACM.

[LNW+02] L. V. S. Lakshmanan, R. T. Ng, C. X. Wang, X. Zhou, and T. J. Johnson. The generalized MDL approach for summarization. In *Proc. Int. Conf. on Very Large Databases* (VLDB'02), pages 766–777, 2002.

[Mas78] W. J. Masek. Some NP-complete set covering problems. Unpublished manuscript, 1978.

[SSW03] S. Suri, T. Sandholm, and P. Warkhede. Compressing two-dimensional routing tables. *Algorithmica*, 35:287–300, 2003.

[TC03] June 2003. `TopCoder` Match Summary, Single Round Match 150, `http://www.topcoder.com/index?t=statistics&c=srm150_prob`.

[Tho03] M. Thorup. Space efficient dynamic stabbing with fast queries. In *Proc. 35th Ann. ACM Symposium on Theory of Computing*, pages 649–658, New York, NY, 2003. ACM.