

ROUGH DRAFT of Full Paper

Compressing Rectilinear Pictures and Minimizing Access Control Lists

DAVID L. APPLGATE^{*} GRUIA CALINESCU[†] DAVID S. JOHNSON[‡]
HOWARD KARLOFF[§] KATRINA LIGETT[¶] JIA WANG^{||}

January 24, 2007

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 $O(\min(n^{1/3}, \text{OPT}^{1/2}))$ -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

^{*}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.

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.

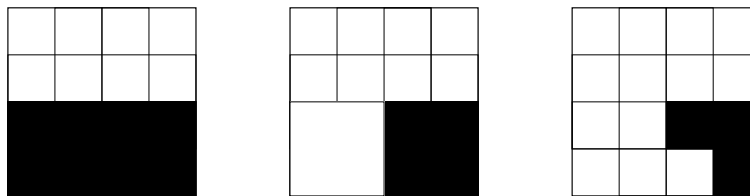


Figure 1: Creating a figure using rectangle rule applications.

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 $(R_1, c_1), (R_2, c_2), \dots, (R_n, c_n)$ 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 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 (rows and columns 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. Moreover, even though there are only two colors in the above application, one can imagine future applications in which there are many colors, with each color corresponding to a desired "quality of service" level for the matched packet.

There are two distinctions between ACLs and RRLs, however, 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 $0 \pmod{2^k}$. 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. In what follows, we shall call such rectangles the *legal* ACL rectangles.

1.3 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

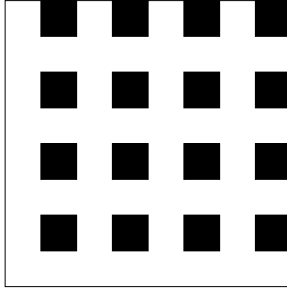


Figure 2: Pattern for which there is a linear factor performance penalty for restricting to RRLs/ACLs with “conflict-free” rules or with only black rules.

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.

Dynamic programming can also be applied to the special case of the 2-dimensional ACL problem in which one requires that the ACL be “conflict-free” in the sense that the rectangles themselves, rather than just their projections, must be laminar. This special case was introduced and studied in [SSW03], which shows that a minimum length conflict-free ACL can be found in time $O(KNw^2)$. Unfortunately, this restriction is not typically obeyed by real-world ACLs. In addition, it imposes a significant performance penalty. Consider a $n \times n$ grid, where $n = 2^w$, in which a cell is black if both of its coordinates are odd and otherwise is white. Figure 2 illustrates such a pattern for the $w = 3, n = 8$ case. An optimum ACL for this pattern has a rule for each of the $n/2$ white rows and $n/2$ white columns, followed by a black rule covering the entire grid, for a total of $n + 1$ rules. The minimum conflict-free ACL for this pattern, however, has a separate rule for each black cell, for a total of $n^2/4 = \Theta(\text{OPT}^2)$ rules.

The 2-dimensional special case of RRL/ACL minimization that has received the most study to date is that in which only black rectangle rules are allowed. For RRLs, this 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 also be a very poor approximation to the optimal black-and-white one. The examples illustrated by Figure 2 suffice to show that this restriction too can cause the best RRL/ACL to increase by a factor of $\Theta(\text{OPT})$.

1.4 Our Results

None of the abovementioned papers considered the general RRL/ACL minimization problem, a decision we can justify in retrospect, since one of our results is that the general RRL minimization problem is NP-hard. (The complexity of general ACL minimization remains open.) In this paper we introduce and study a new restriction on RRLs/ACLs, and then exploit our results about it to derive approximation algorithms for the general problem.

The new restriction is motivated by our study of real-world ACLs at AT&T. In examining a sample of thousands of extended ACLs from AT&T’s access routers, we observed that 95.3% 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.”

Not all patterns are strip-rule patterns, although the set of patterns generated by strip-rule ACLs is identical to the set of patterns generated by strip-rule RRLs. (The minimum-length strip-rule ACL for generating a given strip-rule pattern may be significantly longer than the minimum strip-rule RRL, however.) In Section 2 we characterize the patterns that can be constructed with strip-rules. In Section 3 we present efficient algorithms for determining whether a pattern P is a strip-rule pattern, depending on the initial representation of P , be it by an RRL/ACL that generates it or by what we call an “effective grid” representation. In Section 4 we observe that our recognition algorithms generate RRLs as a byproduct, and determine their worst-case behavior when they are viewed as approximation algorithms for the problem of generating minimum-length strip-rule RRLs/ACLs. In the RRL case the asymptotic worst-case ratio is 2, and we show how this can be improved to a ratio of 1.5 while still staying within an $O(n^2)$ running time bound, when the pattern is given to us by an RRL of length n . Exploiting insights gained from studying these algorithms, we derive $O(n^3)$ algorithms for constructing *optimal* strip-rule RRLs and ACLs for 2-color strip-rule patterns in Section 5.

The remainder of the paper presents our results for unrestricted RRLs/ACLs. In Section 6 we consider how much we could save by using arbitrary RRLs/ACLs to generate strip-rule patterns, for example showing that in the RRL case, the restriction to strip-rule RRLs can cost us as much as an asymptotic factor of 3.5 but at most a factor of 4. We also bound the penalties for using ACLs rather than RRLs to generate strip-rule or arbitrary patterns. In Section 7 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 $O(\min(n^{1/3}, \text{OPT}^{1/2}))$ and $O(w^2 \min(n^{1/3}, \text{OPT}^{1/2}))$ respectively. These are currently the best guarantees known to be attainable in polynomial time. Finally, in Section 8 we present our NP-hardness proof for the problem of RRL minimization for general 2-color patterns. The proof exploits our insights into the strip-rule special case. We conclude in Section 9 by mentioning some additional results and open problems.

All the results presented here trivially extend to the case where the rule lists must specify the color of the entire canvas/grid, i.e., when the background color does not come for free. We leave the appropriate adaptations to interested readers.

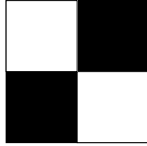


Figure 3: Pattern that cannot be generated using only strip-rules.

2 Strip-Rule Patterns: Characterization Theorems

Not all patterns are strip-rule patterns; for example, the simple 4-cell checkerboard pattern depicted in Figure 3, with the lower left and upper right cells black and the other two white, is not a strip-rule pattern. 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 2.1 *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”). See Figure 4 for an illustration.*

Proof. Let C_1 and C_2 be the two colors. We proceed by showing that (1) and (5) are equivalent, that (2) and (3) are equivalent, that (2) and (4) are equivalent, and that (1) implies (2) and (3) implies (5).

To see that (1) and (5) are equivalent, we first show that (1) implies (5). Suppose P can be constructed using a strip-rule ACL $A = (r_1, r_2, \dots, r_m)$. Then the rows/columns to which r_1 refers are monochromatic in P , and so long as these have not all been picked up by the Pick-Up-Sticks

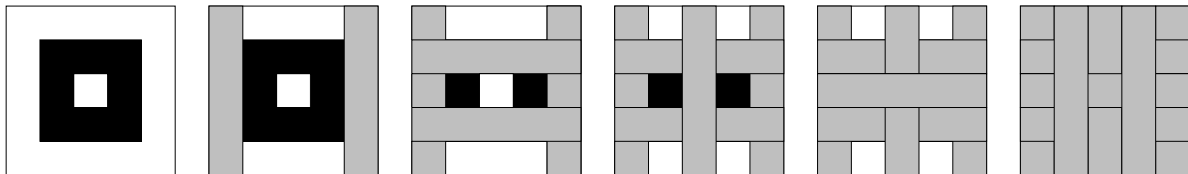


Figure 4: The Pick-Up-Sticks algorithm in action on a strip-rule pattern.

1	3	1
3	3	2
1	2	2

Figure 5: A non-strip-rule pattern for which every 2×2 subarray has a monochromatic column or row.

algorithm, there will be a move available to it. Inductively, suppose r_i is the lowest-index rule from A that still has an unpicked row/column. If there is no such r_i then we must have picked up everything, and the Pick-Up-Sticks algorithm has succeeded. Otherwise, any of the unpicked columns to which r_i refers must be pseudo-monochromatic in the remaining pattern and so the Pick-Up-Sticks algorithm can proceed. Consequently, it never gets stuck until the empty grid is reached as claimed. Conversely, if the Pick-Up-Sticks algorithm reaches an empty grid, the sequence of rows/columns that it picked up constitute an ACL for the original pattern. Thus (1) and (5) are equivalent.

To see that (2) and (3) are equivalent, we first show that (2) implies (3). Suppose every 2×2 subarray of P contains a monochromatic row or column. Consider two non-identical rows of P , R_i and R_j . One of these two, say R_i , must have a black cell in a column C_h where the other has a white cell. The two rows will have the monotonicity property unless there is some column C_k where R_j has a black cell and R_i has a white cell. But then these two rows and columns yield a checkerboard subpattern, a contradiction. On the other hand, suppose P does have a checkerboard subpattern M and suppose M is the intersection of rows i and j with columns h and k . Let $c(a, b)$ denote the color of the cell in column b of row a , and assume without loss of generality that $c(i, h) = C_1$. Then $c(i, k) = C_2$, $c(j, h) = C_2$, and $c(j, k) = C_1$. Consequently the set of columns containing C_1 in row i is neither a subset nor superset of the set of columns containing C_1 in row j , so property (3) is violated. The equivalence of (2) and (4) follows analogously.

To see that (1) implies (2) we argue by contradiction. Suppose P has a strip-rule ACL A and contains a 2×2 subarray M with no monochromatic row or column. Suppose M is the intersection of rows i and j with columns h and k . Let r be the first rule in A to include one of these rows and columns. Then the two cells of M in that row/column must get the same color in the final pattern, contradicting the assumption that M had no monochromatic row or column.

Finally, to see that (3) implies (5), we also argue by contradiction. Suppose (3) holds but that the Pick-Up-Sticks algorithm does not result in an empty grid. Then it must get stuck at some subarray M without any monochromatic row or column. Let i be a row of M with the maximum number of cells colored C_1 . Since this row is not monochromatic, there must be some column k in M such that $c(i, k) = C_2$. We claim that column k must be monochromatic in M . Suppose not. Then some row j in M must have $c(j, k) = C_1$. But then by (3) row j must have color C_1 in all the columns that row i did, so h has more cells colored C_1 than does row i , a contradiction. ■

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 5, 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.

Theorem 2.2 *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.

Proof. We have already observed that (1) and (3) are equivalent, and that argument did not rely on the number of colors. The same basic argument as before also shows that (1) implies (2), since the first rule in an ACL to affect a subarray forces the affected rows/columns of the subarray to all get the same color in the final pattern, and the forbidden subarrays have no monochromatic rows or columns.

The non-trivial challenge is to show that (2) implies (3). So suppose that the Pick-Up-Sticks algorithm gets stuck before it can empty the grid. Then P must contain a subarray with no monochromatic rows or columns. Let M be a minimal such subarray, in that every proper subarray of M has either a monochromatic row or column (or both). We shall argue by contradiction that M must be a 2×2 or 3×3 subarray. Suppose it contains no such subarray.

Let r and c be the number of rows and columns in M . Suppose first that $\max(r, c) > 3$, and assume without loss of generality that $r > 3$. Let M_i be the subarrays obtained from M by deleting its i th row, $1 \leq i \leq r$. Each of these must have a monochromatic row or column by hypothesis. They cannot have monochromatic rows, since their rows are all rows of M , so each must have a monochromatic column. No two can have the same monochromatic column, as this would imply that the column must get the same color in each and that the column would be monochromatic in M as well. Furthermore, no two can get the same color. For suppose two did. Say M_i and M_j , $i \neq j$, both have monochromatic columns with color C . Suppose that M_i 's monochromatic column is h and M_j 's monochromatic column is k . We have already observed that we must have $h \neq k$, so consider the 2×2 subarray determined by rows i and j and columns h and k . Row i is present in M_j and row j is present in M_i , so we have $c(j, h) = c(i, k) = C$. However, since M does not contain any monochromatic column itself we must have $c(i, h) \neq C$ and $c(j, k) \neq C$. Thus this 2×2 subarray has no monochromatic row or column, a contradiction. Thus each of the M_i must have a different monochromatic column (implying $c \geq r > 3$) and each monochromatic column has a different color.

Consider an $r \times 3$ subarray of M' of M obtained by deleting all the columns except for three of these monochromatic columns. This is a proper subarray of M since as we have observed we must have $c \geq r > 3$. It has no monochromatic columns because M had none. Moreover, if we let C_1, C_2 , and C_3 be the three distinct colors associated with the columns of M' , then any row of M' must contain at least two of these colors, as row i can only omit the color associated with M_i . Thus M' has no monochromatic row or column and yet is a proper subarray of M , a contradiction.

Thus we must have $\max(r, c) \leq 3$. The only remaining case to rule out is the case where M is a 3×2 or 2×3 subarray. Assume without loss of generality that we have the first case, and construct the subarrays M_i , $1 \leq i \leq r = 3$, as before. Each must have a monochromatic column, and since there are only two columns available, two must share a column, and hence a color, and hence M must have a monochromatic column, a final contradiction. ■

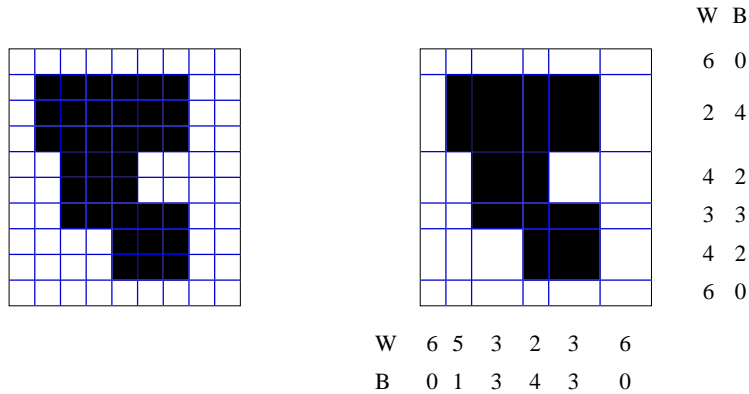


Figure 6: An uncompressed pattern and its effective grid, the latter augmented by its profile.

3 Recognizing Strip-Rule Patterns

The forbidden subarray characterizations in Theorems 2.1 and 2.2 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 all-gray pattern. In this section we study how fast we can simulate the action of Pick-Up-Sticks, depending on the way in which the pattern is represented.

We consider five different input formats. Assume that we are considering patterns on a $H \times W$ grid, and that colors are represented by integers from 1 to k . The *grid lines* of the pattern are the boundaries between consecutive rows or consecutive columns of the matrix. Say such a grid line is an *effective grid line* if the rows (columns) that it bounds are not identical. The external boundaries of the pattern are “effective grid lines” by convention. The five input formats are as follows.

1. The uncompressed $H \times W$ pattern (HW integers from 1 to k).
2. The effective $R \times C$ grid — sorted lists of the effective horizontal and effective vertical grid lines, and for each pair of consecutive horizontal and consecutive vertical effective grid lines, the color associated with the rectangle (*effective grid cell*) they collectively bound. (The coordinates of the effective grid lines in the original uncompressed pattern are also stored.)
3. An arbitrary RRL/ACL that generates the pattern (with n rules).
4. For strip-rule patterns, a strip-rule RRL/ACL that generates the pattern (with n rules).
5. A *profile* for the pattern which stores the coordinates of the effective grid lines and contains, for each color x in the pattern, the following information:
 - For each row i of the effective grid the number $a_x[i]$ of effective grid cells in the row that have color x .
 - For each column j of the effective grid, the number $b_x[j]$ of effective grid cells in the column that have color x .

See Figure 6 for an illustration of the uncompressed pattern, its effective grid, and its profile. Note that for profiles of 2-color patterns, the list of counts for color white are redundant, since

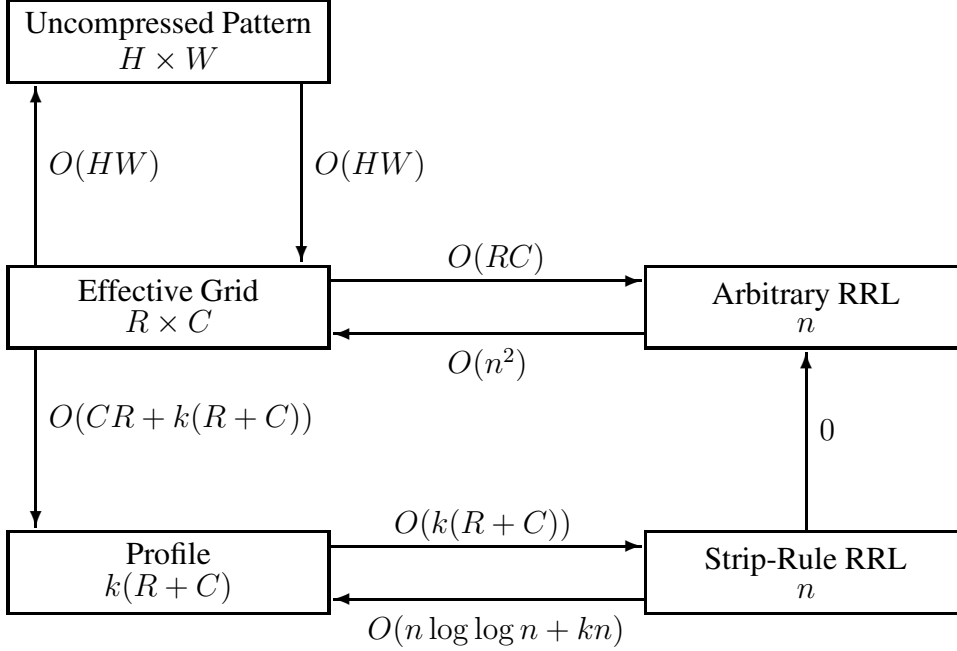


Figure 7: Times to convert between input formats as functions of the relevant parameters, where k is the number of colors.

the row count for white is always the total number of effective rows minus the corresponding row count for black, and similarly for the column counts. In general, the profile representation does not contain enough information to completely specify the pattern — more than one pattern can have the same profile, as can be confirmed by a simple counting argument. This is not true, however, for strip-rule Patterns.

Theorem 3.1 *Let P_1 and P_2 be patterns with $R \times C$ effective grids, and suppose P_1 is a strip-rule pattern. Then if P_1 and P_2 have the same profile the two patterns have the same effective grid.*

Proof. The proof is by induction. The theorem is clearly true if $C = R = 1$. Suppose it is true for all dimension pairs (C, R') with $R' < R$ and all pairs (C', R) with $C' < C$. Since P_1 is a strip-rule pattern, it must contain a monochromatic row or column by Theorem 2.2. Suppose without loss of generality it contains a monochromatic column, and that column is column i and has color x . Then the profile for P_1 says that column i has R cells of color x , which implies that column i of P_2 is also monochromatic and has color x . Consider the two $R \times (C - 1)$ patterns P'_1 and P'_2 obtained by deleting column i from P_1 and P_2 . These have the same profile (the original profile with the entries for column i deleted and the row counts for color x all reduced by 1). By induction, they must be identical. Hence so are P_1 and P_2 . ■

As we shall see, the profile is a key input format since it is the most compact and can be used to implement the Pick-Up-Sticks algorithm in $O(R + C) = o(RC)$ time for any fixed number of colors. Before going into these details, however, let us first consider the computational effort required to go between the various input formats. See Figure 7 which summarizes the running times for the currently best known conversion algorithms in the case of RRLs. (Missing arrows correspond to cases where the best conversion algorithms we know go through other formats as intermediaries.) The algorithm for generating strip-rule RRLs only does so for strip-rule patterns (otherwise reporting that the pattern is not a strip-rule pattern).

Five of the eight conversions depicted in the figure are straightforward. Note first that it is straightforward to go back and forth between the uncompressed pattern and the effective grid in time $O(HW)$, i.e., time linear in the size of the former. Thus we shall ignore the uncompressed pattern in what follows. In another straightforward conversion, one can obtain an (unrestricted) RRL for a pattern from its effective grid in time proportional to the size of the latter, simply by giving a separate rule for each effective grid cell. Also straightforward are the process of going from the effective grid to the profile and the vacuous operation of converting a strip-rule RRL to an (unrestricted) RRL. This leaves three somewhat nontrivial conversions.

Theorem 3.2 *Suppose P is a pattern with k colors.*

- (A) *Given an (unrestricted) RRL of length n for P , the effective grid for P can be constructed in time $O(n^2)$.*
- (B) *Given a strip-rule RRL of length n for P , the profile for P can be constructed in time $O(n \log \log n + k(R + C))$, where R and C are the lengths of the row and column lists in the profile.*
- (C) *Given the profile for P with row lists of length R and column lists of length C , one can determine whether P is a strip-rule pattern and if so generate a Maximal Pick-Up-Sticks strip-rule RRL for it in time $O(k(C + R))$.*

Proof. (A) **RRL for $P \rightarrow$ Effective grid for P .**

Let L be the RRL. We begin by creating sorted lists of the left and right rectangle boundaries and of the top and bottom rectangle boundaries for all the rules L . These provide a (not-necessarily-proper) superset of the effective grid lines of P , and from them we can derive a refinement of the effective grid, call it G . The overall time to construct G is clearly $O(n^2)$.

Next we color the cells of G . Initially label all the cells as *uncolored*. We now process the rules of L in order. Suppose the current rule applies to the rectangle determined by the intersection of columns h through i with the rows j through k in G , and applies color x . We process the rule as follows.

1. For each column c , $h \leq c \leq i$, start with row index $r = j$.
2. **while** $r \leq k$, do the following:
 - (2.1) If grid cell $G_{r,c}$ is uncolored, color it x .
 - (2.2) Set $r^* = \min\{r' > r : G_{r',c} \text{ is uncolored}\}$, with $r^* = n + 1$ if there are no uncolored cells below $G_{r,c}$, and then set $r = r^*$.

At the end all uncolored cells are given the default color. It should be clear that this procedure will properly color G . The overall time to do this is $O(n^2)$. It consists of at most n column starts for each of n rules, plus at most n^2 coloring operations, plus the time to perform all of the at most $2n^2$ “next uncolored cell” operations (at most one for each newly colored cell plus at most one that yields $r^* > k$ for each combination of rule and column). Each of these can be accomplished in amortized constant time using the linear-time union-find approach described in [GT85].

Finally, we need to derive the colored effective grid from G , but this can be done in time $O(n^2)$ by comparing consecutive rows and columns and deleting the grid line between them if the two are identical.

(B) Strip-rule RRL for $P \rightarrow$ Profile for P .

Note that in this case P is a strip-rule pattern. We first sort the top and bottom endpoints of the row rules in the RRL and the left and right endpoints of the column rules to get sorted lists of a superset of the effective vertical and horizontal grid lines. Using the fast deterministic integer sorting algorithm of [Han02] this can be done in $O(n \log \log n)$.

We next will construct a profile on the set of rows and columns thus defined. To get the true profile, we will then have to merge the results for adjacent rows and columns that are identical and delete the grid lines that separate them. Fortunately, the identity of two rows/columns can be tested in $O(k)$ time given the initial profile, where k is the number of colors, rather than $\Theta(n)$ time, yielding a total cost of $O(kn)$ for this postprocessing. This is a consequence of the 2-color strip-rule characterization theorem (Theorem 2.1), which implies that for strip-rule patterns each color x must satisfy the monotonicity property for both rows and columns. (Simply consider the 2-coloring of P derived by replacing all colors other than x by a single new color y .) Thus if two rows have the same cell counts for all k colors, they must be identical.

So we need only show that how to construct a profile on the rows and columns determined by the boundaries of the rules in the RRL. Without loss of generality, all we need show is that this can be done in time $O(n)$ for rows and a specified color x . Suppose we have R rows and C columns. We start with an array of counts $a_x[i]$, $1 \leq i \leq R$, each initially 0, together with variables u_i^{row} , $1 \leq i \leq R$, and u_j^{col} , $1 \leq j \leq C$, all initially 0 as well, where $u_i^{row} = 0$ means that no row rule covering row i has yet been processed and $u_j^{col} = 0$ means that no column rule covering column j has yet been processed. In addition we have two additional parameters, an *uncolored* count $U = C$, and a *x-colored* count $T = 0$.

We process the rules of the RRL in “pick-up-sticks” (ACL) order, where the color of a grid cell in P is that of the *first* rule that contains it. Column rules are processed as follows. Suppose the rule covers k columns starting with column i . Once again using the linear-time union-find approach described in [GT85] we identify all the columns h , $i \leq h < i + k$, such that $u_h^{col} = 0$, processing each in turn. For each such column h , the processing consists of setting $u_h^{col} = 1$, $U = U - 1$, and, if the color for the rule is x , setting $T = T + 1$. The total time for such column processing will be $O(n + C) = O(n)$. The row rules are processed as follows. Again using the linear-time union-find approach described in [GT85] we identify all the rows h , $j \leq h < j + k$, such that $u_h^{row} = 0$, processing each in turn. First we set $u_h^{row} = 1$ and then we assign a value to $a_x[h]$. If the color of the rule is x , we set $a_x[h] = U + T$. Otherwise, we simply set $a_x[h] = T$. The total time for such row processing will be $O(n + R) = O(n)$. After all rules have been processed, we finish by processing all the rows j for which u_h^{row} is still 0. If x is the default color, we set $a_x[j] = U + T$. Otherwise, we set $a_x[j] = T$. The time for this last step is $O(n)$, yielding an overall time of $O(n)$ for rows and color x and $O(kn)$ for both rows and columns and all k colors.

Combining this with the $O(n \log \log n)$ time for the initial sorting, we obtain the claimed overall time bound.

(C) Profile for $P \rightarrow$ Strip-Rule RRL for P .

The profile representation was designed to assist in the implementation of Pick-Up-Sticks. If we let R be the number of rows and C be the number of columns, note that the monochromatic rows in the initial pattern are simply those with indices i such that $a_x[i] = C$ for some color x , and the monochromatic columns are those with indices j such that $b_y[j] = R$ for color y . Our construction process will generate the strip-rules in Pick-Up-Sticks order by essentially simulating

the operation of the Pick-Up-Sticks algorithm while maintaining enough information to always be able to identify the current pseudo-monochromatic rows and columns. (Note that when we are dealing with the effective grid, as we do implicitly when we deal with a profile, a Pick-Up-Sticks move, which picks up a single row or column of the effective grid, may actually be picking up a sequence of contiguous rows/columns in the original, uncompressed pattern.)

We maintain the following collection of auxiliary data structures. For each color x and each integer j , $1 \leq j \leq C$, we have a list $L_{x,j}^{row}$ of those rows i such that $a_x[i] = j$, i.e. that have exactly j cells of color x in the effective grid representation of the initial pattern P , from bottom to top. We also have a pointer $p_{x,j}^{row}$ to the first member of the list that has not yet been picked up, initially pointing to the first element of the list (or to NULL if the list is empty.) Similarly, for each color x , $1 \leq x \leq k$, and each integer i , $1 \leq i \leq R$, we have a list $L_{x,i}^{col}$ of those columns j such that $b_x[j] = i$, from left to right, and a pointer $p_{x,i}^{col}$ to the first element in the list. In addition, for each color x , we have variables w_x^{row} and w_x^{col} , initially 0, that represent the number of rows/columns that have been picked up so far and with color x (i.e. when all its non-gray cells had color x). Finally, we have variables u^{row} and u^{col} , initially set to R and C respectively, that count the number of rows/columns that have not yet been picked up, and an initially empty rule list L , containing the strip-rules for each of the rows/columns so far picked up, in the order in which the pick-ups occurred. All the above variables can be initialized from the profile in cumulative $O(k(C + R))$ time.

Inductively, suppose we are at a point in the process when all the above variables have their correct values, as they do initially. Consider row i . If it has not yet been picked up by a row rule, it will have u^{col} non-gray cells. On the other hand, for any color x , it will have $a_x[i] - w_x^{col}$ cells of color x . Thus if for any color x we have $a_x[i] - w_x^{col} > u^{col}$, the row must already have been picked up. Similarly, if it has not yet been picked up, the row will be pseudo-monochromatic if and only if there is a color x such that $a_x[i] - w_x^{col} = u^{col}$. Analogously, a column j that has not yet been picked up must have $b_y[j] - w_r[x] \leq u_r$ for all colors y and will be pseudo-monochromatic if and only if there is a color y such that $b_y[j] - w_x^{row} = u^{row}$. By Theorem 2.1, if P is a strip-rule pattern, there always must be at least one pseudo-monochromatic row or column. If such a row exists, then for some color x , the pointer $p_{x,u^{col}-w_x^{col}}^{row}$ is not NULL. If such a column exists, there for some color y the pointer $p_{y,u^{row}-w_x^{row}}^{col}$ is not NULL.

There may be more than one such non-NULL pointer, so choose one. Suppose it the pointer is for color x and points to to row i . Then row i is pseudo-monochromatic and all its non-gray cells have color x . Add the rule (A, x) to the end of L , where A is the horizontal strip containing the column i (or the contiguous sequence of rows from the original, uncompressed pattern that were merged into row i of the effective grid, if the two differ). Then set $u^{row} = u^{row} - 1$ and $w_x^{row} = w_x^{row} + 1$. Similarly, if pseudo-monochromatic column j is chosen and y is the color of its non-gray cells, add the rule (B, y) to the end of L , where B is the vertical strip containing the column i (or the contiguous sequence of columns from the original, uncompressed pattern that were merged into column j of the effective grid, if the two differ). Then set $u^{col} = u^{col} - 1$ and $w_x^{col} = w_x^{col} + 1$. In both, we complete the processing of the move by advancing the relevant pointer to the next element in the list, setting it to NULL if this was the last element in its list.

It is not difficult to see that the inductive hypothesis is preserved by these actions. Moreover, the total time required to find and process the picked-up row/column is $O(k)$. If we ever reach a point in the process where u^{col} and u^{row} are both still non-zero but all the relevant pointers are NULL, the original pattern P was not a strip-rule pattern and we report this fact. Otherwise, we

eventually, we reach a state where either u_c or u_r is 0, which implies that the residual pattern is all-gray and hence L will be our desired RRL. Since no row or column can be picked up more than once, the total time for this processing is $O(k(R + C))$, the same bound we had for the preprocessing phase, and hence the overall running time bound as well. ■

Corollary 3.3 *Given a pattern P represented by a length- n , unrestricted RRL, one can in time $O(n^2)$ time determine if P is a strip-rule pattern and, if so, produces a strip-rule RRL for it. If P is not a strip-rule pattern, one can in $O(kn^2)$ time find one of the 2×2 or 3×3 forbidden subarrays guaranteed by Theorem 2.2.*

Proof. In order to perform the strip-rule test, we first use Theorem 3.2(A) to generate the effective grid for P from the given ACL in time $O(n^2)$. Note that the effective grid will contain at most $2n$ effective row grid lines and $2n$ effective columns grid lines since each rectangle rule can introduce at most two of each type (one each for the bottom, top, left, and right boundaries of the rectangle). Thus the profile of P , which we shall denote by $\text{prof}(P)$, can be constructed in $O(n^2)$ time as well. Finally, we apply Theorem 3.2(C) to generate the strip-rule RRL if it exists in time $O(kn)$ which is also $O(n^2)$ since our original n rules can have introduced at most n colors.

To find a forbidden subgraph in the case where P is not a strip-rule pattern, we assume we have already constructed $\text{prof}(P)$ and proceed as follows. We introduce a new temporary pattern P' which will agree with P except that some of its rows and columns will be colored a new color “*”. Initially set $P' = P$ and hence $\text{prof}(P') = \text{prof}(P)$. Now for each row and column (in any order) we do the following:

1. Let P'' be P with all the cells in the chosen row/column colored *, and construct $\text{prof}(P'')$ from $\text{prof}(P')$. (This will take $O(n)$ time since we only have to examine the at most $2n + 1$ cells of the row/column and adjust that many counts in the profile.)
2. Run the profile \rightarrow RRL algorithm on $\text{prof}(P'')$. (This will take $O(kn)$ time.)
3. If P'' turns out to be a strip-rule pattern, set $P' = P''$ and $\text{prof}(P') = \text{prof}(P'')$. Otherwise, continue.

When all the rows/columns have been processed, output the set of rows/columns that were *not* permanently colored *.

The overall time for this procedure is clearly $O(kn^2)$ as desired. To see that the output set of rows and columns specify the desired forbidden subgraph, note first when the procedure ends it still must be the case that P' is not a strip-rule pattern. Thus P' must contain a 2×2 or 3×3 forbidden subarray by Theorem 2.2. Let S be such a subarray. When any row/column not intersecting S was processed, coloring all of its cells * would not have affected S , so the resulting pattern would still have been a non-strip-rule pattern. Thus the coloring of the row/column would have been made permanent. Thus the only rows/columns that were not permanently colored * are those intersecting S , and the output of the procedure is as claimed. ■

Theorem 3.2 and Corollary 3.3 talk about constructing RRLs. If we are given a pattern on a $2^w \times 2^w$ grid we can use these results to produce an ACL that generates the same pattern simply by simulating the RRL rules with ACL rules. In particular, we can exploit the following easily-verified Lemma.

Lemma 3.4 *Suppose we are generating a pattern on a $2^w \times 2^w$ grid. Then*

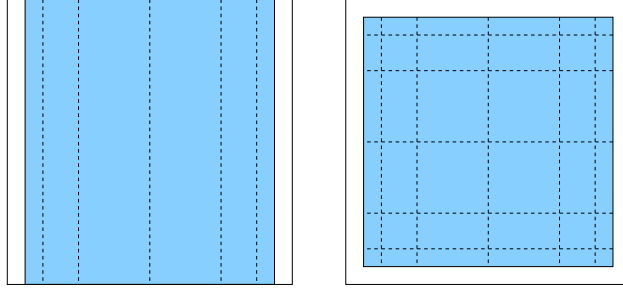


Figure 8: Strip and rectangle rules for $w = 4$ whose simulation by legal ACL rules of the same type require $2(w - 1)$ and $4(w - 1)^2$ rules, respectively.

1. Any strip-rule can be simulated by an equivalent set of $2(w - 1)$ or fewer legal ACL rules.
2. Any rectangle-rule can be simulated by an equivalent set of $4(w - 1)^2$ or fewer legal ACL rules.

Figure 8 illustrates the worst-case simulations. Note that the Lemma implies that a minimum-length strip-rule ACL for a strip-rule pattern on a $2^w \times 2^w$ grid is no longer than $2(w - 1)$ times the length of a minimum-length strip-rule RRL for the pattern. Similarly, a minimum-length ACL for an arbitrary pattern on a $2^w \times 2^w$ grid is no longer than $4(w - 1)^2$ times the length of a minimum-length strip-rule RRL for the pattern. We suspect these bounds are not tight, however, since an optimal ACL may not need to directly simulate all the RRL rules. For example, the patterns in Figure 8 can be produced by ACLs consisting of 3 and 5 strip-rules respectively if we use white as well as black rules. We currently know of no tighter upper bounds than the above, but there is a significant gap between these and our currently best lower bounds on the worst-case penalty for using an ACL rather than an RRL. We will present the latter in Section 6 after we have had more to say about the construction of optimal ACLs.

4 Worst-Case Behavior of Pick-Up-Sticks and its Variants

The description of the Pick-Up-Sticks algorithm in the previous section leaves many choices unspecified. In this section we investigate how the choices affect the quality of the ACLs it produces. In the process, we shall introduce some of the machinery needed to describe the optimization algorithms we present in the next section.

4.1 Maximal Pick-Up-Sticks

A first observation is that, in implementing Pick-Up-Sticks, there can be no harm in always using *maximal* strip-rules, i.e., rules that pick up maximal contiguous “legal” sequences of pseudo-monochromatic rows or columns. In RRLs all sequences are legal, while in ACLs the only legal sequences are those whose members make up a legal ACL rectangle. Replacing a non-maximal strip-rule by the maximal one that contains it does not affect the pseudo-monochromaticity of any later rule. So in what follows, we shall restrict attention to the variant that does this, which we shall call *Maximal Pick-Up-Sticks* (MPUS). Moreover, once that pattern is reduced to one in which all the cells are either gray or the background color, we shall assume that MPUS stops, since a final

rule that grayed out all the cells containing the background color is not required by either of our applications.

We first note that there is not a major running-time penalty for using MPUS. Given a profile for a k -color pattern P whose effective grid has R rows and C columns, we can generate an MPUS ACL in time $O(k(C + R))$, the same bound we had in Corollary 3.3 for a one row/column at a time ACL. The key is to augment our data structures of Theorem 3.2(C) with two doubly-linked lists, the first containing all the as-yet-unpicked-up rows, from bottom to top, and the second containing all the as-yet-unpicked-up columns, from left to right. Whenever a row/column is picked up, it is deleted from the appropriate list, at a total cost of $O(C + R)$. Now when we go to pick up the the current bottommost pseudo-monochromatic row or leftmost pseudo-monochromatic column, we index into the appropriate doubly-linked list and check to see if the next unpicked-up row/column is also pseudo-monochromatic with the same color. If so, we add it to the current rule and keep checking. The total time for constructing the maximal strip-rule is then proportional to the number of as-yet-unpicked-up rows/columns it contains, again for a total cost of $O(C + R)$.

Note that MPUS, as described, is still a nondeterministic algorithm since we do not say how to break ties. For any instance, there must be a way to break ties that yields an RRL/ACL of minimum possible length, but we know of no simple tie-breaking rule that accomplishes this. Fortunately, no tie-breaking rule can be too bad. Let $\text{OPT}_S(P)$ be the length of an optimal strip-rule RRL for pattern P . We will not distinguish in our notation between the RRL and ACL cases since the relevant case will always be clear from context.

Theorem 4.1 *Suppose P is a strip-rule pattern.*

- (A) *Any RRL produced by the Maximal Pick-Up-Sticks RRL algorithm contains at most $2 \cdot \text{OPT}_S(P) + 1$ rules and there exist strip-rule patterns P with arbitrarily large values of $\text{OPT}_S(P)$ such that MPUS, with the appropriate tie-breaking choices, generates RRLs that are that bad.*
- (B) *If P is a pattern on the $2^w \times 2^w$ grid, then any ACL produced by the Maximal Pick-Up-Sticks ACL algorithm contains at most $(w - \lfloor \log(\text{OPT}_S(P)) \rfloor + 2)(\text{OPT}_S(P) - 1)$ rules and for any k , $1 < k < w$, there exist patterns $P_{w,k}$ with $\text{OPT}_S(P_{w,k}) = 2^{k+1}$ such that MPUS, with the appropriate tie-breaking choices, generates ACLs of length $(w - \log(\text{OPT}_S(P_{w,k})) + 2)\text{OPT}_S(P_{w,k}) - 2$.*

Note that this theorem implies that for RRLs the asymptotic worst-case ratio for MPUS is 2 and for ACLs it is w , the length of an IP address, assuming $\text{OPT}_S(P) = o(2^w)$.

Proof. (A) As already observed in the proof of Theorem 3.2, the number of effective grid lines in the effective grid for P can be at most $2 \cdot \text{OPT}_S(P)$. Each maximal strip has two boundaries, and if it is maximal each boundary must either be an effective grid line or one of the four borders of the overall grid. Denote the set of these at most $4 + 2 \cdot \text{OPT}_S(P)$ potential boundary lines by D , and say two members of D are *equivalent* if they are both horizontal or both vertical and if all the rows/columns between them contain only gray cells in the current figure. Initially no two are equivalent as there are no gray cells in the initial pattern P , so the number of equivalence classes is $|D|$. However, each time MPUS makes a move (grays a strip), the borders of that strip become equivalent, reducing the number of equivalence classes by at least one. (The number will be reduced by more than one if the strip contains more than one not-all-gray row/column.)

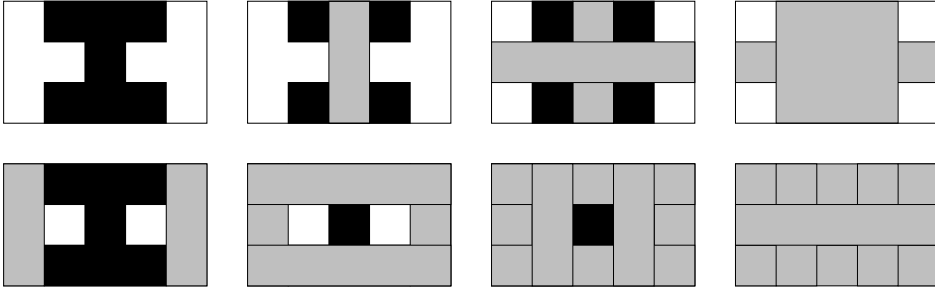


Figure 9: Pattern yielding worst-case behavior for the fixed-strip Maximal Pick-Up-Sticks algorithm. The top row shows the application of an optimal list with 3 rules, while the bottom shows that with different tie-breaking rules the algorithm can generate an RRL with 7 rules.

Suppose that instead of stopping MPUS when all cells are either gray or the background color, we run it until *all* cells are gray. This can only lengthen the resulting RRL. At the end of the process we will have one equivalence class of horizontal borders and one of vertical ones. But note that the first rule to reduce one of these two counts to one will simultaneously reduce the other to one, since after it is applied there will be no non-gray cells. Hence the total number of rules in the RRL generated by MPUS is at most $|D| - 3 \leq 2 \cdot \text{OPT}_S(P) + 1$, as claimed.

To show that the bound is tight, consider the pattern at the top left of Figure 9. If one uses the tie-breaking rule that prefers black columns to white columns and white rows to black rows, then the algorithm will yield 3 rules (the optimal number), as shown in top half of the figure. On the other hand, if one uses the tie-breaking rule that prefers white columns to black columns and black rows to white rows, then the algorithm will yield 7 rules (bottom half of figure). This can be generalized to patterns with arbitrarily long optimal RRLs. Figure 10 illustrates the next two examples in the sequence.

(B) For the ACL case we argue somewhat differently. Now our rules must correspond to legal strips, which can be viewed as corresponding to k -bit IP address prefixes, $0 \leq k \leq w$, where for rows the prefix p corresponds to all rows whose indices has p as a prefix, and similarly for columns. (Recall that the rows and columns are each indexed from 0 to $2^w - 1$.) Let us call two legal strips *mates* if their prefixes are the same length and differ only in their last symbol, so that their union, which we shall call the *parent* of both, is itself a legal strip. For example, the strips corresponding to prefixes 0110 and 0111 are mates, with their union being the strip corresponding to prefix 011. Say that a strip-rule R *contains* another strip rule R' if they are both of the same type (row or column) and all the members (rows or columns) of the strip for R' are members of

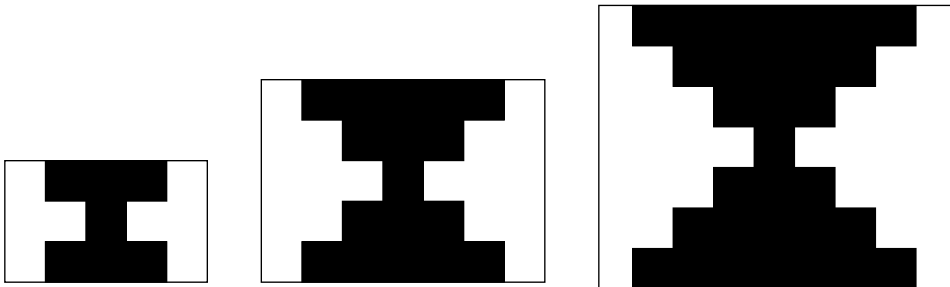


Figure 10: First three patterns in a sequence such that $\text{OPT}(P_k) = 2k + 1$ but Maximal Pick-Up-Sticks can generate RRLs with as many as $4k + 3$ rules.

the strip for R . Equivalently, the prefix for R must be a (not-necessarily-proper) prefix of R' .

Let L be a minimum-length ACL for P . We claim that for every rule R in an MPUS ACL for P , there must be a rule R' in L that is either contained in R or in the mate of R . Suppose not. Then no rule in L differentiates between the rows/columns in R and its mate, and so those rows/columns must all be identical in P . Thus when the rows/columns in R were pseudo-monochromatic during the operation of MPUS, so must have been the rows/columns of its mate, and hence the parent of R also had all its rows/columns pseudo-monochromatic. Hence MPUS, which picks maximal legal moves, could not have picked R , a contradiction.

Let L_M be the set of rules in an MPUS ACL for P . For each rule R in L , let C_R denote the set of all rules that either contain R or are mates of rules that contain R . By the above claim we must have $L_M \subseteq \cup_{R \in L} C_R$. Next observe that L_M can contain no two rules that are mates. For suppose it did contain two mates, say rules with prefixes $p0$ and $p1$. Consider the second of the two rules to be generated by MPUS, say without loss of generality the rule with prefix $p1$. At the time $p1$ was generated, $p0$ had already been applied and hence all its members had only gray cells. Thus the parent rule, with prefix p , must also have been applicable, and MPUS could not have chosen $p1$.

Consequently, if we let $D_R = L_M \cap C_R$, we must have $|D_R| \leq w + 1$, and so $|L_M| \leq (w + 1)|L| = (w + 1)OPT_S(P)$. To get the tighter bound claimed in the Theorem statement, suppose that the optimal ACL contains OPT_{row} row rules and OPT_{col} column rules. Consider first the row rules, and suppose $0 \leq k \leq w$. For each row rule R in the optimal ACL, D_R can contain at most $w - k$ row rules with prefixes of length greater than k . Moreover, L_M can contain at most 2^k row rules of length less than or equal to k since it can contain at most half the row rules of each prefix length. Thus the total number of row rules in L_M is at most $(w - k)OPT_{row} + 2^k$ for any k , $0 \leq k \leq w$, or this amount minus 1 if we do not use a rule with the empty prefix (i.e., the full-height strip). An analogous argument says the number of column rules in L_M obeys the above bound with “row” replaced by “col,” again with the bound reduced by 1 if we do not use a rule with the empty prefix. Then, since MPUS cannot use both the full-height and the full-width strips (the second would be redundant) we have

$$|L_M| \leq (w - k) \left(OPT_{row} + OPT_{col} \right) + 2^{k+1} - 1 \leq (w - k)OPT_S(P) + 2^{k+1} - 1.$$

If $OPT_S(P) = 0$ then by our above observation that $|L_M| \leq (w + 1)OPT_S(P)$, we have that $|L_M| = 0$ and hence the claimed bound holds. So we may assume that $OPT_S(P) > 0$ and hence $\log(OPT_S(P))$ is defined and at least 0. Thus if we take $k = \lfloor \log(OPT_S(P)) \rfloor$ we get $2^{k+1} \leq 2OPT_S(P)$ and hence

$$|L_M| \leq \left(w - \lfloor \log(OPT_S(P)) \rfloor + 2 \right) OPT_S(P) - 1$$

as claimed.

Lower bound examples $P_{w,k}$ that nearly match this upper bound exist for all integers $w \geq 1$ and k , $1 \leq k < w$. As illustrated in Figure 11 for the case of $w = 5$ and $k = 2$, pattern $P_{w,k}$ is drawn on a $2^w \times 2^w$ grid, and contains 2^{k+1} width-1 black rectangles: $2^k - 1$ full-height black strips located in columns $i2^{w-k}$, $1 \leq i < k$, together with 2^k height $2^{w-k} - 1$ rectangles, all located in column 0, with bottom cells located in rows $i2^{w-k} + 1$, $0 \leq i < 2^k$. The optimal strip-rule ACL proceeds by first picking up all the full-height black strips ($2^k - 1$ rules), then picking up the now pseudo-monochromatic white rows at positions $i2^{w-k}$, $0 \leq i < 2^k$ (2^k rules), and then finally picking up the now-pseudo-monochromatic black column in position 0, for a total of 2^{k+1} rules.

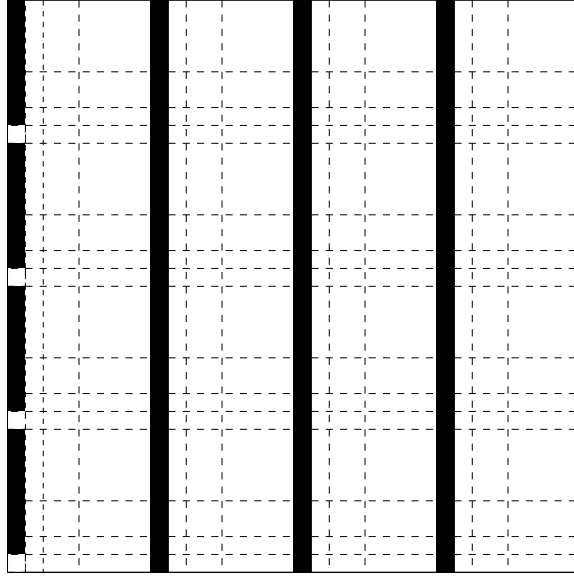


Figure 11: Pattern $P_{5,2}$ used in the MPUS lower bound proof for ACLs.

In contrast, if MPUS chooses badly, it could start by proceeding from right to left, repeatedly picking up the maximal pseudo-monochromatic vertical strip that contains the rightmost column of the figure, until everything has been picked up except column 0. In Figure 11, the left-most boundaries of these rules are indicated by the dotted lines and the boundaries of the black columns. There total number will be $2^k(w - k + 1) - 1$. Then the badly-choosing MPUS could handle the rest of the pattern from top to bottom, repeatedly picking up the maximal pseudo-monochromatic horizontal strip that contains the topmost row of the figure, for an additional total of $2^k(w - k + 1) - 1$; note that we can stop as soon as the last black cell has been picked up. Thus MPUS can use as many as $(w - k + 1)2^{k+1} - 2$ rules. Since $\log(OPT_S(P)) = k + 1$ in this case, we have that the total number of rules is $(w - \lfloor \log(OPT_S(P)) \rfloor + 2)OPT_S(P) - 2$, as claimed. ■

4.2 Understanding MPUS in Terms of Equivalence Classes

The lower bound examples for Theorem 4.1 seem to require very stupid behavior on the part of the MPUS implementer. Could we get better bounds if we used an intelligent tie-breaking rule? As a first step, let us look at the operation of MPUS in more detail. Let us say two rows (or two columns) in a pattern P are *equivalent* if they are identical, that is, if in each pair of corresponding cells, both cells have the same color. Thus we can view the rows and columns of a figure as each partitioned into a set of equivalence classes. Figure 12 illustrates this concept for a simple 2×9 figure. 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.

During the operation of MPUS, we shall say a class is *active* if some member of it is pseudo-monochromatic, has not been picked up, and contains at least one non-gray cell. If that color is c , we say the class is pseudo-monochromatic- c . We say a rule picks up a strip “with color x ” if at the time the rule was applied the strip was pseudo-monochromatic and contained at least one cell

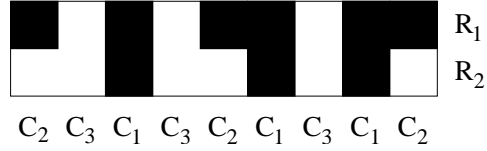


Figure 12: Pattern with rows and columns labeled by their equivalence class.

with color x . We shall also say that a row/column is “picked up” only by the first rule in the RRL/ACL that applies to it. The following theorem lists some of the ways that MPUS interacts with the equivalence classes.

Lemma 4.2 *During the operation of MPUS on any strip-rule pattern, the following properties hold.*

1. *At any time, two rows/columns that have not yet been picked up are identical if and only if they were equivalent to begin with.*
2. *If one unpicked-up member of an equivalence class is pseudo-monochromatic, they all are.*
3. *In each generated rule, all the not-previously-picked-up rows/columns that are picked up are in the same equivalence class.*
4. *At most one row class and one column class can fail to have all its members picked up.*
5. *If the first member of an equivalence class to be picked up is picked up by a rule of color c , then all members of that class that are picked up must be picked up by color- c rules.*
6. *At any given time, the set of active classes consists either only of row classes, only of column classes, or two classes, one of each type, both being pseudo-monochromatic- c for the same color c .*
7. *As long as at least one member of each of the currently active classes remain unpicked-up, no new class can become pseudo-monochromatic.*

Proof. (1) is proved by induction, noting that it is true by definition in the initial pattern, and thereafter no rule can change the relationship (identical or not) between two as-yet-unpicked-up rows/columns. Assume without loss of generality that we are dealing with two rows. As long as neither is picked up, they can only be affected by column rules. Such a rule is applicable only if the two rows already agree in that particular column, so applying that rule has no effect on whether the two rows differ. (2) follows as an immediate corollary.

For (3), suppose members of two different equivalence classes were picked up by the same rule and were not previously picked up. Since they are in different equivalence classes, they are still non-identical by (1). Assume without loss of generality that we are dealing with rows, and consider the column in which the two rows differ. If c is the color in that column’s cell for the first row, then the column’s cell in the second row must be some non-gray color c' . (Since the row and column that intersect in that cell both have non-gray colors, no rule can have turned it gray.) But then the two rows contain two distinct non-gray colors and so cannot be picked up by a single rule.

For (4), we may without loss of generality restrict attention to row classes. If a member of a row class does not need to be picked up, it must be the case that all its cells are grayed by column

rules. However, rows in two different classes must differ in some column, and hence if they are both never picked up, no vertical rule can have picked up that column, a contradiction.

For (5), suppose the first member of equivalence class E is picked up with a color- c rule, and suppose without loss of generality that E is a row class. By (3), that rule picked up only members of E . By (1) they were all identical and must have contained only color- c non-gray cells, with at least one such cell in each member; otherwise there would have been no need for a rule. Suppose, contrary to hypothesis, that some row in E is picked up by a color- c' rule for a $c' \neq c$. By definition, this must have been the first rule in the RRL/ACL that is applicable to the row. This rule could not be applicable unless all color- c cells in the row had been grayed, in which case the row must be all gray. However, since no earlier rule in the RRL/ACL applied to the row, this could only happen if all its cells had been picked up by column rules. This in turn would imply that the current pattern is entirely gray, in which case no further rules are needed, a contradiction.

Property (6) follows from the fact that one cannot have an active row class and an active column class of different colors. Suppose we did, and the colors were $c \neq c'$ for the row and column class, respectively. Then, since both classes are still active, some row in the row class must contain a color- c cell and some column in the column class must contain a color- c' cell. But this means that the cell in the intersection of the row and column must also be non-gray, since it can not have been picked up with either a row or column rule. But this means that it must be color c because it is in a pseudo-monochromatic- c row and color c' because it is in a pseudo-monochromatic- c' column, a contradiction.

For (7), consider without loss of generality a currently non-pseudo-monochromatic row. It must contain cells of at least two distinct non-gray colors. In order for it to become pseudo-monochromatic, all the cells with at least one of these two colors must be turned gray. But one cannot gray all the cells in the row that currently are a given color c without picking up all columns that contain those cell, and hence all of at least one equivalence class. ■

As a consequence of Lemma 4.2 we can view the operation of MPUS as consisting of a series of *phases*, each terminated by the disappearance (total pickup) of an active (pseudo-monochromatic) class.

This suggests a strategy of breaking ties by picking a particular type of available rule and applying all available maximal rules of that type. A first attempt at this might be the *Greedy Algorithm* that picks a currently available type that currently has the fewest maximal rules (further ties broken arbitrarily). Let us denote this algorithm by $\text{MPUS}_{\text{greedy}}$. Note that $\text{MPUS}_{\text{greedy}}$ will construct optimal RRLs/ACLs for all the lower bound examples used in the proof of Theorem 4.1. Unfortunately, it can still perform poorly in the worst-case.

Theorem 4.3 *Consider the asymptotic worst-case ratio for $\text{MPUS}_{\text{greedy}}$ as $\text{OPT}_S(P) \rightarrow \infty$.*

1. *For RRLs, the asymptotic worst-case ratio is 2.*
2. *For ACLs, the asymptotic worst-case ratio lies in the interval $[w/2, 2]$.*

Proof. The upper bounds follow from the general MPUS upper bounds of Theorem 4.1. For the RRL lower bound, we simply augment the examples from the general MPUS lower bound proof of Theorem 4.1 and Figure 10 with one additional row, through the middle, and four additional columns, three on the left and one on the right, as illustrated in Figure 13 for the case of $k = 3$. The optimal RRL first picks up the three black columns, then the now-pseudo-monochromatic central

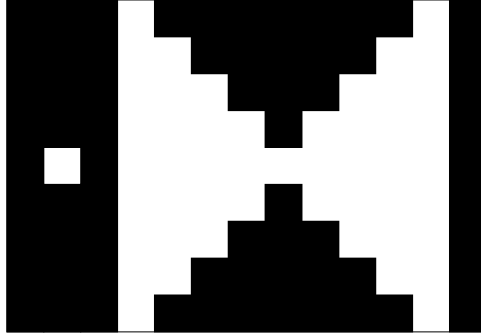


Figure 13: Pattern P_3 , in the series of lower bound examples for $\text{MPUS}_{\text{greedy}}$, where that $\text{OPT}_S(P_k) = 2k + 6$ and $\text{MPUS}_{\text{greedy}}(P_k) = 4k + 8$.

white row, and then the now-pseudo-monochromatic black column on the left. It then proceeds as in the original example, for an overall RRL that has five more rules than the original for a total of $2k + 6$. $\text{MPUS}_{\text{greedy}}$ on the other hand cannot start by picking up the black columns, since that equivalence class requires 3 rules whereas the white columns only require 2. Indeed, it cannot do anything about these framing columns until it has essentially reproduced the worst-case $(4k + 3)$ -rule solution for the original example, after which it can finish off with the 5 rules required to handle the framing columns, for a total of $4k + 8$ or $2\text{OPT}_S(P_k) - 4$ rules. This implies that the worst-case ratio for $\text{MPUS}_{\text{greedy}}$ is at least 2.

For ACLs our examples are a bit more complicated. Figure 14 shows the pattern $P_{4,2}$ from a collection of patterns $P_{w,k}$, $k < w - 1$, where each pattern $P_{w,k}$ is on the $2^w \times 2^w$ grid. In $P_{w,k}$ the pattern can be viewed as partitioned into a $2^k \times 2^k$ grid of $2^{w-k} \times 2^{w-k}$ supercells. Every supercell at or below the diagonal is all-white except for its leftmost column. Every supercell above the diagonal is all-black except for its top row, which is only black in its leftmost cell.

An optimal ACL for this pattern would first use 2^k rules to pick the 2^k black columns. It would then use $2^k - 1$ rules to pick up the top $2^k - 1$ newly-pseudo-monochromatic white rows. Then it would alternate from bottom to top and right to left between picking up the next pseudo-monochromatic-white horizontal strip of width 2^{w-k} and the next pseudo-monochromatic-black vertical strip, each one rendered newly pseudo-monochromatic by the previous pickup, until all that is left is the top-leftmost supercell, which does not need to be picked up since it is pseudo-

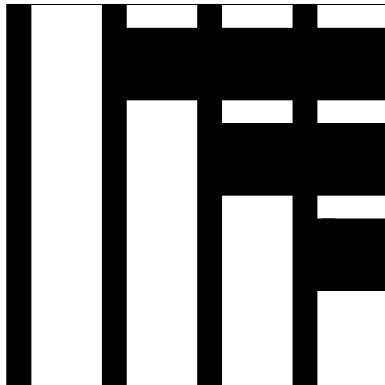


Figure 14: Pattern $P_{4,2}$ used in the proof that $\text{MPUS}_{\text{greedy}}$ performs poorly in the worst case.

monochromatic-white and white is the background color. This alternation requires $2(2^k - 1)$ rules, for an overall total of $2^{k+2} - 3$ rules in the optimal ACL.

Now consider what happens with $\text{MPUS}_{\text{greedy}}$. Let us assume that $2^k > w - k$. Then it will be cheaper to pick up the equivalence class of all-white columns than the one of all black columns, so the algorithm does so, using $w - k$ rules. This renders the topmost class of mostly-black rows pseudo-monochromatic, and the algorithm will pick *them* up with $w - k$ rules. As long as there are more than $w - k$ columns of supercells that contain no picked-up ordinary columns, the algorithm will then continue choose to alternate between picking up the newly-pseudo-monochromatic white column class and the then newly-pseudo-monochromatic black row class, each requiring $w - k$ rules. Thus we will use at least

$$\begin{aligned} 2(w - k)(2^k - (w - k)) &= (w - k)2^{k+1} - 2(w - k)^2 \\ &> \frac{w - k}{2} \text{OPT}_S(P_{w,k}) - 2(w - k)^2 + \frac{1.5}{w - k} \end{aligned}$$

rules. If w is sufficiently large that $w > 3\lceil \log w \rceil$ and we take k to be that latter value, then we will have $2(w - k)^2 = o(\text{OPT}_S(P_{w,k}))$, and so the limit as $w \rightarrow \infty$ of $\text{MPUS}_{\text{greedy}}(P_{w,k})/\text{OPT}_S(P_{w,k})$ will be $w/2$, proving the claimed lower bound on the asymptotic worst-case ratio. ■

4.3 MPUS and 2-Color Strip-Rule Patterns

All the lower bound examples for MPUS presented so far have involved 2-color strip rule patterns. Thus we as yet have seen no advantage for MPUS in the restriction to such patterns. However, since they are particularly relevant to the ACL application, it is worth considering them further, to see if the restriction to two colors *can* be exploited. In Section 5 we shall see that if we are willing to spend $O(n^3)$ time, the restriction can be exploited to the full extent of finding optimal strip-rule RRLs and ACLs. For now, we shall lay the groundwork for those results, and consider what can be done if we want to stay within a constant factor of the $O(n^2)$ time for running MPUS.

The first thing to observe is that the operation of MPUS in terms of equivalence classes is significantly constrained in the 2-color case. It what follows, we shall use the abbreviations WR, BR, WC, and BC to stand for “white rows,” “black rows,” “white columns,” and “black columns,” respectively.

Lemma 4.4 *During the operation of MPUS on a 2-color strip-rule pattern P , as long as there is a non-gray cell remaining, there are precisely two active classes, one each from the pairs $\{WR, BC\}$ and $\{BR, WC\}$.*

Proof. Suppose the colors are black and white and the current pattern contains at least one non-gray cell and does not contain a pseudo-monochromatic black row. We shall show it must contain a pseudo-monochromatic white column. Let C be a column with the maximum number of white cells in the current pattern. We claim that C cannot contain a black cell. Suppose it did, and let R be the row containing that black cell. Since R is not pseudo-monochromatic black, it must contain a white cell, say in column C' . But then, since P is a strip-rule pattern, the Monotonicity Property of Theorem 2.1 implies that in the original pattern C' contained more white cells than did C . Since every row rule so far applied had the same effect on both columns, C' must hence *still* have more white cells than C , a contradiction. ■

This implies that either of the following two *fixed-strip* tie-breaking rules will handle all situations when applying MPUS to 2-color patterns.

- Fixed-strip (BR,WC) Maximal Pick-Up-Sticks ($MPUS_{\underline{B}}$). In each phase, pick-up the members of the active black-row equivalence class until they are all gone or, if no such active class exists, the members of the active white-column equivalence class.
- Fixed-strip (WR,BC) Maximal Pick-Up-Sticks ($MPUS_{|\underline{B}|}$). In each phase, pick-up the members of the active black-column equivalence class until they are all gone or, if no such active class exists, the members of the active white-row equivalence class.

These rules do not specify how to break ties between two different maximal strips of members in the chosen class, but it is not difficult to see that it does not matter. If one restricts attention to rules picking up members of the class, it is easy to verify that *all* the maximal rules must be chosen and it does not matter in which order. In other words, if we let $OPT_{\underline{B}}(P)$ denote that minimum-length RRL/ACL for pattern P consisting entirely of black-row and white-column rules and define $OPT_{|\underline{B}|}(P)$ analogously, we have the following result.

Theorem 4.5 *For all strip-rule patterns P and either RRLs or ACLs,*

1. *The rule list produced by $MPUS_{\underline{B}}$ contains $OPT_{\underline{B}}(P)$ rules.*
2. *The rule list produced by $MPUS_{|\underline{B}|}$ contains $OPT_{|\underline{B}|}(P)$ rules.*

This result may have practical significance in the ACL application, since we have observed that engineers seem to prefer these kinds of ACLs. In the sample of thousands of AT&T access router ACLs mentioned in the introduction, for which 95.3% were strip-rule ACLs, fully 94.3% were $|\underline{B}|$ -ACLs, where recall that columns correspond to source IP addresses, rows correspond to destination IPs, black corresponds to *permit*, and white corresponds to *deny*. These engineers may be giving up something by retaining this restriction, however, in view of the following.

Theorem 4.6 *Let algorithm A be either $MPUS_{\underline{B}}$ or $MPUS_{|\underline{B}|}$.*

1. *There exist strip-rule patterns P with arbitrarily large values of $OPT_S(P)$ such that A generates RRLs of length $2OPT_S(P) + 1$.*
2. *There exist patterns P_k with $OPT_S(P_k) = 2^k$ such that A generates ACLs of length $(w - k + 2)OPT_S(P_k) - 2$.*

Proof. Note that these lower bounds are precisely the same as were shown to hold for unrestricted MPUS in Theorem 4.1. Indeed, the bounds for $MPUS_{\underline{B}}$ follow from the same examples used in the proof of that theorem. The bounds hold true for $MPUS_{|\underline{B}|}$ by symmetry. ■

On the other hand, note that although $MPUS_{\underline{B}}$ performs poorly in each of these lower bound examples, $MPUS_{|\underline{B}|}$ generates an optimal strip-rule RRL/ACL. This suggests that we run both $MPUS_{\underline{B}}$ and $MPUS_{|\underline{B}|}$ on the pattern P and output the better RRL/ACL. Unfortunately, this heuristic is asymptotically just as bad as $MPUS_{greedy}$. Indeed, we have the following.

Theorem 4.7 *Let A be that algorithm that runs $MPUS_{\underline{B}}$, $MPUS_{|\underline{B}|}$, and $MPUS_{greedy}$ on the given pattern and returns the best RRL/ACL found.*

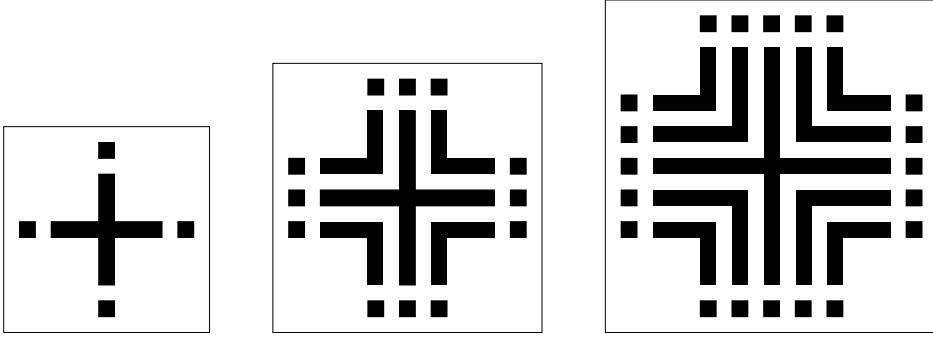


Figure 15: Patterns P_k , $k = 1, 2, 3$, such that $OPT_S(P_k) = 4k + 6$ and $MPUS_{\underline{B}}(P_k) = MPUS_{|B|}(P_k) = MPUS_{greedy} = 8k + 2$.

1. For RRLs, the asymptotic worst-case ratio for A is 2.
2. For ACLs, the asymptotic worst-case ratio for A lies in the interval $[w/4, 2]$.

Proof. The lower bound for the RRL case is implied by a series of patterns P_k , the first three of which are depicted in Figure 15. The optimal RRL starts by picking up the four monochromatic white rows and the four monochromatic white columns. It then picks up the remainder of the pattern from the inside out, alternating between picking up the maximal pseudo-monochromatic black vertical and horizontal strips and then the maximal white pseudo-monochromatic vertical and horizontal strips until nothing but white and gray cells remain, for a total of $4k+6$ rules. The reader may verify that $MPUS_{\underline{B}}$ and $MPUS_{|B|}(P_{w,k})$ use $8k+2$ rules. They work in two phases. The first works from the inside out, but cannot pick everything up so that every class except the first requires two rules to be picked up because its members are separated by unpicked-up rows/columns. The second phase works from the outside in to pick up what was left behind, but again must use two rules to pick up most of the classes. $MPUS_{greedy}$ must initially choose between the equivalence class of monochromatic white rows and the class of monochromatic white columns, and based on that choice will then be forced to mimic the corresponding fixed-strip MPUS variant.

For the ACL case, Figure 16 illustrates the relevant lower bound patterns. Pattern $P_{w,k}$ exists so long as $2(w-k+2) \leq 2^{w-k}$ and contains 2^{k-1} matched horizontal and vertical black segments of width 1, nested as indicated and with their horizontal columns having index congruent to 0 mod

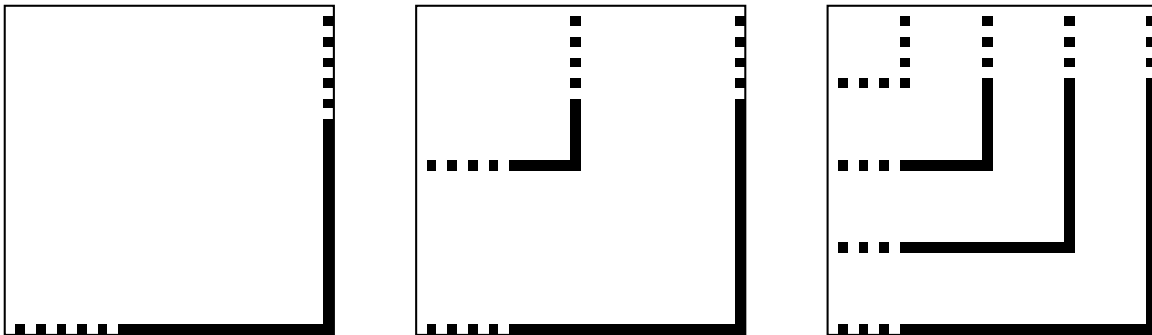


Figure 16: Patterns $P_{w,k}$ for $w = 5$ and $k = 1, 2, 3$, such that $OPT_S(P_{w,k}) = 2^{k+1} + 2(w - k + 1)$ and $MPUS_{\underline{B}}(P_{w,k}) = MPUS_{|B|}(P_{w,k}) = MPUS_{greedy} = (2^{k-1} + 2)(w - k + 2)$.

2^{w-k+1} . In addition, the leftmost $w - k + 2$ even-indexed columns have been turned white, as have the top $w - k + 2$ odd-indexed rows. An optimal ACL begins by picking up these $2(w - k + 2)$ monochromatic white rows and columns, and then uses $4(2^{k-1}) - 2$ white and black strips to finish the job, so $OPT_S(P_{w,k}) \leq 2^{k+1} + 2(w - k + 1)$. Consider $MPUS_{\underline{B}}$ (the argument for the other fixed-strip algorithm is analogous). It begins by picking up the $w - k + 2$ white monochromatic rows. Then it will alternate between picking up the rightmost now-pseudo-monochromatic black column with one rule, and then the bottommost now-pseudo-monochromatic white columns, which will take $w - k + 1$ rules. At this point, only the bottom row will remain to be picked up, which will require $w - k + 2$ black column rules, leading to an overall total of $(2^{k-1} + 2)(w - k + 2)$. $MPUS_{greedy}$ must start by picking either the class of monochromatic white columns or the class of monochromatic white rows, and thereafter will be forced to mimic the relevant fixed-strip algorithm. Thus if we set $k = 2\lceil \log w \rceil$ and let w go to infinity, we can conclude that for these patterns all three algorithms produce ACLs whose ratio to optimal approaches $w/4$ as claimed. ■

In the MPUS tie-breaking rules discussed so far, we have been implicitly assuming that in order to pick up an equivalence class, we must restrict ourselves to rules that individually only pick up members of that class. This may not always be the best way to proceed, however. Consider the pattern in Figure 12. Here the initially active classes are the white columns and black columns. If we only use white column rules to pick up the white column class and black column rules to pick up the black column class, then it will take three rules to pick up the first of the two classes and two to pick up the second, plus one final row rule to handle the black cells in columns that are not initially monochromatic, for a total of six rules. We can, however, pick up all white columns plus one black column with three rules, the same number needed if we only use white column rules: First pick up the third column with a black rule, rendering the strip consisting of the second through fourth columns pseudo-monochromatic. We then can pick up these columns with a single rule and use a final rule to pick up the seventh column. Now we can pick up the remaining black columns with a single rule, covering the sixth through eighth columns, which are now pseudo-monochromatic-black. With one final black row we will have then generated an RRL with only five rules.

This suggests the following general “opportunistic” tie-breaking rule: Whenever picking up an equivalence class, use the minimum number of rules (covering that class and possibly others) required to pick up that class. Subject to this constraint, use a set of rules that picks up the maximum number of rows/columns. We then can consider the *opportunistic fixed-strip* variants of $MPUS_{\underline{B}}$ and $MPUS_{|B|}$. Each uses the same rule as its original version to identify the next class to pick up, but then picks that class up opportunistically (i.e., using the opportunistic tie-breaking rule). Denote the corresponding variants as $OMPUS_{\underline{B}}$ and $OMPUS_{|B|}$. Similarly, $OMPUS_{greedy}$ will denote the version of MPUS that chooses as the next class to pick-up one whose cost is minimum, breaking ties in favor the rule whose opportunistic version picks up the most rows/columns.

As we shall see in Section 5, opportunistic tie-breaking rules play a key role in our $O(n^3)$ and $O(n^3w)$ optimization algorithms for strip-rule pattern RRLs/ACLs. (There always exists an optimal RRL/ACL that picks up each class opportunistically.) Unfortunately, by itself, opportunistic tie-breaking is not enough to improve the asymptotic worst-case behavior of our MPUS variants. Indeed, the patterns illustrated in Figure 15 and 16 have been designed so that opportunism does not help any of our three MPUS variants, as the reader can verify.

Thus at this point we know of no simple tie-breaking rules that can yield worst-case ratios better than 2 for RRLs or $\Theta(w)$ for ACLs, even if we are allowed to run a fixed set of heuristics

and take the best RRL/ACL found. However, if we are not required to bound the number of runs, we can for RRLs get within $3/2$ times optimal in $O(n^2)$ time. Note that this is the same time it would take to generate an MPUS RRL for a strip-rule pattern given by an arbitrary (not-necessarily strip-rule) RRL. In the next section we describe an algorithm that accomplishes this by performing $O(n)$ distinct runs of opportunism-exploiting, MPUS-like subroutines and returning the best RRL found.

4.4 Finding $3/2$ -Optimal Strip-Rule RRLs in $O(n^2)$ time

Our algorithm is based on the following structural lemma.

Lemma 4.8 *For any strip-rule pattern P there exists a strip-rule RRL that contains no more than $\frac{3}{2}OPT_S(P) - 1$ rules and consists either of*

- (A) *A sequence of BR/WC rules followed by a sequence of WR/BC rules, or*
- (B) *A sequence of WR/BC rules followed by a sequence of BR/WC rules.*

Proof. We shall show that an optimal RRL A^* can be transformed into one of the desired form without increasing the number of rules by more than a factor of $3/2$. First, we may assume without loss of generality that A^* is an MPUS RRL. This implies that the row rules in A^* are properly laminar, in that any pair are either disjoint or one is contained in the other, and similarly for the column rules. To see this, suppose two row rules contain a common row but neither is contained in the other. Then the second rule would not be maximal, since it could be expanded to cover all the columns in the first without affecting the final pattern. Next, we can assume without loss of generality that if rule r_1 is contained in rule r_2 , then r_1 comes first in the ACL and is properly contained in r_2 . If it comes later it is redundant and can be deleted, and if comes first but is not properly contained then r_2 is redundant and can be deleted. In what follows we shall refer to these two properties as the *laminar properties*.

To obtain from A^* an equivalent RRL in format (A) we perform a sequence of modifications, each leading to a new RRL that generates the same pattern and continues to obey the laminar properties. Our goal is to construct an RRL in which no white row or black column rule precedes a black row or white column rule. While we have not yet reached this state, we repeatedly perform the following operation. Consider the first BR/WC rule r_1 whose immediate predecessor in the current RRL is a WR/BC rule r_2 . (If we have not reached the desired state, such rules must exist.) If the two rules are the same color, we can interchange them without any effect on the final pattern or the laminar properties, and this is the operation we perform. Otherwise, either r_1 is a black row and r_2 is a white row or r_1 is a white column and r_2 is a black column. If the two rules cover disjoint sets of rows/columns, we again interchange them without any effect on the final pattern or the laminar properties. This leaves the case where r_2 is a proper subset of r_1 . The set of rows/columns in r_1 but not r_2 is then either a single set of contiguous rows/columns or two disjoint contiguous sets, separated by the rows/columns of r_2 . Replace r_1 by the one or two rules corresponding to these contiguous subsets, and then interchange the new rules with r_2 . This will preserve the laminar properties and again the final pattern will be unaffected, although we may now have one additional rule.

It is easy to see that this process must terminate with the desired RRL in format (A). Call the resulting RRL A_1 . The construction of an equivalent RRL in format (B), call it A_2 , is analogous.

All that remains is to show that one of these two RRLs obeys the claimed length bound. To help us prove this, we shall make use of two directed trees, T_{row} and T_{col} . The vertices of T_{row} are the row rules of A^* , and there is an arc from rule r to rule r' if r is properly contained in r' and there is no third rule r'' that contains r and is contained in r' . T_{col} is constructed analogously, with the vertices being the column rules of A^* .

Note that a given rule of A^* cannot be split in both the construction of A_1 and A_2 since only BR/WC rules are split in the former and only WR/BC rules are split in the latter. Moreover, by the laminar property for A^* , no rule can be split more times than the number of arcs into it in its tree. Since the total number of arcs in T_{row} and T_{col} is at most $|A^*| - 2$, this means that the total number of splittings in the construction of A_1 and A_2 , and hence the total number of additional rules created, also obeys this bound. Thus we must have $(|A_1| - |A^*|) + (|A_2| - |A^*|) \leq |A^*| - 2$ and hence $|A_1| + |A_2| \leq 3|A^*| - 2$. Hence the shorter of the two constructed RRLs will have no more than $(3/2)|A^*| - 1 = (3/2)OPT_S(P) - 1$ rules, as claimed. ■

We exploit Lemma 4.8 with the following algorithm. Let m be the number equivalence classes of rows/columns in our given pattern P . Recall that any RRL can be viewed as partitioned into *phases* where each phase ends when a new class becomes pseudo-monochromatic. We use a parameterized subroutine which we shall call *One-Switchover OMPUS*, and denote by $OMPUS_{X,j}$, where X is either $\overline{\mathbf{B}}$ or $|\mathbf{B}|$ and $0 < j \leq m$. $OMPUS_{X,j}$ operates by running $OMPUS_X$ for the first j phases, and then running $OMPUS_{\overline{X}}$ for the remaining phases, where \overline{X} is the other member of $\{\overline{\mathbf{B}}, |\mathbf{B}|\}$. Our algorithm, which we shall call *Multi-MPUS (M-MPUS)*, works as follows

1. For $0 < j \leq m$, run $OMPUS_{\overline{\mathbf{B}},j}$ and let $A_{\overline{\mathbf{B}},j}$ be the RRL it produces.
2. For $0 \leq j < m$, run $OMPUS_{|\mathbf{B}|,j}$ and let $A_{|\mathbf{B}|,j}$ be the RRL it produces.
3. Return the shortest of all the RRLs produced.

Theorem 4.9 *For RRLs and all 2-color strip-rule patterns P , $M-MPUS(P) \leq \frac{3}{2}OPT_S(P) - 1$.*

Proof. Let B be a strip-rule RRL for P with the structure guaranteed by Lemma 4.8, and assume without loss of generality that it consists of a sequence B_1 of maximal BR/WC rules followed by a sequence B_2 of maximal WR/BC rules. We claim that at least one of the RRL's generated by SFS will be of length $|B_1| + |B_2|$, which by Lemma 4.8 will yield the theorem.

Note that B_1 and B_2 must each proceed in phases in which all the rules are of the same type as the current pseudo-monochromatic class being picked up (BR, WR, BC, or WC). We cannot have opportunistic pick-ups since neither B_1 nor B_2 contains both colors of row rule or both colors of column rule. Moreover, we cannot intermix WR and BC or BR and WC pick-ups, since if such a row/column combination both still had non-gray cells, then the cell at their intersection could not be gray and hence would have to be both black and white.

Thus if the end of B_1 corresponds to a phase termination, i.e., if the last equivalence class of rows/columns handled by B_1 has been completely picked up, then running $OMPUS_{\overline{\mathbf{B}}}$ will generate an initial RRL that is no longer than B_1 and picks up the same classes, along with possibly some additional rows and columns that were picked up opportunistically. Moreover, switching over to $OMPUS_{|\mathbf{B}|}$ at this point will generate a final RRL that is no longer than B_2 , since the pattern remaining will contain only rows and columns that were picked up by B_2 . This means that at

least one of the $A_{\underline{B},j}$ RRLs generated by the algorithm will have length no more than $|B|$ and by Lemma 4.8 we will be done.

Note that the above argument would hold even if we used ordinary MPUS rather than OMPUS in our subroutines. The complication that requires us to exploit opportunism comes when B_1 ends in the middle of a phase, with only some of the maximal rules for the current equivalence class included. In this case there is no easy way to tell just which rules were included. Fortunately, we do not need to.

Suppose without loss of generality that B_1 ends in the middle of an equivalence class E of monochromatic black rows, after using a set S of maximal BR rules to pick up a nonempty proper subset of E . Consider the RRL generated by our algorithm when the switchover occurs at the end of the previous phase. In this case our initial OMPUS $_{\underline{B}}$ run has generated no more than $|B_1| - |S|$ rules. Moreover, all the rows in E are pseudo-monochromatic when we are about to start the OMPUS $_{|B|}$ run, so all the rules in $|S|$ cover pseudo-monochromatic sets of rows.

What would be the effect of the omission of the rules in S on B_2 ? Since there are rows of the equivalence class E that were not picked up by BR rules in B_1 , we know that BC rules in B_2 must already pick up all the black cells in the columns of the omitted rules. We also know that no BC rule in B_2 can be blocked by the fact that these BR rules were not used.

The only problem that can arise from our failure to use the rules in S is that some WR rules from B_2 may no longer be legal. In particular the continuing presence of monochromatic black rows may interrupt what was supposed to be a sequence of contiguous monochromatic white rows, splitting it into two or more separate sequences. In particular, if a rule r in B_2 contains j so-far-omitted BR rules from S that do not include the endpoints of r , they would split r into $j + 1$ separate WR rules. However, OMPUS $_{|B|}$, being opportunistic, would not perform these $j + 1$ rules separately, but instead would first perform the j BR rules from S and then the original rule r , for the same total number of rules but a much-increased number of picked-up rows. Thereafter, it will be just as if these j rules from S had been picked up at the beginning of OMPUS $_{|B|}$ run. At the end of the run, the number of extra rules generated by OMPUS $_{|B|}$ will consequently be at most the number of rules in S , for a total of at most $|B_2| + |S|$. Thus the length of the overall RRL will be at most $(|B_1| - |S|) + (|B_2| + |S|) = |B|$, as desired. ■

The bound of Theorem 4.9 is asymptotically tight, as can be proved using the series of patterns illustrated in Figure 17. In pattern P_k from this series, each quadrant contains k nested all-white square frames (counting the frame for the overall figure). In a sense these patterns simulate k nested versions of the patterns P_k in Figure 15 (with the isolated black cells at the ends of each black row and column in those patterns replaced by a complete width-1 black border around the pattern). Those modified patterns still cause both OMPUS $_{\underline{B}}$ and OMPUS $_{|B|}$ to produce RRLs that are off optimal by an asymptotic factor of 2.

The optimal RRLs for the new nested patterns of Figure 17 start by using 8 rules to pick up the outermost white frame and the black frame immediately inside it. They then pick up the $2k$ pairs of alternating black/white crossing pseudo-monochromatic horizontal/vertical strips from the center of the remaining pattern that do not intersect the next level of frames. This results in the merging of the four frames at the next level into one overall frame (the second nested copy of the modified pattern from Figure 15). We then alternate between picking up frames and crossing center strips, with one pair of rules unneeded at the end because only background colors remain. The total number of rules is thus $8k + 4k^2 - 2$, where the asymptotically dominant component

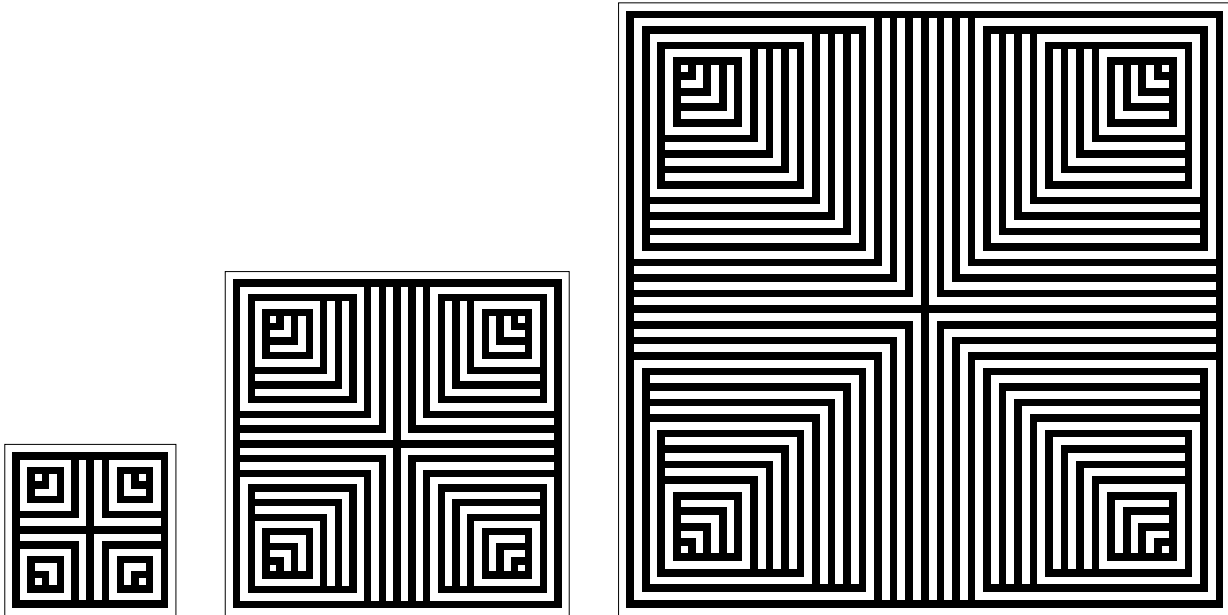


Figure 17: Patterns P_k , $k = 2, 3, 4$, such that $OPT_S(P_k) = 4k^2 + 8k - 2$ and $M\text{-MPUS}(P_k) \geq 6k^2 + 6k - 4$.

consists of the $4k^2$ inner rules.

If One-Switchover OMPUS never switches over, then it will be off by an asymptotic factor of 2, as with the patterns from Figure 15. OMPUS will handle each of the nested patterns in two phases, as it did for those in Figure 15, although now the first phases for all the patterns must precede all the second phases, and in every phase essentially all the classes picked up with one rule by the optimal RRL will require two rules under OMPUS. If One-Switchover OMPUS does switch over, it will still pay essentially a factor of two in the first phase of each nested copy of the original pattern, except possibly the one in which the switchover occurs. Maximal savings can only be obtained if the switchover is done near the end of the run of first phases, in which case OMPUS will be able to pick up nearly all the 2nd phase classes with one rule per class. The damage will already be done in the first phases, however, which is why we get an asymptotic factor of $3/2$.

We now show that One-Switchover OMPUS can be implemented to run in linear time, given a profile of size $O(n)$ for the pattern. This will imply that M-MPUS requires only $O(n^2)$ time if the pattern is originally given by a (not-necessarily strip-rule) RRL that generates it. This is because the $R \times C$ effective grid for a pattern given in this way will have $R + C = O(n)$ and it and its profile can together be constructed in $O(n^2)$ time 3.2, while the number of equivalence classes for the pattern will be no more than $O(R + C)$, so that M-MPUS makes only $O(n)$ calls to One-Switchover OMPUS.

Theorem 4.10 *There is an implementation of One-Switchover OMPUS which, given the profile for any 2-color strip-rule pattern P with an $R \times C$ effective grid and any choice of the OMPUS parameters, constructs its RRL in time $O(R + C)$.*

Proof. We shall augment the original data structures in the linear-time implementation of Pick-Up-Sticks from Theorem 3.2. The additional data structures can be initialized in $O(R + C)$ time and thereafter used to find the next rule under either $OMPUS_{\overline{B}}$ or $OMPUS_{|B|}$ in constant time,

including the time to update them after the rule has been applied (but not including the time to update the original data structures, which we already know takes $O(n)$ time overall). Since the resulting RRL will be of length at most $R + C$, the theorem will follow. The data structures are based on the concept of a *block* of rows/columns. There are two types of blocks:

- (A) A maximal sequence of consecutive rows/columns all of which are either all-gray (picked up) or members of the same equivalence class and whose first and last members have not yet been picked up.
- (B) A sequence of consecutive all-gray rows/columns, either having the pattern border at one end or such that the row/column immediately preceding the sequence and the row/column immediately following are unpicked members of distinct equivalence classes.

We maintain a doubly-linked list for both the rows and columns, consisting of all the row (column) blocks in order. For each block we store its type, and if that type is (A) the identity of the equivalence class for its unpicked-up members. In addition, we store the identity of its first and last row/column, and for each row/column that starts a block, we store a pointer to that block. Note that by Theorem 2.1 each equivalence class is determined by the number of black cells the row/column contains in the initial pattern, and this information is provided by the profile for the pattern. Thus the lists and associated pointers for the original pattern can all be constructed in linear time.

Let us first show how to apply a rule and make the appropriate data structure updates in constant time. Each rule consists of at most three blocks: a block containing unpicked-up members of some pseudo-monochromatic class, call it E , augmented by the immediately preceding block if it is all-gray, as well as the immediately succeeding block if it is all-gray. The blocks before and after the rule (if they exist) must represent as-yet-incompletely-picked up classes different from E (or the pattern boundary). If there is only one bounding block, or two blocks representing distinct classes, then all we have to do is merge all the blocks included in the rule into a single block of type (B). Otherwise, the blocks of the rule are bounded on each side by blocks of type (A) and the same equivalence class E' , so all we have to do is merge the blocks of the rule with those bounding it, creating a single new type-(A) rule for E' . It is easy to see that this all can be done in constant time (plus the time to update the original Pick-Up-Sticks data structures).

Now we show how to identify the next rule(s) in constant time. Suppose without loss of generality that we are supposed to apply $\text{OMPUS}_{\underline{E}}$ and currently there are pseudo-monochromatic black rows, all from the class E . Let E' denote the current class of pseudo-monochromatic white rows (if any exist). Using the basic Pick-Up-Sticks data structures, identify the first unpicked-up row in class E . From this we can identify the first class E block, call it b_1 .

1. If the next type-(A) block is anything but all-gray or E' , our rule is simply to pick up b_1 together with its predecessor if the latter is all-gray.
2. If the next block is all-gray and the block after that is anything but E' , our rule is simply to pick up b_1 , its all-gray successor, and its predecessor if the latter is all-gray.
3. Otherwise, the next not-all-gray block is of class E' , call it b_2 .
 - (3.1) If the next not-all-gray block following b_2 has any class other than E , then our rule once again consists of b_1 together with its predecessor if it is all-gray and its successor if that is all-gray.

- (3.2) If the next not-all-gray block following b_2 has class E , then the next rule consists of block b_2 together with its predecessor if it is all-gray and its successor if that is all-gray.

It is again easy to see that this takes constant time (plus the time to update the original Pick-Up-Sticks data structures). It is also easy to verify that this in fact is a correct implementation of opportunism. We conclude that the overall time used by the algorithm is $O(n)$ as claimed. ■

5 2-Color Strip-Rule Patterns: Constructing Optimal RRLs and ACLs

In this section we present 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 row/column 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 row/column can be picked up more than once. The “color” of the rule will be the color of the non-gray cells in the rows/columns picked up by the rule.

We exploit the fact that the rows and columns of a pattern P can be divided into equivalence classes, as described in the previous section. and the fact that, by Lemma 4.4 at any point before the all-gray pattern is reached in the MPUS process, precisely two equivalence classes will be active (pseudo-monochromatic and containing at least one non-gray cell) and hence ripe for pick-up. We will either have a “mixed state,” where one active class is a row class and the other a column class and both have the same color, or a “row state,” where the two active classes are row classes of different colors, or a “column state,” where the two active classes are column classes of different colors. In addition, we shall make use of the following easily-verified observation.

Lemma 5.1 *During the operation of MPUS on any 2-color strip-rule pattern P , the following properties hold:*

1. *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 2.1).*
2. *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.*

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 active. 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 active 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, as we did in the previous section with the opportunistic heuristics $\text{OMPUS}_{\underline{B}}$ and $\text{OMPUS}_{|\underline{B}|}$. To further illustrate this point, consider the pattern in Figure 12 with which we originally introduced the notion of equivalence class. 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. The first, like the previous one, holds for both RRLs and ACLs. Suppose P is the target pattern, L is an RRL/ACL for P , and S is a segment of L . Define $\text{pattern}(P, L, S)$ 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 rows and columns covered by rules in those segments.

Lemma 5.2 *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 $\text{pattern}(P, L, S) = \text{pattern}(P, L', S')$.*

Proof. It suffices to show that both L and L' must have picked up precisely the same equivalence classes before S was applied.

First, suppose $S = (E_1, c_1, E_2, c_2, U)$ corresponds to a mixed state, in which case $c_1 = c_2$. Let c be the other color. Assume for specificity that E_1 is a row class and E_2 is a column class (the other case is analogous). Then by Lemma 5.1(2) every row class with more cells of color c_1 than E_1 must have been picked up with color c_1 and no row class with fewer can have been picked up with that color. As to row classes picked up with color c , these must be precisely those that intersect column class E_2 in a cell of color c . They need to have been picked up before S since E_2 is pseudo-monochromatic at the beginning of S , and any row class that contains color c_1 in its intersection with E_2 could not have been picked up with c before S since such a class cannot be pseudo-monochromatic- c as long as members of E_2 remain unpicked-up. Similarly, row classes picked up before S with color c are precisely those whose intersection with E_1 contains a cell of color c . Thus the set of classes so far picked up is entirely determined by S , as required.

Next, suppose that S corresponds to a row state, in which case $c_1 \neq c_2$. By Lemma 5.1(2), the row classes that have been picked up so far are precisely those with more cells of color c_1 in P than class E_1 and those with more cells of color c_2 in P than class E_2 . As to column classes, these too are totally determined by S . Clearly any column class whose intersection with E_i does not have color c_i , $i \in \{1, 2\}$, must have been picked up. Otherwise E_i could not be pseudo-monochromatic at the beginning of S . However, any class whose intersection with each E_i is c_i cannot be pseudo-monochromatic in $\text{pattern}(P, L, S)$ since $c_1 \neq c_2$ and there is at least one member of each class E_1 and E_2 that had not yet been picked up when segment S began. Thus no such column class can have been picked up, and the set of row and column classes picked up before S is completely determined by S . An analogous argument holds if S corresponds to a column state. ■

5.1 RRL-Specific Details.

The next lemma specifies a normal form for segments of optimal RRLs. Call a pseudo-monochromatic row/column *active* if it contains at least one non-gray cell, and otherwise call it *gray*. Call an active row/column *color- c* if the non-gray cells it contains are color c . Call a sequence of contiguous pseudo-monochromatic rows/columns 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 5.3 *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 $\text{pattern}(P, L, S)$, 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 $\text{pattern}(P, L, S)$ 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 $\text{pattern}(P, L, S)$, 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 row and column 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 5.2 and 5.3 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 5.3 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 5.3 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 5.4 (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. ■

Lemma 5.5 *Suppose P is a 2-color pattern with n effective grid lines. Then the number of reachable vertices in the RRL segment graph for P is $O(n^2)$.*

Proof. Let n_R be the number of rows in the effective grid and n_C be the number of columns. Since each effective grid line creates one additional row or column, we must have $n_R + n_C \leq n + 2$. Let R_i , $1 \leq i \leq N_R$, denote the row equivalence classes in P , and C_j , $1 \leq j \leq N_C$, denote the row equivalence classes. Note that we must have $\sum_{i=1}^{N_R} |R_i| = n_R$ and $\sum_{j=1}^{N_C} |C_j| = n_C$. In addition, by Lemma 5.4 the number of reachable states for an ordered pair (E_i, E_j) of equivalence classes that can appear in a label is no more than $|E_j|$. If T is the total number of reachable states summed over all such ordered pairs, we thus have

$$\begin{aligned}
T &\leq \sum_{i=1}^{N_R} \left((N_C + N_R - 1) |R_i| \right) + \sum_{j=1}^{N_C} \left((N_C + N_R - 1) |C_j| \right) \\
&< (n + 1) \left(\sum_{i=1}^{N_R} |R_i| + \sum_{j=1}^{N_C} |C_j| \right) \\
&= (n + 1)(n_R + n_C) \leq (n + 1)(n + 2) = O(n^2). \quad \blacksquare
\end{aligned}$$

5.2 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 rows/columns 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 row/column 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 $(x_0, x_1, x_2, \dots, x_w)$, 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 5.6 *Suppose we are given a segment with label (E_1, c_1, E_2, c_2, U) and action i and $c_1 \neq c_2$. Then any two minimum-length lists of rules that pick up all the active members of E_i and have the dominant signature among such segments must have the same set of dominant rule prefixes and thus pick up the same set of rows/columns.*

Proof. Suppose not and assume for specificity that we are dealing with column classes. Let the two lists be L_1 and L_2 and let m be the length of the shortest prefix such that they do not have the same set of undominated rule prefixes of that length. Since L_1 and L_2 have the same signature, each must contain a dominant rule prefix of length m that the other does not. Let (p, c) denote the corresponding rule for L_1 . Let L'_1 be the set of rules in L_1 that are contained in or equal to (p, c) , and let L'_2 be the set of rules in L_2 that are contained in (p, c) . Since (p, c) is a dominant rule in L_1 , only rules in L'_1 can pick up columns addressed by p . Similarly, since L_1 and L_2 agree on all dominant prefixes shorter than p , no rule in L_2 outside those in L'_2 can pick up any columns addressed by p . Thus the segment obtained from L_1 by replacing the rules in L'_1 by those in L'_2 will also pick up all active rules in E_i and hence can be no shorter than L_1 . We thus have $|L'_2| \geq |L'_1|$, and so the segment obtained from L_2 by replacing L'_2 by L'_1 will also pick up all active rules in E_i and will be no longer than L_2 and hence will also be of minimum length. However, it now has one more undominated rule prefix of length m than did L_2 . Since it still has the same number of undominated rule prefixes of all shorter lengths, it thus has a stronger signature than L_2 , a contradiction of our assumption that L_2 already had the dominant signature for such segments. ■

Lemma 5.7 *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 (row/column) 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. ■

To complete our description of the ACL algorithm, we must show how to construct minimum-length segments that consist only of column rules or of row rules, pick up all members of E_i , and have the dominant signature among such segments. To show that the algorithm runs in polynomial time, we will then need to prove an analogue of Lemma 5.4.

We can assume for specificity that we are dealing with column classes. The case of rows is analogous. We may also assume that E_i is a white column class. In the general setting, corresponding to the second part of Lemma 5.7, we may have both active white and active black columns. If there are no active black columns, we have the special case corresponding to the first part of that lemma.

Let P' be the pattern corresponding to the label (E_1, c_1, E_2, c_2, U) . In what follows, we shall call a column “black” if in P' it contains no white cells and at least one black one, “white” if contains no black cells and at least one white one, “gray” if all its cells are gray, and “red” if it contains both white and black cells in P' , that is, if it is not pseudo-monochromatic.

Recall that we say that a rule $r = (p, c)$ is *contained in* a rule $r' = (p', c')$ if p' is a prefix of p , i.e., the range of columns covered by r' contains that for r . Note that if r and r' are such that neither contains the other, the ranges of columns covered by the two rules must be disjoint. Note further that in a minimum-length segment, if r' contains r then the containment must be proper (p' must be a proper prefix of p), since the later of the two rules with the same range would be redundant and could be omitted without affecting the final pattern. Similarly, in a minimum-length segment, if r is contained in r' it must precede it in the rule ordering. The relative ordering of disjoint rules, however, is irrelevant. Thus a minimum-length segment can be specified simply by the rules it contains; any ordering consistent with the above remarks will suffice.

Note that the only rules eligible for inclusion in our segment are ones whose prefixes do not cover red columns. Call such prefixes *legal*. Our algorithm works by computing two values $cost(p, x)$ and $rule(p, x)$ for each legal prefix p and character from $\{W, B\}$, and using these to identify the rules in the segment. The value $cost(p, x)$, $x \in \{W, B\}$, is defined to be the size of the smallest set of rules that have p as a (not-necessarily proper) prefix and collectively pick up all active white columns addressed by p , given that our segment contains a color- c rule whose prefix p' is a proper prefix of p . The value $rule(p, x)$ equals 1 if such a smallest set can contain a rule whose prefix is p and otherwise is 0. Note that since we do not have to pick up black columns, $cost(p, b)$ is also the size of the smallest set of rules that have p as a (non-necessarily proper) prefix and collectively pick up all active white columns addressed by p , given that our segment contains *no* rule whose prefix p' is a proper prefix of p .

We compute the values for $cost$ and $rule$ using dynamic programming, starting with the values for prefixes addressing individual columns. The values for such prefixes are easily seen to be as follows. Let $c \in \{W, B, G\}$ be the color of the addressed column, where the given characters abbreviate “white,” “black,” and “gray,” respectively. In what follows, let $\bar{W} = B$ and $\bar{B} = W$. Then the values for $cost(p, x)$ when p addresses a single column can be seen to be as follows:

$$(1) \quad cost(p, x) = \begin{cases} 1, & c = \bar{x} \\ 0, & \text{otherwise} \end{cases}$$

It is not difficult to see that for such p we have $rule(p, x) = cost(p, x)$.

If p is a legal prefix that does not address a single column, then both $p0$ and $p1$ are also legal prefixes. In this case we have the following recurrences:

$$cost(p, x) = \min \left(cost(p0, x) + cost(p1, x), 1 + cost(p0, \bar{x}) + cost(p1, \bar{x}) \right), \text{ and}$$

$$rule(p, x) = \begin{cases} 1, & cost(p0, x) + cost(p1, x) \geq 1 + cost(p0, \bar{x}) + cost(p1, \bar{x}). \\ 0, & \text{otherwise.} \end{cases}$$

Using the above relations and definitions we can compute the values of $cost(p, x)$ and $rule(p, x)$ for all pairs (p, x) where p is a legal prefix and $x \in \{W, B\}$ in time $O(2^w)$. However, we will not in fact need all these values. We can in fact start with those legal prefixes that address maximal pseudo-monochromatic blocks of columns. And it is easy to see that the above relations imply that (1) and the fact that $rule(p, x) = cost(p, x)$ continue to hold even for legal prefixes p that address blocks of columns, where a block has color B if its columns contain no white cells and a least one black one, color W if its columns contain no black cells and at least one white one, and color G if its columns contain only gray cells. Thus the time to compute all the relevant values will actually be $O(N)$ where N is the number of legal prefixes p that address pseudo-monochromatic blocks and themselves have no prefix that addresses a pseudo-monochromatic block, which is in turn $O(wn)$, where n is the number of effective grid lines in the pattern.

Having computed all these values, we construct the normal-form segment by calling the following recursive function f with arguments p and B for all minimal-length legal prefixes p .

Function $f(p, x)$
 $\{$
 If $rule(p, x) = 1$, add the rule (p, \bar{x}) to the segment and set $x = \bar{x}$.
 If p addresses a pseudo-monochromatic block, return.
 Otherwise, call $f(p0, x)$ and then $f(p1, x)$.
 $\}$

Let $S((E_1, c_1, E_2, c_2, U), i)$ denote the segment constructed by this algorithm (or its appropriate analogue with columns replaced by rows and/or white replaced by black). The following result is relatively straightforward.

Lemma 5.8 For label (E_1, c_1, E_2, c_2, U) and action i , $S((E_1, c_1, E_2, c_2, U), i)$ is a minimum length segment that picks up all active members of E_i and has the dominant signature among all such segments.

Our remaining task is to prove the analogue of the Containment Lemma (Lemma 5.4) for this ACL normal form segment. For this we will exploit an auxiliary function of legal prefixes that we shall call *bias*. We once again assume for specificity that we are dealing with columns and that our goal is to pick up all the active white columns. We define *bias* as follows:

$$\text{bias}(p) \equiv \begin{cases} -1, & \text{cost}(p, B) < \text{cost}(p, W) \\ 0, & \text{cost}(p, B) = \text{cost}(p, W) \\ +1, & \text{cost}(p, B) > \text{cost}(p, W). \end{cases}$$

Bias in a sense indicates the preferred background color for the given prefix, with -1 indicating a preference for black, $+1$ indicating a preference for white, and 0 indicating indifference.

Note that the following recurrence can be used for computing the values of $\text{bias}(p)$. If p is a prefix identifying a single column whose color is $c \in \{W, B, G\}$, then

$$\text{bias}(p) \equiv \begin{cases} -1, & c = B \\ 0, & c = G \\ +1, & c = W. \end{cases}$$

Otherwise, $p0$ and $p1$ must be legal prefixes. In this case, the fact that we must have $|\text{cost}(p, B) - \text{cost}(p, W)| \leq 1$ for all legal p implies that

$$\text{bias}(p) = \begin{cases} -1, & \text{bias}(p0) + \text{bias}(p1) < 0 \\ 0, & \text{bias}(p0) + \text{bias}(p1) = 0 \\ +1, & \text{bias}(p0) + \text{bias}(p1) > 0. \end{cases}$$

The following two lemmas are the key to exploiting the *bias* function:

Lemma 5.9 (Monotonicity Lemma) *The bias function is monotonic, in the sense that if any of the original columns is changed from black to gray or from gray to white, then for no prefix p does $\text{bias}(p)$ decrease.*

Proof. Follows from the recurrence by which $\text{bias}(p)$ is computed. ■

Lemma 5.10 *For all legal prefixes p , $\text{bias}(p) = +1$ if and only if $\text{rule}(p, B) = 1$.*

Proof. First, suppose p addresses a column of color c . If $\text{bias}(p) = 1$ then by definition we must have $c = W$ in which case by definition $\text{rule}(p, B) = 1$. Conversely, suppose $\text{rule}(p, B) = 1$. Then $c = W$, $\text{cost}(p, W) = 0$, and $\text{cost}(p, B) = 1$, all by definition. But, then $\text{bias}(p) = +1$ as desired.

Now let us consider the case where p does not address a column and hence $p0$ and $p1$ are legal prefixes. To simplify the expressions in what follows, we introduce the following shorthand notation:

$$\begin{aligned} c_1 &= \text{cost}(p0, B) \\ c_2 &= \text{cost}(p1, B) \\ c_3 &= \text{cost}(p0, W) \\ c_4 &= \text{cost}(p1, W). \end{aligned}$$

First suppose that $rule(p, B) = 1$. By definition, this means that $1 + c_3 + c_4 \leq c_1 + c_2$, which implies that

$$cost(p, B) = \min \left(c_1 + c_2, 1 + c_3 + c_4 \right) = 1 + c_3 + c_4$$

But then

$$cost(p, W) = \min \left(c_3 + c_4, 1 + c_1 + c_2 \right) = c_3 + c_4 < cost(p, B)$$

and so $bias(p) = +1$.

Suppose now that $bias(p) = +1$, which means that

$$cost(p, W) = \min \left(c_3 + c_4, 1 + c_1 + c_2 \right) < cost(p, B) = \min \left(c_1 + c_2, 1 + c_3 + c_4 \right).$$

If $rule(p, B) = 0$, then by definition we must have $c_1 + c_2 < 1 + c_3 + c_4$ and hence $c_1 + c_2 \leq c_3 + c_4$. This implies $cost(p, B) = c_1 + c_2$. But then since $cost(p, W) < cost(p, B)$ we must have either $1 + c_1 + c_2 < c_1 + c_2$ or $c_3 + c_4 < c_1 + c_2$, both of which yield contradictions. Hence we must have $rule(p, B) = 1$, as desired. ■

The following is an immediate corollary, with analogous results holding when white is replaced by black and/or columns by rows.

Corollary 5.11 *Suppose we are given label (E_1, c_1, E_2, c_2, U) and action i , where E_1 and E_2 are column classes and E_i is a white column class. Let X be the set of prefixes of undominated rules in the list $S((E_1, c_1, E_2, c_2, U), i)$ constructed by our algorithm and let Y be the set of legal prefixes p that have $bias(p) = +1$ but which themselves have no legal prefix with this property. Then $X = Y$.*

We are now prepared to prove the analogue of the RRL Containment Lemma for our ACL algorithm. Recall that the algorithm works by computing a shortest path in the segment graph for our given pattern P .

Lemma 5.12 (ACL Containment Lemma) *Suppose that 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 $U' \subset U$.*

Proof. As in the RRL case, we argue by induction on the depth of the vertices in the graph, with the claim clearly holding for the source vertex. Suppose it holds for all vertices with depth d or less and v and v' are at depth $d + 1$. Once again the hard case is when $c_1 = c_2$, so that one of E_1 and E_2 is a column class and the other is a row class. Assume for specificity that E_2 is the column class. (The other cases can be argued analogously.)

As before, 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 $c_1 = c_2$ and E'_1 is the previously picked-up column class with the least cells of color c among those that were picked up with color c . Since our two given vertices are by assumption reachable from the start vertex, there must be at least one arc into each from a reachable vertex of depth d . Let a be such an arc into the first and a' be such an arc into the second and let x and x' be the tail vertices for a and a' respectively. We break into cases depending on the types of x and x' , with the case where the types are different following from the two cases where they are the same.

First, assume that they are both of Type 1, in which case we can denote them as $x = (E'_1, c, E_2, c_2, X)$ and $x' = (E'_1, c, E_2, c_2, X')$, where $U \subseteq X$ and $U' \subseteq X'$ and by our induction hypothesis we know that either $X \subseteq X'$ or $X' \subseteq X$. We can 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 let us assume, again without loss of generality, that c_2 is white. The sets of legal prefixes for the segments corresponding to arcs A and A' are the same, but our normal-form algorithm may yield different results since some of the columns that are black in $P_{x'}$ are gray in P_x . Let the corresponding bias functions be denoted by $bias_x(p)$ and $bias_{x'}(p)$. By our Monotonicity Lemma (Lemma 5.9), we have that for all legal prefixes p , $bias_x(p) \geq bias_{x'}(p)$. But then Corollary 5.11 implies that every undominated rule (p, c) in the normal-form segment for a' is either undominated in the segment for a or dominated by a rule (p', c') where p' is a prefix of p . Thus every column addressed by a rule in the segment for a' is addressed by a rule in the segment for a , which implies that $U \subseteq U'$, as desired.

The argument for the case where both x and x' are of Type 2 is analogous. ■

Thus the total number of states we encounter in the algorithm is once again $O(n^2)$ by Lemma 5.5. Because our bound on the running time for generating the segments is $O(nw)$, the overall running time becomes $O(n^3w)$.

6 The Strip-Rule Penalty and the ACL Penalty

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. Let $OPT(P)$ denote the length of the optimal RRL/ACL for pattern P when all possible rectangle rules are allowed.

Theorem 6.1 *For RRLs,*

1. *For any strip-rule pattern P , $OPT_S(P) \leq 4 \cdot OPT(P) + 1$.*
2. *There exist patterns P_k such that $OPT(P_k) = \Theta(k)$ and $\lim_{k \rightarrow \infty} \frac{OPT_S(P_k)}{OPT(P_k)} = 3.5$.*

For ACLs on a $2^w \times 2^w$ grid,

1. *For any strip-rule pattern P , $OPT_S(P) \leq 8(w - 1) \cdot OPT(P) + 1$.*
2. *For $w > 1$ and k , $1 < k < w$, there exist patterns $P_{w,k}$ such that $OPT(P_{w,k}) = 2^k - 1$ and $OPT_S(P_{w,k}) = 2^{k-1}(w - k + 2)$, implying an asymptotic worst-case ratio of at least $w/2$.*

The upper bound for RRLs follows simply 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 covering the entire pattern and picking up any non-gray cells remaining after all effective grid lines have been eliminated.

For the RRL lower bound, Figure 18 illustrates the lower bound patterns P_k . Pattern P_k can be constructed with $2 + 2k$ rectangle rules, as we illustrate in Figure 19, which shows the construction

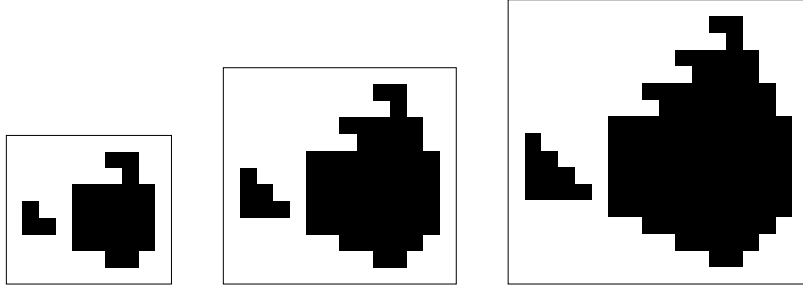


Figure 18: RRL Patterns P_k for $k = 1, 2, 3$, where $\text{OPT}(P_k) = 2k + 2$ and $\text{OPT}_S(P) = 7k + 6$.

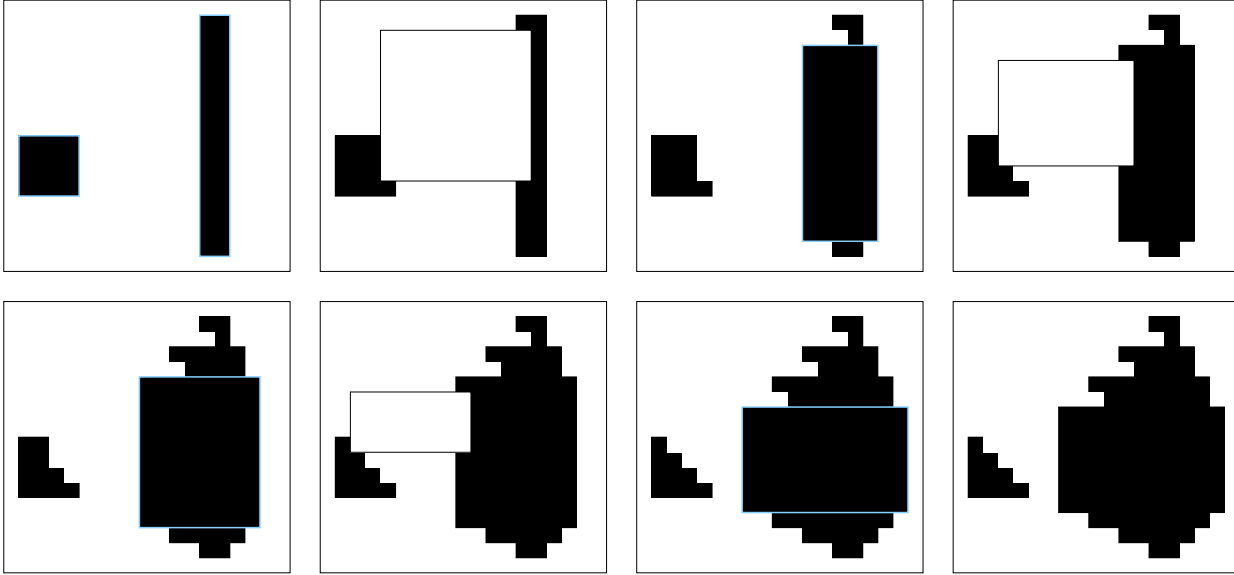


Figure 19: Optimal construction of factor-of-3.5 lower bound pattern P_3 .

of P_3 . We show the construction in RRL order (as opposed to pick-up-sticks order) since this helps us to derive lower bounds on any strip-rule RRL for the pattern. We start with two black rectangles, and follow it with k stages, each of which introduces a white rectangle followed by a black rectangle. After each white rectangle is placed, the pattern temporarily stops being a strip-rule pattern, since the subarray formed by bottommost row and rightmost column of the white rectangle, together with the row immediately above the rectangle and the column immediately to its left contains a checkerboard. The subsequent black rectangle eliminates this defect, however, and the pattern once again becomes a strip-rule pattern.

Our key observation has to do with the equivalence classes created in the process. Note that after the first two black rectangles are placed we have 3 row equivalence classes and 3 column equivalence classes. Thereafter, each time a white rectangle is placed, the numbers of row and column equivalence classes each go up by 2. Each time a black rectangle is placed, the numbers of row and column equivalence classes each go up by 1, but the new classes are split in that their most distant members are separated by members of other row/equivalence classes. The reader may verify that once an equivalence class is created, it is never totally eradicated, although it may lose some members. Thus since no strip-rule can pick up members from different equivalence classes, we already know that $\text{OPT}_S(P_k) \geq 6 + 3k$. Things are worse than this however.

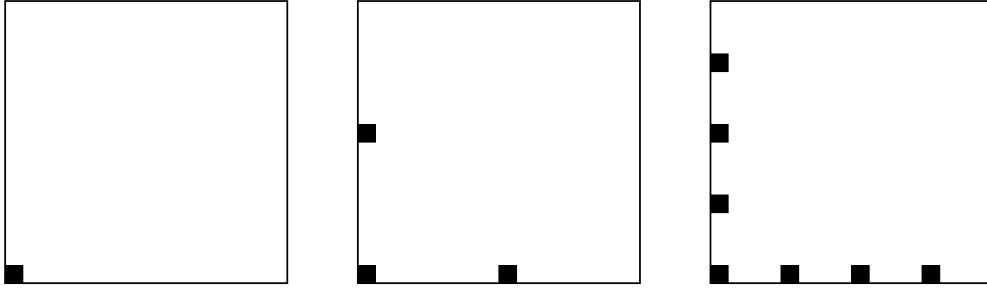


Figure 20: ACL Patterns $P_{w,k}$, $k = 1, 2, 3$, for which $\text{OPT}(P_{w,k}) = 2^k - 1$ and $\text{OPT}_S(P_{w,k}) = (2^{k-1})(w - k + 2) + k - 1$.

Consider the k pairs of row/column equivalence classes created by the k (non-initial) black rectangles, together with the initial pair of the monochromatic white row class and the monochromatic white column class. Each such pair requires at least two strip-rules, one for each class. We claim that in any minimum-length strip-rule RRL, each such pair actually requires *three* strip-rules, with the possible exception of the pair that contains the last rule in the RRL, viewed in pick-up-sticks order (Note that this last rule must gray at least one black cell if the RRL has minimum length.) Suppose there were a pair of classes that does *not* contain that last rule and yet is picked up by just two rules. The key observation is that if a single rule picks up all members of the first class and a single rule picks up all members of the second, then every row/column between the most-distant pair of members must also be gray after both rules have been applied. But it is an easy observation that then *all* black cells in the pattern must have been turned gray. When a black rectangle is placed in the optimal construction described above, all the black cells in the pattern are contained in the union of the horizontal and vertical strips defined by the rectangle, even if one removes the leftmost column from the vertical strip (as the next white rectangle will do). Thereafter, each subsequent black rectangle is contained in the horizontal strips defined by its predecessors. Thus the second rule to be applied must have been the last in the RRL to pick up a black cell, and hence the last in the RRL overall, a contradiction. Thus the total number of rules in the strip-rule RRL must be at least $4 + 4k + 3(k + 1) - 1 = 7k + 6$, for an asymptotic ratio of 3.5 as claimed. (The optimum RRL is no longer than this, since it can be verified that $\text{MPUS}_{|\mathbb{B}|}$ produces an RRL of this length.)

For the case of ACLs, our upper bound on the strip-rule penalty derives from the RRL upper bound together with Lemma 3.4, which bounds how many legal ACL rules we need to simulate an RRL. The lower bound patterns $P_{w,k}$ are all white except that for each i , $0 \leq i < 2^w$, that is congruent to $0 \pmod{2^{w-k+1}}$, the cells $C(0, i)$ and $C(i, 0)$ in the bottommost row and leftmost column are black, as illustrated in Figure 20. This yields a pattern with $2^k - 1$ isolated black cells, and consequently $\text{OPT}(P_{w,k}) = 2^k - 1$, as claimed in the figure caption. As to the optimal strip-rule ACL, this must either pick up all the monochromatic white rows or all the monochromatic columns, which will require $2^{k-1}(w - k + 1)$ strip-rules. Assuming we pick up the rows, we next pick up the now-pseudo-monochromatic-black left column with a single rule, then the $2^{k-1} - 1$ now-pseudo-monochromatic-white rows, which can be picked up with $k - 1$ strip rules, the first picking the top half of the rows, the second taking up the next quarter, etc. Finally, we pick up the $2^{k-1} - 1$ now-pseudo-monochromatic-black columns with single-column rules, yielding the claimed value for $\text{OPT}_S(P_{w,k})$. ■

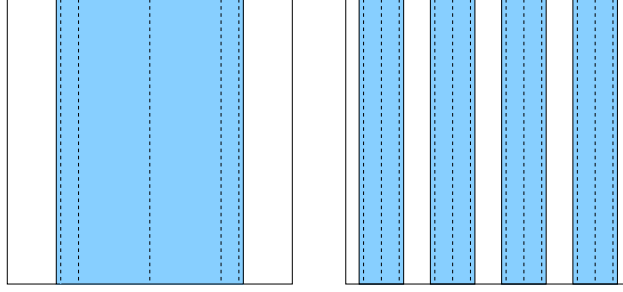


Figure 21: Patterns $P_{w,k}$ for $w = 6$ and $k = 0, 1$ for which $\text{OPT}_S^R(P_{w,k}) = 4^k$ and $\text{OPT}_S^A(P_{w,k}) = (4^k)(w - 2k)$.

Note that the above upper bounds on the strip-rule penalty for ACLs could be tightened if we could improve on our bound for the ACL penalty, which itself is currently not tight. Here are the best bounds we currently know. If P is a pattern on a $2^w \times 2^w$ grid, let $\text{OPT}^A(P)$ and $\text{OPT}^R(P)$ denote the lengths of its optimal ACL and RRL respectively. If in addition, P is a strip-rule pattern, let $\text{OPT}_S^A(P)$ and $\text{OPT}_S^R(P)$ denote the lengths of its optimal strip-rule ACL and RRL respectively.

Theorem 6.2 *Strip-rule patterns:*

1. For any strip-rule pattern P , $\text{OPT}_S^A(P) \leq 2(w - 1) \cdot \text{OPT}_S^R(P)$.
2. For even $w > 0$ and k , $0 < k < w/2$, there exist patterns $P_{w,k}$ such that $\text{OPT}_S^R(P_{w,k}) = 4^k$ and $\text{OPT}_S^A(P_{w,k}) = 4^k(w - 2k)$, implying an asymptotic worst-case ratio of at least w .

Unrestricted patterns:

1. For any pattern P , $\text{OPT}^A(P) \leq 4(w - 1)^2 \cdot \text{OPT}^R(P)$.
2. For $k \geq 0$ and $w > 2k$ such that $w - k$ is even, there exist patterns $P_{w,k}$ such that $\text{OPT}^R(P_{w,k}) = 2^k + 1$ and $\text{OPT}^A(P_{w,k}) > 2^k(w - k + 1)^2/4$, implying an asymptotic worst-case ratio of at least $w^2/4$.

Proof. The upper bounds both follow from Lemma 3.4, which showed how an RRL strip- or rectangle-rule can be partitioned into legal ACL rules of the same type. The claimed lower bound for strip-rule patterns is implied by the patterns $P_{w,k}$, for even $w \geq 3$ and $0 \leq k < w/2$, as illustrated in Figure 21. Pattern $P_{w,k}$ consists of 4^k black strips of width $2 * (2^{w-2k} - 1)/3$ with centers at $(2i - 1) * 2^{w-2k-1}$, $1 \leq i \leq 4^k$. Each such strip is made up $w - 2k$ paired substrips. Starting from the outside these paired substrips have width $1, 4, 16, \dots, 4^{w/2-k-1}$. Let us view the pattern as divided into $2 * 4^k$ equal-width vertical strips, each containing half of one of our 4^k black strips. Note that until such a strip has been rendered pseudo-monochromatic, any legal ACL rule that includes some of its columns must be contained entirely in the strip. To render the strip pseudo-monochromatic, we must either pick up all the white columns or all the black columns. However, we have chosen the width of the maximal black RRL strips so that the each of the two options (pick up the black columns, pick up the white columns) will require $(w/2 - k)$ rules.

For the case of unrestricted patterns, our lower bound examples for $k > 1$ are all based on the ones for odd $w > 4$ and $k = 0$, which are illustrated in Figure 22 for the case of $w = 7$. Note that this is not a strip-rule pattern because it contains checkerboard subarrays. The top black rectangle

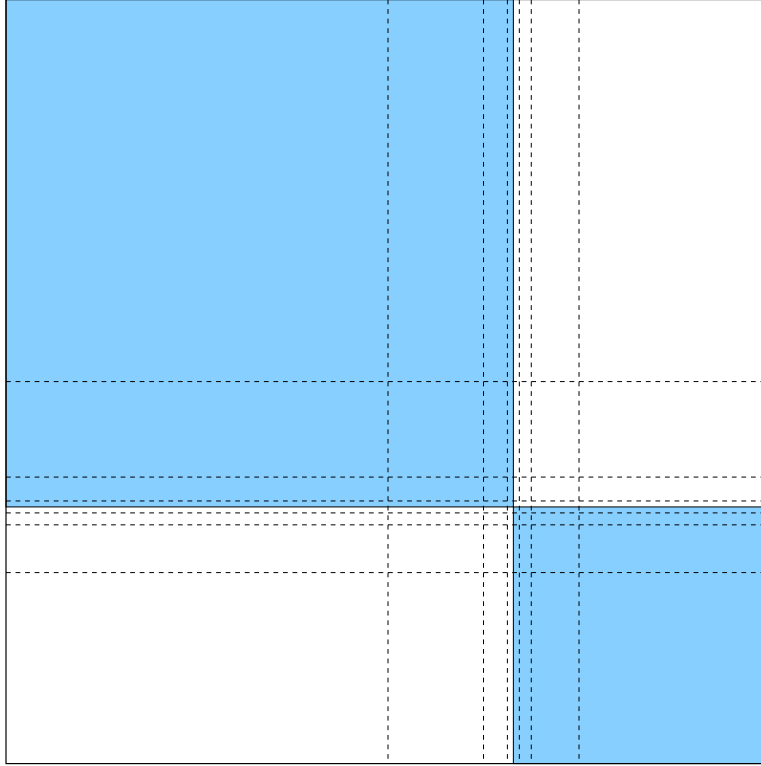


Figure 22: Pattern $P_{w,0}$ for $w = 7$ for which $\text{OPT}_S^R(P_{w,0}) = 2$ and $\text{OPT}_S^A(P_{w,0}) \geq (w + 1)^2/4$.

is a square whose side length is $\sum_{i=0}^{(w-1)/2} 4^i$, implying that the bottom rectangle is a square with side length $1 + 2 \sum_{i=0}^{(w-2)/2} 4^i$. Note also that each of the two black RRL rectangle rules would require $(w + 1)^2/4 = 16$ legal ACL rules. The figure includes dashed lines that partition the canvas into the maximal legal ACL rectangle-rules initially applicable to the pattern. Note that each of the black squares contains $(w + 1)^2/4$ such rectangles, the lower bound on $\text{OPT}^A(P_{w,0})$ claimed by the theorem, as do the two white rectangles that complete the pattern.

Let us call the lines through the black rectangle boundaries on the interior of the canvas the *axes* of the pattern and assign coordinates to the rectangles of the partition based on their distance (in rectangles) from those separators. Thus, for example, the four single-cell rectangles located where the axes intersect all have coordinates $(0, 0)$, the cells immediately above and below them have coordinates $(0, 1)$, etc. Note that for each realized coordinate pair there are four rectangles with those coordinates, and these rectangles will be the at the four corners of their convex hull. In addition, these four can be partitioned into two horizontally-aligned pairs or two-vertically aligned pairs. Moreover, the convex hull of any of these pairs, if pseudo-monochromatic, would be a legal move. Let R_1 and R_2 be such a pair, and consider the minimum strip that contains both R_1 and R_2 . Then any ACL rule that contains a cell in the strip that is not in the convex hull of R_1 and R_2 and a cell in the strip not in the convex hull must contain all of R_1 and R_2 .

Suppose we are given a minimum-length ACL for the pattern in pick-up-sticks format, with each rule consisting of a rectangle that was maximally pseudo-monochromatic at the time it was applied. We claim that for any four rectangles in our partition with the same coordinates, the first rule that picks up a portion of any one of them must be entirely contained in the convex hull of the four rectangles. Consequently, there must be a rule in the ACL for each realized coordinate pair, of which there are $(w + 1)^2/4$, as claimed.

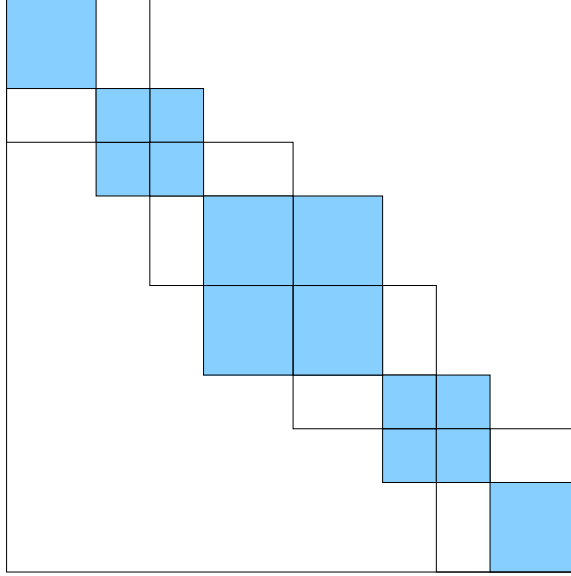


Figure 23: Schematic of pattern $P_{w,k}$ for $k = 2$ with copies of $P_{w-k,0}$ highlighted.

The argument so far covers only the case where $k = 0$. To get a pattern $P_{w,k}$ for odd $w - k$ and $1 \leq k < w/2$, we start by placing 2^k mirrored copies of $P_{w-k,0}$ along the diagonal of the $2^w \times 2^w$ grid. We then replace each interior pair of mirrored black square by a single black square equal to their convex hull. See Figure 23. The above lower bounding argument will apply to each copy of $P_{w-k,0}$ separately and independently, implying our claimed lower bound on $OPT_S^A(P_{w,k})$, whereas the optimal RRL will consist of a rule for each of the black rectangles in the figure, so that $OPT_S^R(P_{w,k}) = 2^k - 1$, as claimed. ■

7 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 1.3, even the best possible RRL/ACL consisting only of black rules can be off by a factor of $OPT(P)$ for patterns where $OPT(P) = \Theta(n)$.

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 $OPT(P) \geq \lfloor N_{col}/2 \rfloor$ 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 , $0 \leq q \leq \log(N_{col})$, we do the following. First we partition the grid into 2^q vertical supercolumns of widths as close to $N_{col}/2^q$ as possible. Then we

partition each supercolumn into a sequence B_1, B_2, \dots of blocks, starting from the bottom. The odd-indexed blocks (the *good* blocks) are subrectangles that extend to the full width of the supercolumn 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 supercolumn. (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 7.1 *The iterated strip-rule algorithm runs in $O(n^3)$ time and produces an RRL whose length is $O(\text{OPT}(P) \min(n^{1/3}, \text{OPT}^{1/2}))$. Moreover, the bound is tight in that there exist patterns with arbitrarily large values of $\text{OPT}(P)$ for which the algorithm produces RRLs whose lengths are $\Theta(\text{OPT}(P)^{3/2})$.*

Proof. Let us first consider the claimed running time guarantee. Assume we are given an RRL of length n that generates P . We first construct the effective grid for P , which takes time $O(n^2)$ by Theorem 3.2. Then it will cost at most $O(n)$ for each run of MPUS, assuming we have the relevant profile. However, the relevant profiles can also be constructed in $O(n)$ time since we are either starting a new profile from scratch, based on a single row of the supercolumn, which can be constructed in time $O(n)$, or else augmenting an existing profile with a single column, again requiring only $O(n)$ time. For a given value of q , the total number of calls to MPUS is roughly $(N_{col}/2^q)N_{row} = O(n^2/2^q)$. Thus the total time for all values of q is

$$O\left(\sum_{i=0}^{\log n} \frac{n^3}{2^i}\right) = O(n^3)$$

as claimed.

Let us now consider the approximation guarantee. Key to the proof is the observations that 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), which allows us to exploit 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 7.2 *Suppose there are r disjoint forbidden subrectangles. Then $\text{OPT}(P) \geq r/4$.*

Proof. Let \mathcal{R} be a collection of r disjoint forbidden rectangles, and let E be any rectangle in \mathcal{R} . Note that by the definition of forbidden rectangle, all the bounding rows/columns 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 row/column 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 $r \leq 4|A| = 4 \cdot \text{OPT}(P)$, and the claim follows. ■

Returning to the approximation guarantee proof, recall from the proof of Theorem 4.1 that if a block has x rows and y columns, the number of rules needed to handle it by MPUS is at most $x + y - 1$. We can thus bound the total number of rules in \mathcal{R}_q , the RRL constructed by ISR when P is divided into 2^q supercolumns, as follows. First note that each combination of a good block together with a bad block immediately above it contains a forbidden rectangle. These forbidden rectangles are all disjoint, and so if the total number of bad blocks is B , then $\text{OPT}(P) \geq B/4$ by Lemma 7.2. Consider the blocks in supercolumn i . These all have width roughly $N_{col}/2^q$. They alternate between good and bad blocks, and so if there are B_i bad blocks in the i th supercolumn, there are at most $B_i + 1$ good blocks. The total height of the good blocks in the strip is at most N_{row} .

Thus there are at most $B + 2^q$ good blocks, all with width roughly $N_{col}/2^q$ and cumulative height at most $2^q N_{row}$, and B bad blocks each of which is one column high and roughly $N_{col}/2^q$ rows wide. Exploiting the fact that $N_{row} \leq N_{col}$ we thus have that the total number of rules $|\mathcal{R}_q|$ is roughly bounded by

$$\begin{aligned} (B + 2^q) \frac{N_{col}}{2^q} + 2^q N_{col} + B \frac{N_{col}}{2^q} &= 2B \frac{N_{col}}{2^q} + N_{col} + 2^q N_{col} \\ &\leq 8\text{OPT}(P) \frac{N_{col}}{2^q} + N_{col}(2^q + 1). \end{aligned}$$

Now since no rule can create more than two effective vertical grid lines, we must have $N_{col} \leq 2\text{OPT}(P) + 1$. Consider the above bound on $|\mathcal{R}_q|$ for $q = \lceil \log(\sqrt{\text{OPT}(P)}) \rceil$. In this case we obtain

$$|\mathcal{R}_q| = O \left(\text{OPT}(P) \frac{N_{col}}{\sqrt{\text{OPT}(P)}} + N_{col} \sqrt{\text{OPT}(P)} \right) = O(\text{OPT}(P)^{3/2}).$$

Thus the best of the \mathcal{R}_q 's contains at most $O(\text{OPT}(P)^{3/2})$ rules. This will be $O(\text{OPT}(P)n^{1/3})$ unless $\text{OPT}(P) > n^{2/3}$, in which case the input RRL, which has length n , will be $O(\text{OPT}(P)n^{1/3})$. Our claimed overall bound of $O(\text{OPT}(P) \min(n^{1/3}, \text{OPT}(P)^{1/2}))$ follows from the fact that we return the input RRL if it is shorter than all the constructed \mathcal{R}_q 's.¹ This completes the upper bound proof.

Finally, we need to show that ISR can be this bad. For every M we exhibit a 2-color pattern P_M on a $2M^2 \times 2M^2$ grid, that has $\text{OPT}(P) \leq 3M^2$ and yet $\text{ISR}(P) = \Omega(M^3)$. To construct our pattern, we start with a white background, then color all even rows and columns black, and finish off by recoloring the following M^2 black cells white, with each recoloring forming a local checkerboard pattern. The recolored cells have coordinates $(M(i-1) + 2i, M(j-1) + 2j)$ for $1 \leq i, j \leq M$. Note that this pattern can be described with $3M^2$ rules — M^2 row rules plus M^2 column rules plus M^2 individual cell rules. See Figure 24 for the pattern P_4 .

We suppose that the pattern is given to us inefficiently, with one rule for each of its $4M^4$ cells, in which case returning the input RRL will be of no use to us. Now note that for a given value of q , ISR must handle q strips of approximate dimensions $2M^2 \times 2M^2/q$, having roughly $2M^2 + 2M^2/q$ effective grid lines. Thus it generates $\Theta(M^2)$ rules for each strip. If $q > M$ the overall RRL will thus have to contain $\Theta(M^3)$ rules and we are done. So assume $q \leq M$. Then all the strips have

¹Thanks to Mihai Patrascu for this observation.

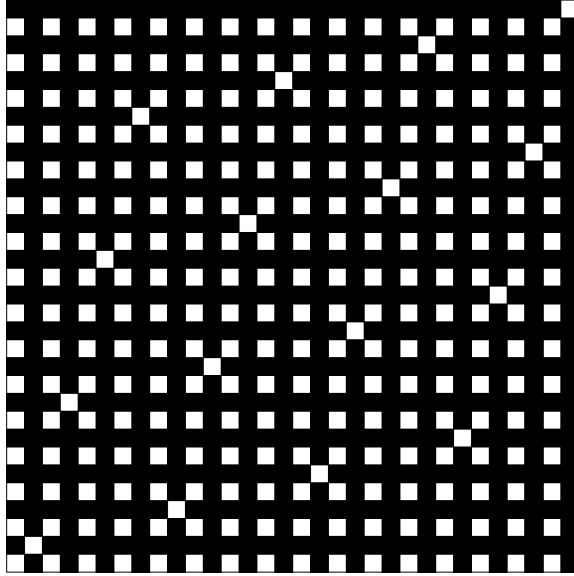


Figure 24: ISR lower bound example for $M = 4$.

height roughly $2M$ or greater. In running the algorithm we will encounter M^2 checkerboards overall, no two at the same time. Each will yield a bad rectangle for which MPUS will generate a number of rules equal to roughly half the strip height, for a total once again of $\Omega(M^3)$ rules. (Note that the above argument works even if we run the algorithm with the roles of rows and columns interchanged, so trying both possibilities and taking the better will not yield improved results.) ■

To obtain “near-optimal” ACLs, we can simply use ISR to generate an RRL and then partition each rectangle-rule of the RRL into a set of maximal legal ACL rules. Since the minimum-length ACL is no shorter than the minimum-length RRL, this guarantees an ACL within a factor of $O(w^2 \text{OPT}(P) \min(n^{1/3}, \text{OPT}(P)^{1/2}))$.

At present, we know of no polynomial-time approximation algorithm that has a better guarantee than ISR for either RRLs or ACLs. We do know that a natural Greedy heuristic is no better than ISR for RRLs. This heuristic (GR) starts from the effective grid and generates rules one at a time in the order they will appear in the output ACL. At each step we repeatedly choose a rule that colors the current monochromatic rectangle with the most as-yet-uncolored cells, where a rectangle is monochromatic if all its as-yet-uncolored cells have the same color.

Theorem 7.3 *There exist patterns P with arbitrarily large values of $\text{OPT}(P)$ for which the Greedy algorithm can produce ACLs with $\Omega(\text{OPT}(P)^{3/2})$ rules.*

Proof. For every M we exhibit a 2-color pattern P_M on a $4M^2 \times 4M^2$ grid, that has $\text{OPT}(P) \leq 4.5M^2$ and yet the Greedy algorithm can produce lists with as many as $\Omega(M^3)$ rules. To construct pattern P_M , we first create a coarse grid consisting of an $M \times M$ grid, each super-row and super-column of which contains $4M$ of our original rows and columns, and with the resulting grid cells alternating between black and white in checkerboard fashion, the bottom leftmost cell being black. We then overwrite this pattern by recoloring the i th (original) row black for $i \equiv 1 \pmod{4}$ and white for $i \equiv 3 \pmod{4}$. Finally, for the j th row of the coarse grid, $1 \leq j \leq M$, and each coarse grid cell in the row, color the j th and the $M + j$ th columns the same color as the original color of the grid cell. See Figure 24 for the pattern P_4 .

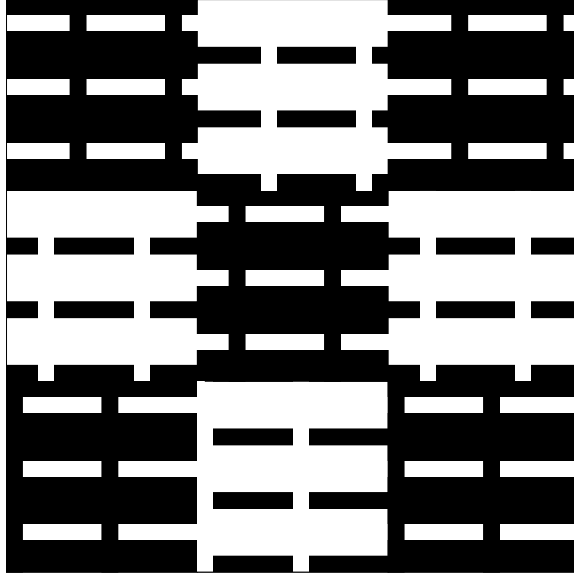


Figure 25: GR lower bound example for $M = 3$.

First note that as claimed P_M can be constructed with $4.5M^3$ rules: $M^2/2$ for the coarse grid cells, preceded by $2M^2$ for the black and white rows, preceded by $2M^2$ for the grid cell columns (actually rectangles whose height is that of a coarse grid cell). Also note that every grid line of the pattern is indeed an effective grid line. Next observe that the largest monochromatic rectangles are the $3 \times 4M$ subrectangles of the coarse grid cells that retain the cells' original colors. Also note that if one of these is colored in isolation, no new monochromatic subrectangles are created that have this many as-yet-uncolored cells from the original grid. With infelicitous tie-breaking, the Greedy algorithm can pick $\Theta(M^3)$ such rectangles in such a way that none of them interact: From every other coarse black cell in each row, pick every fourth such black subrectangle from bottom to top. ■

It may well be that there exist tie-breaking rules for the Greedy algorithm that can circumvent the examples in the above proof, and several other appealing heuristics have yet to be fully analyzed, but for now ISR provides the best guarantee known.

8 NP-Completeness for 2-Color RRLs

In this section we prove 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 $P' = P$, but unfortunately, as we have seen, white rules can make a difference and 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 exploit some basic observations about strip-rule patterns.

The basic building block of our new pattern is the 7×7 tile depicted in Figure 26. Note that this figure contains 12 effective grid lines, 6 horizontal and 6 vertical. (We do not count the boundaries of the tile itself, which will not be effective grid lines in our overall construction.) A strip-rule RRL for this pattern will need at least 3 horizontal and 3 vertical rules since no strip-rule can create more than 2 effective grid lines. Moreover, such an RRL exists: Let us view the RRL in pick-up-sticks order, that is, the order in which earlier rules overwrite later ones and we model this by saying that a rule “picks up” or “turns gray” the cells to which it refers. We first pick up the vertical and horizontal black strips covering the middle row and column, then the width-3 horizontal and vertical white strips covering the middle three rows/columns, and finally the vertical and horizontal black strips covering the middle five rows/columns. This RRL is uniquely optimum among strip-rule RRLs: There are four equivalence classes of rows and of columns, so we must pick up at least three column classes and three row classes. Hence in a length-6 RRL for this pattern, each rule must pick up an entire equivalence class, and consequently the sequence of moves is forced (up to an interchange of horizontal/vertical moves of the same width).

Let N be a large number to be specified later. Our basic construction involves a $21N \times 21N$ grid (the *overall grid*) constructed from these tiles as follows. First completely fill up the grid with $3N$ rows of $3N$ tiles. Let the *frame* of the pattern consist of the union of all those tiles in the first N rows of tiles, the last N rows of tiles, the first N columns of tiles, and the last N columns of tiles. The remaining (central) $N \times N$ grid of tiles constitutes the *picture portion* of the grid. To complete the construction, we modify the picture portion of the grid by turning a subset of its white cells black, that subset to be specified later. There are certain common observations we can make about RRLs for the overall grid, independent of how this last alteration is performed. In what follows, we shall continue to view the RRL in pick-up-sticks order.

Consider just the $12N - 4$ tiles on the boundary of the grid. Collectively they contain $72N$ transitions between black and white perpendicular to the boundary and touching it. Note that no rectangle rule can generate more than four of these transitions, so the RRL must be of length at least $18N$. Suppose in what follows that the RRL is of length $18N + x$ for $x < N/4$. What can we say about it? Note that a strip-rule can generate four of the $72N$ transitions, two on one side of the overall grid and two on the other. Call a rule *defective* if it either is not a strip-rule, is a strip-rule one of whose boundaries is not an effective grid line of the pattern, or is a strip-rule one of whose boundaries is also the boundary of an earlier strip-rule in the RRL. Note that no defective rule can generate more than two of the $72N$ boundary transitions. Thus our length- $(18N + x)$ RRL can contain at most $2x$ defective rules and must contain at least $18N - x$ non-defective strip-rules.

Say a row of tiles is *clean* if no defective rule in the RRL has a horizontal boundary inside it. Similarly, call a column of tiles *clean* if no defective rule has a vertical boundary inside it. By the above, there must be at least $3N - 4x$ clean rows of tiles and $3N - 4x$ clean columns of tiles, since each defective rule has at most two horizontal and two vertical boundaries. Say a tile is *pure*

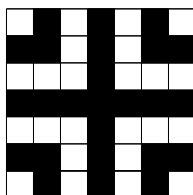


Figure 26: Tile in NP-completeness proof for RRLs.

if it is in both a clean row and a clean column of tiles. Since every row and column contains $2N$ frame tiles and $4x < N$, every clean row of tiles must contain more than N pure frame tiles, as must every clean column. Moreover, each clean row and column must contain a pure tile on each side of the picture portion of the overall grid.

Now by definition, the only rules with boundaries inside a pure tile are strip-rules. As the reader may verify, the only way to generate the pattern in a pure frame tile using strip-rules (even ones with only one boundary inside the tile) is to use the previously mentioned set of rules that were optimal for an individual tile, in the specified order. As can be confirmed by case analysis, any failure to follow that scheme while using strip rules leads to the necessity of using two strip-rules that share a boundary, implying that one of them is defective, which is not allowed since this is a pure tile. (Note that a pure tile can be contained in the *interior* of a defective rule, so one must consider separately the cases in which it is so contained or it is not, and if it is, the case in which the first defective rule containing it is black and the case in which the first such rule is white. Fortunately, this white rule case is equivalent to that in which no defective rule contains the tile, since in the latter case the default background applies, and it is white.) Thus, since every clean row and column must contain at least one pure frame tile, we have the following: for each clean row of tiles, our RRL must contain three non-defective strip-rules that pass through its center — a width-1 black rule, a width-3 white rule, and a width-5 black rule — and they must occur in this order. The analogous property holds for clean columns.

Our NP-completeness proof works by embedding our given RECTILINEAR PICTURE COMPRESSION pattern P into the picture portion of the grid in such a way that P can be constructed out of k rectangles if and only if the resulting $21N \times 21N$ grid pattern has an RRL of length $18N + k$ or less.

We may assume without loss of generality that P is given as a pattern in a $3k \times 3k$ grid, since if it can be realized as a union of k black rectangles it has at most $2k$ effective horizontal grid lines and $2k$ effective vertical grid lines. Let $M = 4k + 1$ and $N = 3kM$. We can then view our overall grid (before the picture portion is modified) as a $9k \times 9k$ grid of *supertiles*, each supertile in turn being an $M \times M$ grid of our standard tiles. The picture portion of the grid is thus a $3k \times 3k$ grid of supertiles, which we can put in one-to-one correspondence with the grid cells of the $3k \times 3k$ grid that contains P . The modification of the picture portion of the overall grid is then obtained by turning black all the white cells in the set of supertiles corresponding to black grid cells in P . This completes the construction of P' . See Figure 27 for a schematic example, where in order to fit the grid on the page, we have taken $N = 3$ and have pretended that a supertile consists of a single tile rather than an $M \times M$ grid of them.

We now wish to show that P' has an RRL of length $18N + k$ if and only if P is representable by a collection of k or fewer rectangles. One direction is easy. If P is so representable, then our RRL consists of the $18N$ strip-rules required to construct the frame of the grid, preceded by the k or fewer black rectangles of supertiles corresponding to the rectangles in the collection of rectangles that represent P .

Now suppose that the overall grid has an RRL of length $18N + k$ or less. Since by construction $k < N/4$ the above structural arguments apply with $x = k$ and we know that the RRL contains at most $2k$ defective rules, yields at most $4k$ unclean rows and $4k$ unclean columns, and is structured as claimed above for the clean rows and columns. Consider any black supertile in the picture portion of the grid. Since it is M tiles high and M tiles wide and $M - 4k = 1$, there must be at least one clean row and one clean column of tiles passing through the interior of the supertile, and

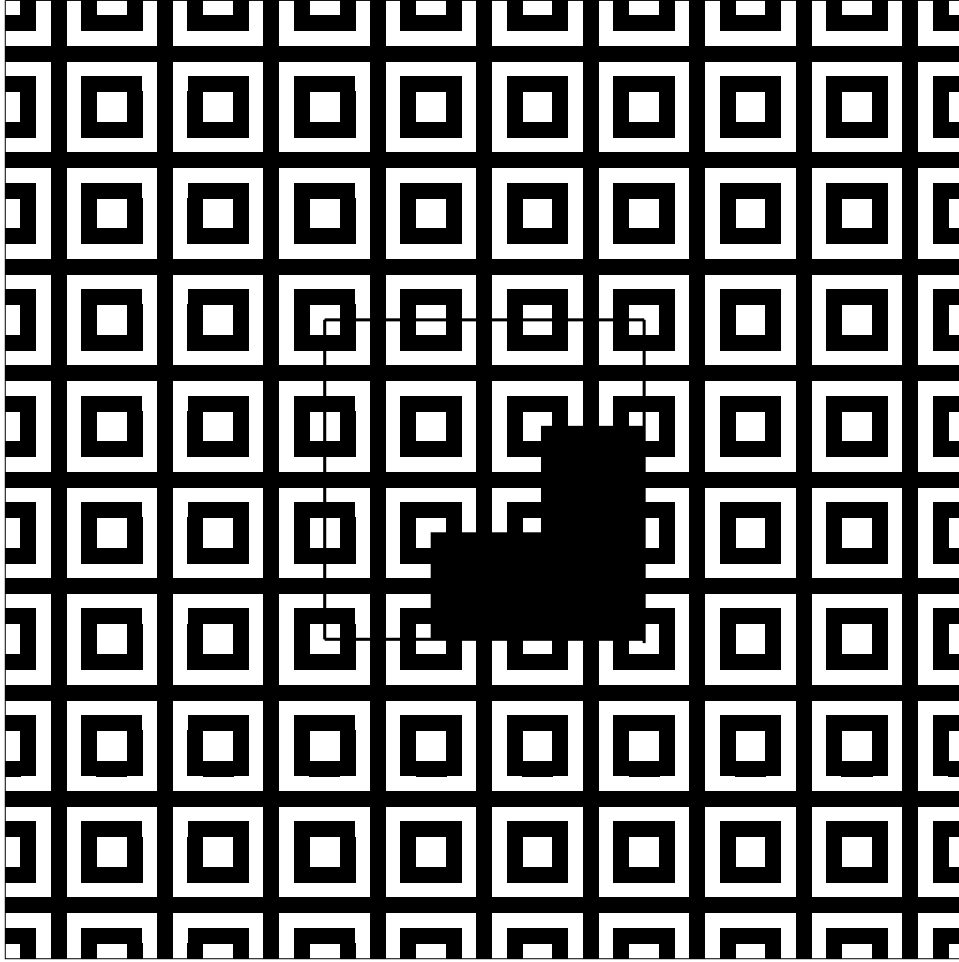


Figure 27: The grid constructed in the NP-completeness proof for RRLs. The picture portion of the construction is outlined. In order to fit the picture on a page, we take $N = 3$ and we consider a supertile to consist of a single tile.

the supertile must contain at least one pure tile. If one looks only at the effect of non-defective rules on this tile, they must yield the default pattern of Figure 26. Since the tile is all-black in the final pattern, it must be intersected by at least one defective black rule that occurs before the width-3 white strip-rules that pass through its center. Since the tile is pure, that defective black rule must include all of the tile. Call such a defective rule a *covering black rule*.

For each black supertile, pick one covering black rule that contains a pure tile from that supertile and label the supertile with that rule. Call this the *chosen rule* for the supertile. Also, pick a pure tile in the intersection of the rule with the supertile and call this the *chosen pure tile* for the supertile.

Claim 8.1 *Let S be a black supertile, R the chosen rule for S , and T the chosen pure tile for S . Then any pure tile T' that is in R and in the same clean row or column as T must be contained in a black supertile from the image P' of P .*

Proof of Claim. Suppose that T' is not in a black supertile from the image of P . Then T' must look like Figure 26 in the final pattern. As mentioned above, R must precede the width-3 white strip-rules that go through T' , and hence would turn the whole tile black unless it in turn were preceded by additional rules that recreated the desired pattern for the tile. The only rules from our required

set that can precede the required width-3 white rules are the required width-1 black rules through the center of the tile. So at least one additional rule is needed to create the transitions that occur at the top and bottom (or left and right) of the width-3 white rule. But rule boundaries are needed if transitions are to be created, so this would mean that two rules sharing a common boundary would intersect the tile (this new rule and the original width-3 white rule). One of these would by definition have to be defective, thus contradicting our assumption that T' is a pure tile. ■

Note that the Claim implies that R cannot share any border with the boundary of the overall grid, for if it did, it would of necessity contain a pure tile from the frame that shared a clean row or column with T , and the black supertiles in P' are all contained in the picture portion of the overall grid. Thus the chosen rules cannot create even one of the $72N$ boundary transitions and so there can be at most k such rules if there are to be $18N + k$ or fewer rules over all.

Now, for each chosen covering black rule, consider the smallest rectangle that contains all the supertiles that it labels. Each such rectangle will be made up of supertiles entirely contained in the picture portion of our grid, and by the previous paragraph there will be at most k of them. We claim that the corresponding collection of rectangles in the $3k \times 3k$ grid for P must be the desired representation for P , that is, their union will precisely equal P .

By construction their union *includes* all the black cells of P . Suppose it also includes a white cell C , and let R be the rectangle in our cover that includes it. Let R' be the covering black rule in our RRL that gave rise to R and $h(R')$ be the smallest rectangle that encloses all the supertiles labeled by R' . Let C' be the supertile in our overall grid corresponding to C , and note that C' is not one of the black supertiles of P' since C is not black in P . Since C' is in $h(R')$, there must be supertiles C'' and C''' labeled by R' such that the two are either to the left and right of C' , above and below C' , or one is above/below and the other is left/right. Consider the chosen pure tiles T'' and T''' for C'' and C''' respectively. Let $p(R')$ be the smallest rectangle that encloses these two tiles. This rectangle is contained in R' and must contain tiles in C' .

If C'' and C''' are respectively to the left and right of C' or respectively above and below it, then $p(R')$ must contain the intersection of C' with the clean row/column that contains T'' . Since C' is of width/height $M = 4k + 1$, this clean row/column must be intersected by at least one clean row/column inside C' , and hence must contain a pure tile inside C' . But by the Claim this pure tile would have to be in a black supertile of P' , a contradiction.

Suppose on the other hand that C'' and C''' are not on opposite sides of C' . Assume without loss of generality that C'' is to the left or right of C' and C''' is above or below. Now $p(R')$ must contain the tile that is in the intersection of the clean row containing T'' and the clean column containing T''' . This tile would thus be a pure tile in R' sharing a clean row with T'' and hence would have to be in a black supertile of P' , again a contradiction.

Thus the hypothesized white cell C contained in our cover of P cannot exist and so our cover is exact. We conclude that there does exist a representation of P with k or fewer black rectangles, as claimed, and our construction represents the desired polynomial transformation. Hence it is NP-hard to construct an optimal RRL for an arbitrary rectilinear black and white pattern.

9 Additional Results and Open Problems

At present 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?

Among the more technical problems left open in the paper, we would be particularly interested in determining the precise asymptotic strip-rule penalty for RRLs, which we have shown lies in the interval $[3.5, 4]$.

In addition, there are several interesting ways in which the current research might be extended. For example, 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? Also, can we extend them to case where the initial pattern has just two colors, but certain cells can be labeled as “don’t-care” and are hence allowed to get either color. For ACLs, this would correspond to the case where the pattern is given by an ACL including some true rectangle rules (restricting both source and destination IP addresses) but these all come at the beginning of the list so that subsequent strip rules cannot affect them.

A final extension, which currently seems of more academic than practical interest but has been raised by several of our colleagues, is the following. What about strip rule RRLs in $d > 2$ dimensions, when strips can restrict at most $k < d$ of the dimensions. Are there interesting characterization theorems for various values of k and d ?

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_licn/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.
- [GT85] H. N. Gabow and R. E. Tarjan. A linear-time algorithm for a special case of disjoint set union. *Journal of Computer and System Sciences*, 30(2):209–21, 1985.
- [Han02] Y. Han. Deterministic sorting in $O(n \log \log n)$ time and linear space. In *Proc. 34th Ann. ACM Symposium on Theory of Computing*, pages 602–608, 2002.
- [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=srml50_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.