Updating a table of bounds on the minimum distance of binary linear codes


**Document status and date:**
Published: 01/01/1985

**Document Version:**
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

**Please check the document version of this publication:**
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

**Link to publication**
UPDATING A TABLE OF BOUNDS ON THE
MINIMUM DISTANCE OF BINARY LINEAR CODES

by

Tom Verheesf

AMS Subject Classification: 94B05

EUT Report 85-WSK-01
ISSN 0167-9708
Coden: TEUEDE
Eindhoven, January 1985
November 1984, Revised January 1985

Updating a Table of
Bounds on the
Minimum Distance of
Binary Linear Codes

Tom Verhoeven
Department of Mathematics and Computing Science, Eindhoven University of Technology

Abstract--This paper presents an algorithm for updating a table of bounds on the minimum distance of binary linear codes. Using a PASCAL program implementing this algorithm an updated table of bounds has been generated for codeword lengths less than 128.

CONTENTS

1. Introduction 2
2. Theoretical Discussion 5
2.1. General Code Construction Techniques 5
2.2. The Table of Bounds and Some Operations on It 10
2.3. The Invariance of the Table of Bounds 12
2.4. Disturbing Invariance and Restoring it 14
2.5. Propagation Rules 17
2.6. Combining Propagation Rules 19
2.7. Associating References 22
3. Pragmatic Considerations 24
3.1. Our Choice of Propagation Rules 24
3.2. The Updating Algorithm Revisited 24
3.3. Generating an Initial Invariant Table 30
4. Implementation Details 32
4.1. Interpreting a Table (and other user aspects) 32
4.2. Inside Details of Programs 34
5. Concluding Remarks 37
Acknowledgments 38
REFERENCES 39
INDEX 41

APPENDICES
I. The Updated Table of Bounds
II. Lists of External Improvements Used for:
The Reconstructed Helgert and Stinaff Table
The Updated Table
III. Report for the Updated Table
IV. Program Listings
1. Introduction

In this paper a \([n,k,d]\)-code is a binary linear code with codeword length \(n\), dimension \(k\) and minimum distance at least \(d\); \(n\), \(k\) and \(d\) are called the code's parameters, and \([n,k,d]\) its type. Let \(d_{\text{MC}}(n,k)\) be defined by

\[
\text{d}_{\text{MC}}(n,k) := \text{MAX} \{ d \mid \text{there exists a } [n,k,d]\text{-code} \}
\]

No practical general method to determine \(d_{\text{MC}}(n,k)\), given \(n\) and \(k\), is currently known. The existence of a \([n,k,d]\)-code implies \(d_{\text{MC}}(n,k) \geq d\); similarly, the nonexistence of \([n,k,d]\)-codes means \(d_{\text{MC}}(n,k) < d-1\). Using the best codes known and the sharpest nonexistence results, Helgert and Stinaff compiled a table of upper and lower bounds on \(d_{\text{MC}}(n,k)\) for \(n < 128\), which was published as [A] in 1973. Many new codes and nonexistence proofs tightening their bounds, however, have appeared since then.

We were assigned the task of producing an updated table of bounds incorporating these new results. This task reduces to two subtasks: collecting new results, and updating the table. The first subtask is a matter of scanning the available literature, extracting actual bounds, and checking their usefulness. Usually this raises no additional problems, because articles on bounds (existence and nonexistence of binary linear codes) are readily identifiable as such, and often the table in [A] is explicitly referred to.

The second subtask, which is the topic of this report, turns out to be somewhat more complicated than may at first seem. Updating, of course, simply consists in replacing some values by new ones. Consider, however, the improvement of a lower bound by the discovery of a new code. From this new code many others can be derived by applying general code construction techniques, possibly involving some previously known codes as well. A few standard techniques involving only the new code are: adding a parity check, shortening, and puncturing; among the techniques combining two codes are: concatenation, and the, so called, \((u,u+v)\)-construction (for details see section 2). This construction process can be repeated, i.e. from the newly derived codes still more codes can be derived using the same techniques (like repeatedly shortening). Not all of these (indirect) derivatives are necessarily good codes, but some may improve the currently known lower bounds.

In general, whenever a promising new code is found, most derivatives will not yet have been constructed (a notable exception being the addition of a parity check). When compiling a table of bounds, however, one is obviously interested in the best that is achievable. So, if easily derived codes could improve a bound, they will have to be constructed and taken into consideration. The situation for upper bounds is analogous, because the nonexistence of codes of one type leads--via similar constructions--to the nonexistence of others. To put all of this in a

2
somewhat different language: the improvement of one bound can result in the improvement of others. You might say that improvements can propagate according to certain "propagation rules." It is this erratic—but desirable—propagation of improvements that complicates the updating task and makes the "pencil-and-eraser" approach unworkable.

We set ourselves the goal of producing a table of bounds that is "closed" or "invariant" under a set of propagation rules, i.e., the final table has to be such that no more improvement is possible by applying any propagation rule (the table in [A] is not closed, see section 5). This still leaves us the freedom of choosing our propagation rules. The more rules the better the bounds, but also the more effort it will demand to compute the closure. We shall come back to the choice of propagation rules in section 3. A second goal is to maintain a reference with each bound, indicating where it originated, like the table in [A]. This reference tells whether a bound resulted from a propagation rule (and if so which one) or whether it was obtained otherwise (and if so in what article). The references allow one to trace each bound to its ultimate origins and thus should improve the usability of the table, ease its checking, and thereby increase its credibility.

We shall aim at an algorithm that will restore the table's invariance after the improvement of a single bound, by systematically applying propagation rules until no more change is possible. This algorithm should also keep the references up to date. Making one improvement and applying the algorithm will be called making an update based on this improvement. Compiling a complete table (8128 pairs of bounds for \( n < 128 \)) is now a matter of starting with a trivial invariant table and making an update based on each improvement found in the literature. The table in [A] can roughly be thought of as constructed from a little over 150 of such updates (but, again, it is not closed, as opposed to any table resulting from the suggested updating algorithm). Our new table embodies more than 80 additional updates (making over 30 previous improvements superfluous).

Implementing this algorithm as a computer program has several advantages over manually updating the table. Automatic updating is less prone to error, it takes care of future updates on the table, and it enables us to check the table in [A] (this table has to be reconstructed anyway, since typing it into the computer would defeat the first advantage).

We have developed a package of programs, written in PASCAL on the university's mainframe computer, for the maintenance of a table of upper and lower bounds on \( \binom{n}{k} \) with \( 1 \leq k \leq n < 128 \). Besides the updating algorithm this package includes such features as generating an initial (not too trivial) table, maintaining statistics, generating reports on updates, and pretty-printing (part of) a table.
1. Introduction

The following sections expose the underlying ideas. We have tried to separate several levels of concerns. Section 2 deals with theoretical aspects: propagation rules, how to obtain them from general construction techniques, some of their properties, how to use them in computing the closure after the improvement of a bound, and the placement of associated references. Section 3 is about the more practical (program design) aspects: our choice of propagation rules, the updating algorithm, the generation of an initial invariant table, this includes the problems due to the finiteness of the table. Section 4 is concerned with implementation details of the actual programs, it also explains how to use our table (interpret the references as justification for an entry). Finally, section 5 summarizes some of our experiences after using the package. The appendices contain the results, especially the updated version of the table of bounds, and complete source listings of the two most important programs.
2. Theoretical Discussion

In this section we shall give a more detailed treatment of improvement propagation. In order to describe the notion of a propagation rule and some related concepts, we need a suitable context and a few notations. A complete formalization of all the concepts involved would lead us too far astray and wouldn't make for better reading, therefore we shall sometimes rely on intuition, if it does not seem to harm understanding. PASCAL-like constructs will be used for some descriptions.

2.1. General Code Construction Techniques

First we deal with a familiar aspect of coding theory: general techniques (for binary linear codes) to construct a new code from one or more known codes. These general construction techniques form the basis of the propagation rules. The adjective "general" is intended to restrict attention to techniques that (a) allow codes of (almost) any type to serve as input code, and (b) need not know more about the structure of the input codes than their type. We are not so much interested in the construction techniques themselves as in their effect. By the effect of a construction we mean the resulting code's parameters expressed as a function of the parameters of the input codes (this is possible because of assumption (b)). We use the same identification for constructions as in [AI]. Proposition 1 will be abbreviated P1, etc. Construction B (discussed below) shows how a technique that is not general, can be transformed into a (somewhat weaker) general technique.

Below we describe the effects of several general construction techniques. The proposition "there exists a \([n,k,d]\)-code" is abbreviated to \([n,k,d]\) (the code's type). The effect of a construction is in the first place written as a logical implication (\(\Rightarrow\)), which can also be read as "yields", with the type of the input code(s) on the left-hand side and the type of the resulting code on the right-hand side. Universal quantification over free variables (often \(n\), \(k\) and \(d\)) is intended, but not explicitly mentioned; sometimes a restriction on their range will be included (1 \(\leq k \leq n\) is always required).

Each code construction technique can also be used to derive the nonexistence of codes of one type from that of another, by contraposition or the standard reduc'tio ad absurdum argument (if the one would exist then so would the other by the construction under discussion, but the other does not exist, contradiction, so the one does not exist). The following relationship between code (non)existence and bounds on \(d_{\text{max}}\) allows other equivalent formulations of the effect of a construction technique:

\[
\begin{align*}
\text{[n,k,d]} & \quad \text{is equivalent to} \quad d_{\text{max}}(n,k) \geq d & (2) \\
\text{not [n,k,d]} & \quad \text{is equivalent to} \quad d_{\text{max}}(n,k) \leq d-1 & (3)
\end{align*}
\]
For each construction technique, therefore, we give a number of equivalent propositions describing its effect, these are true propositions by virtue of the construction. This enables one to switch from one mode of thinking to another, without too much interruption. Eventually we derive the propagation rules from them.

Property 1 is a code construction without input codes, and thus is not useful for improvement propagation. It is mentioned, because it is used when generating an initial table of bounds (see below), and because a "static" bound on \( d_{\text{max}} \) depends on it ((20), see remarks below P4). All these equivalences require proofs, which we have omitted, since they are all very similar. Property 2 serves as an example of how the proofs run.

INTERMEZZO: Some functions and a few of their properties.

Integer valued functions defined for a real argument:

\[
\begin{align*}
\text{ceil}(x) & := \text{smallest integer not less than } x \\
\text{floor}(x) & := \text{largest integer not greater than } x \\
\text{evenceil}(x) & := \text{smallest even integer not less than } x \\
\text{evenfloor}(x) & := \text{largest even integer not greater than } x \\
\text{oddceil}(x) & := \text{smallest odd integer not less than } x \\
\text{oddfloor}(x) & := \text{largest odd integer not greater than } x
\end{align*}
\]

Boolean valued functions defined for an integer argument:

\[
\begin{align*}
\text{even}(n) & := n \text{ is even} \\
\text{odd}(n) & := n \text{ is odd}
\end{align*}
\]

Some properties (for real \( x \) and \( y \), and integer \( n \)):

- the 6 \text{ceil}/\text{floor} functions are idempotent: \( f(f(x)) = f(x) \), and monotonous: if \( x \geq y \) then \( f(x) \geq f(y) \) \hspace{1cm} (4)
- \( \text{evenceil}(x) = 2 * \text{ceil}(x/2) \) \hspace{1cm} (5)
- \( x \geq \text{evenceil}(y) \) is equivalent to \( \text{evenfloor}(x) \geq y \) \hspace{1cm} (6)

Using PASCAL constructs (on the right-hand side) we can write:

\[
\begin{align*}
\text{ceil}(n/2) & := (n+1) \text{ DIV } 2 \\
\text{floor}(x) & := \text{trunc}(x) \\
\text{evenceil}(n) & := n + \text{ord}(\text{odd}(n)) \\
& = n + n \text{ MOD } 2
\end{align*}
\]

(End of Intermezzo)
2. Theoretical Discussion

P1: Boundary cases

\[ [n,1,n] \text{ and not } [n,1,n+1] \text{ and } [n,n,1] \text{ and not } [n,n,2] \]
\[ \text{and } d_{\text{max}}(n,1) = n \text{ and } d_{\text{max}}(n,n) = 1 \]

P2: Adding a parity check

\[ [n,k,d] \implies [n+1,k,\text{evenceil}(d)] \quad (10) \]
\[ \text{if } d_{\text{max}}(n,k) \geq d \text{ then } d_{\text{max}}(n+1,k) \geq \text{evenceil}(d) \quad (11) \]
\[ d_{\text{max}}(n+1,k) \geq \text{evenceil}(d_{\text{max}}(n,k)) \quad (12) \]

\[ \text{not } [n,k,d] \implies \text{not } [n-1,k,\text{oddfloor}(d)] \quad (13) \]
\[ \text{if } d_{\text{max}}(n,k) \leq d \text{ then } d_{\text{max}}(n-1,k) \leq \text{oddfloor}(d) \quad (14) \]
\[ d_{\text{max}}(n-1,k) \leq \text{oddfloor}(d_{\text{max}}(n,k)) \quad (15) \]

In (13) to (15) \( n > 1 \) is supposed. Propositions (10) to (15) are equivalent. The equivalence proofs are fairly straightforward, relying on (2) and (3), and often some substitutions.

\( (10) \implies (11) \) on account of (2)
\( (11) \implies (12) \) take \( d = d_{\text{max}}(n,k) \) in (11)
\( (11) \implies (12) \) due to (4) (monotonicity of \( \text{evenceil} \))
\( (10) \implies (13) \) substitute \( n-1 \) for \( n \) and \( \text{oddfloor}(d) \) for \( d \) in (10), use (6) and contraposition
\( (10) \implies (13) \) similar
\( (13) \implies (14) \) substitute \( d+1 \) for \( d \) in (13), use (3) and (9)
\( (14) \implies (15) \) take \( d = d_{\text{max}}(n,k) \) in (14)
\( (14) \implies (15) \) monotonicity of \( \text{evenfloor} \)
\( (12) \implies (15) \) substitute \( n-1 \) for \( n \) in (12) and use (6)

REMARKS:

--P2 is slightly different from Property 2 in [A], although, no doubt, the formulation we have given was intended (the parity check is not supposed to be added only to optimal codes with odd minimum distance). From P2 and P3 (see below) follows:

\[ \text{if } \text{odd}(d_{\text{max}}(n,k)) \text{ then } d_{\text{max}}(n+1,k) = d_{\text{max}}(n,k)+1 \quad (16) \]

--P2 implies: \( d_{\text{max}}(n+1,k) \geq d_{\text{max}}(n,k) \).

P3: Puncturing (deleting a coordinate)

\[ [n,k,d] \implies [n-1,k,d-1] \]
\[ \text{if } d_{\text{max}}(n,k) \geq d \text{ then } d_{\text{max}}(n-1,k) \geq d-1 \quad (k<n) \]
\[ d_{\text{max}}(n-1,k) \geq d_{\text{max}}(n,k)-1 \]

\[ \text{not } [n,k,d] \implies \text{not } [n+1,k,d+1] \]
\[ \text{if } d_{\text{max}}(n,k) \leq d \text{ then } d_{\text{max}}(n+1,k) \leq d+1 \]
\[ d_{\text{max}}(n+1,k) \leq d_{\text{max}}(n,k)+1 \]
P4: Shortening
(selection on and subsequent deletion of a coordinate)

\[
\begin{align*}
[n,k,d] & \Rightarrow [n-1,k-1,d] \\
\text{if } d_{\text{max}}(n,k) & \geq d \text{ then } d_{\text{max}}(n-1,k-1) \geq d & (k>1) \\
d_{\text{max}}(n-1,k-1) & \geq d_{\text{max}}(n,k) \\
\text{not } [n,k,d] & \Rightarrow \text{not } [n+1,k+1,d] \\
\text{if } d_{\text{max}}(n,k) & \leq d \text{ then } d_{\text{max}}(n+1,k+1) \leq d \\
d_{\text{max}}(n+1,k+1) & \leq d_{\text{max}}(n,k)
\end{align*}
\]

REMARKS:
--P2, P3 and P4 are monotonicity properties of \(d_{\text{max}}\). These are needed for some of the equivalences below. By induction we get from

P2: if \(n \geq m\) then \(d_{\text{max}}(n,k) \geq d_{\text{max}}(m,k)\) \hspace{1cm} (17)

P3: if \(n \geq m\) then \(d_{\text{max}}(n,k) \leq d_{\text{max}}(m,k)+n-m\) \hspace{1cm} (18)

P4: if \(s \geq 0\) then \(d_{\text{max}}(n,k) \geq d_{\text{max}}(n+s,k+s)\) \hspace{1cm} (19)

--Using (19) and P1 we get:
\[
d_{\text{max}}(n,k) \leq d_{\text{max}}(n-k+1,1) = n-k+1
\]

--Combining (19) and (17), taking \(s \geq 0\), yields:
\[
d_{\text{max}}(n,k) \geq d_{\text{max}}(n+s,k+s) \geq d_{\text{max}}(n,k+s), \text{ or equivalently}
\]
\[
\text{if } j \geq k \text{ then } d_{\text{max}}(n,k) \geq d_{\text{max}}(n,j)
\]

A: Helgert and Stinaff construction ([A], residual code)

\[
\begin{align*}
[n,k,d] & \Rightarrow [n-d,k-1,\text{ceil}(d/2)] \\
\text{if } d_{\text{max}}(n,k) & \geq d \text{ then } d_{\text{max}}(n-d,k-1) \geq \text{ceil}(d/2) & (k>1) \\
d_{\text{max}}(n-d_{\text{max}}(n,k),k-1) & \geq \text{ceil}(d_{\text{max}}(n,k)/2) \\
2d_{\text{max}}(n-d_{\text{max}}(n,k),k-1) & \geq \text{evenceil}(d_{\text{max}}(n,k)) \\
2d_{\text{max}}(n-d_{\text{max}}(n,k),k-1) & \geq d_{\text{max}}(n,k)
\end{align*}
\]

E: One-step Griesmer bound (equivalent to A)

\[
\text{not } [n,k,d] \Rightarrow \text{not } [n+2d,k+1,2d] \\
\text{not } [n,k,d] \Rightarrow \text{not } [n+2d-1,k+1,2d-1] \\
\text{if } d_{\text{max}}(n,k) & \leq d \text{ then } d_{\text{max}}(n+2d,k+1) \leq 2d \\
d_{\text{max}}(n+2d_{\text{max}}(n,k)+1,k+1) & \leq 2d_{\text{max}}(n,k)
\]

REMARKS:
--There is only a historical reason for using different identifiers (A and E) to denote these equivalent properties.
--The last line of A and of E are quite similar, but it is not so trivial to prove their equivalence using (17) and (18).
--From \(k>1\) and (20) follows \(d_{\text{max}}(n,k) < n\), so that the first argument of \(d_{\text{max}}\) in the last three lines of A is positive.
2. Theoretical Discussion

B: Construction using dual codes (ii and repeated shortening)

\[[n,k,d] \text{ and not } [n,n-k,s+1] \Rightarrow [n-s,k-s+1,d] \]

if \( \max(n,k) \geq d \) and \( \max(n,n-k) \leq s \)
then \( \max(n-s,k-s+1) \geq d \)
\( \max(n-n+s,k-s+1) \geq \max(n,k) \)

not \([n,k,d]\) and not \([n+s,n-k+s+1]\) \Rightarrow \[n+s,k+s-d\]

if \( \max(n,k) \leq d \) and \( \max(n+s,n-k+1) \leq s \)
then \( \max(n+s,k+d) \leq d \)
\( \max(n+s,n-k+1) \leq \max(n,k) \)

whenever \( s_0 = \min \{ s > 0 : \max(n+s,k-s+1) \leq s \} \)

REMARKS:

-- Construction Y1 (see [M]) is not a general technique in our sense, since it involves the minimum distance of the dual code as well. Construction Y1 states: if there exists a \([n,k,d]\)-code \( C \) and \( t \) is the minimum distance of the dual code of \( C \), then there exists a \([n-t,k-t+1,d]\)-code. Repeated application of P4 yields a \([n-s,k-s+1,d]\)-code for any \( s \geq t \). Obviously \( \max(n,n-k) \geq t \) holds, so it is fine if \( t \leq \max(n,n-k) \). This results in a general technique that constructs a new code from one code and the nonexistence of another.

-- The existence of \( s_0 \) is not asserted. In fact, it does not exist for \( k = n \), and rightly so.

-- Other equivalences might have been added, e.g.:

\([n,k,d] \Rightarrow [n,n-k,s+1] \text{ or } [n-s,k-s+1,d] \)

C: Concatenation of codes

\[[n,k,d] \text{ and } [m,k,c] \Rightarrow [n+m,k,d+c] \]

if \( \max(n,k) \geq d \) and \( \max(m,k) \geq c \)
then \( \max(n+m,k) \geq d+c \)
\( \max(n+m,k) \geq \max(n,k) + \max(m,k) \)

not \([n,k,d]\) and \([m,k,c]\) \Rightarrow not \([n-m,k,d-c]\)

if \( \max(n,k) \leq d \) and \( \max(m,k) \geq c \)
then \( \max(n-m,k) \leq d-c \)
\( \max(n-m,k) \leq \max(n,k) - \max(m,k) \)

D: \((u,u+v)\)-construction

\[[n,k,d] \text{ and } [n,j,c] \Rightarrow [2n,k+j,\min(d,2c)] \]

if \( \max(n,k) \geq d \) and \( \max(n,j) \geq c \)
then \( \max(2n,k+j) \geq \min(d,2c) \)
\( \max(2n,k+j) \geq \min \{ \max(n,k), 2\max(n,j) \} \)

REMARKS:

-- We omitted the nonexistence consequences because they turn out to be of little practical value, besides, they do not allow a compact presentation (case analysis required).
2. Theoretical Discussion

2.2 The Table of Bounds and Some Operations on It

Although we may have more than one table of bounds at our disposal, the operations we have in mind work on only one such table (we shall not consider operations combining two or more tables). From now on we shall speak of the table of bounds, being the table that is the subject of all operations. We represent that single table of bounds for the theoretical discussion (in the programs a different representation is used) by two triangular lower-left matrices (two-dimensional arrays, mappings) $Lb(n,k)$ and $Ub(n,k)$ with $1 \leq k \leq n$; for the time being we do not bother about their size, take the range of $n$ to be upwardly unbounded.

A restriction on the range of $n$ introduces what we shall call an artificial boundary, $k = 1$ and $k = n$ are called natural boundaries. The matrix elements of $Lb$ and $Ub$ are, respectively, lower and upper bounds on $d_{\text{max}}$, i.e. positive integers, satisfying:

$$Lb(n,k) \leq d_{\text{max}}(n,k) \tag{22.a}$$
$$d_{\text{max}}(n,k) \leq Ub(n,k) \tag{22.b}$$

If $Lb(n,k) = Ub(n,k)$, then $d_{\text{max}}(n,k)$ is known and equals the common value. The integer $d$, $d \leq d_{\text{max}}(n,k)$, is an improvement for the lower bound in the table at parameter pair $(n,k)$ if $d > Lb(n,k)$; similarly: if $d > d_{\text{max}}(n,k)$, then $d$ improves the upper bound at $(n,k)$ when $d < Ub(n,k)$.

Using pseudo-PASCAL we want to define the boolean function $\text{IsImprovement}$ to catch this fact. Before doing so we introduce the notion of a location triple, in order to distinguish between lower and upper bounds, and the notion of a location triple, indicating a position in the table by specifying a bound kind, a length, and a dimension. Furthermore, we use the term bound quadruple to stand for the quadruple of the bound’s kind, the length and dimension of the code type it refers to, and the actual bound on the minimum-distance, i.e. for a triple combined with a bound. From now on we shall often use bound in the sense of bound quadruple instead of a single integer: confusion is not very likely because of the context, but at times we shall say bound quadruple to make the distinction clear.

The function $\text{Bound}$ returns the bound quadruple corresponding to a location in the table. We also introduce the auxiliary function $\text{IsBound}$ as a boolean function for the proposition "$q$ is a proper bound quadruple", i.e. if $q = (b,n,k,d)$: "$d$ is a bound of kind $b$ for $d_{\text{max}}(n,k)$." $\text{IsBound}$ is a conceptual function only, for we could program it efficiently in PASCAL. Many problems concerning $d_{\text{max}}$ would have been solved. So when invoking $\text{IsImprovement}$, its pre-condition must be known to hold by some other means than the use of $\text{IsBound}$. We also need a way to change the table. We have chosen the one-point modification by assignment defined below as $\text{AssignBound}$. 


TYPE
  boundkind = (lower, upper) ;

  triple = RECORD ( indicating location in the table)
    b  : boundkind ;
    n, k : integer
  END ( triple ) ;

  quadruple = RECORD ( indicating bound and its location )
    t  : triple ;
    d  : integer
  END ( quadruple ) ;

VAR
  Lb, Ub: ARRAY [ 1.."inf", 1.."inf" ] OF integer ;
  ( both Lb and Ub are lower-left matrices )

FUNCTION Bound(t: triple): quadruple ;
  ( Pre-condition: 1 ≤ t.k ≤ t.n.
    Returns the current bound in the table at location t. )
BEGIN
  Bound.t := t ;
  WITH t DO CASE b OF
    lower: Bound.d := Lb(n,k) ;
    upper: Bound.d := Ub(n,k) ;
  END ( case )
END ( Bound ) ;

FUNCTION IsBound(q: quadruple): boolean ;
  ( Is q a proper bound quadruple? Conceptual function! )
BEGIN
  WITH q, t DO
    IF ( 1 ≤ k ) AND ( k ≤ n ) THEN
      CASE b OF
        lower: IsBound := ( d ≤ \max(n,k) ) ;
        upper: IsBound := ( d ≥ \max(n,k) ) ;
      END ( case )
      ELSE IsBound := false
  END ( IsBound ) ;

FUNCTION IsImprovement(q: quadruple): boolean ;
  ( Pre-condition: IsBound(q).
    Does q improve the current bound in the table at
    the corresponding position? )
BEGIN
  WITH q, t DO
    CASE b OF
      lower: IsImprovement := ( d > Lb(n,k) ) ;
      upper: IsImprovement := ( d < Ub(n,k) ) ;
    END ( case )
END ( IsImprovement ) ;
PROCEDURE AssignBound(q: quadruple); 
( Pre-condition: IsBound(q). Assign q as the new bound in the table. )
BEGIN
WITH q, t DO CASE b OF
lower: Lb(n,k) := d ;
upper: Ub(n,k) := d ;
END ( case )
END ( AssignBound );

REMARKS:
-- We could have defined the table of bounds to be a mapping of triples to integers. This was not done because the bound kind is usually fixed; it would result in longer expressions.
-- Since PASCAL lacks a RECORD constructor, we shall use the conventional mathematical notation for triples and quadruples. Often we shall omit the parentheses around a tuple, when it is the only parameter to a function or procedure, or when tuples are nested.
-- (22) is equivalent to IsBound(Pound(b,n,k)), for all b,n,k.
-- The pre-condition on AssignBound assures that afterwards Lb and Ub still form a table of bounds, i.e. satisfy (22).

There is one way of changing the table that particularly interests us: raising a lower bound or lowering an upper bound, it will be called improving the table (bringing the lower and upper bounds closer together, when they meet, dmax is known). Of course we never want to lower a lower bound (or raise an upper bound). In successive tables Lb(n,k) (n and k fixed) should be non-decreasing and Ub(n,k) non-increasing. This kind of change may appropriately be called the one-point Monotonic Modification. It can be defined in pseudo-PASCAL as follows.

PROCEDURE MM(q: quadruple); 
( Pre-condition: IsBound(q). Incorporate the bound quadruple q into the table. Post-condition: NOT IsNonBound(q) AND Bound(q) either be worse than before. )
BEGIN
IF IsImprovement(q) THEN IsBound(q)
END ( MM );

2.4 The Inference of the Table of Bounds

In section 2.1 bounds were treated individually, we described how one set of bounds can justify others. Now we can apply this to the ensemble of bounds as collected in the table.
For example combining (22,3) a 0.1, taking n = 0.1, k = 0, gives us
2. Theoretical Discussion

\[ d_{\max}(n+1,k) \geq \text{even ceil}(Lb(n,k)), \quad (23) \]

or equivalently

\[ \text{IsBound}(\text{lower},n+1,k,\text{even ceil}(Lb(n,k))). \quad (24) \]

This means that MM's pre-condition is satisfied, and hence the invocation

\[ \text{MM}(\text{lower},n+1,k,\text{even ceil}(Lb(n,k))) \quad (25) \]

produces a correct and possibly improved table of bounds. Such an improvement will be called an internal improvement. We say that the table is invariant with respect to (or closed under) P2's lower bound (or existence) formulation (P2_{lower} for short), if the table cannot be improved in this way, that is if NOT \( \text{IsImprovement}(\text{lower},n+1,k,\text{even ceil}(Lb(n,k))) \) holds for all \( n \) and \( k \). Applying the definition of \( \text{IsImprovement} \), this can also be written as: for all \((n,k)\) \( \text{even ceil}(Lb(n,k)) \leq Lb(n+1,k) \). Invariance implies that internal improvements are impossible. We now list the quadruples for which \( \text{IsBound} \) holds, and that can be derived from the construction techniques of section 2.1 using (22). All free variables are shown in parentheses following the construction's name. In general there is a lower and upper quadruple for each construction.

- \( \text{P2}(n,k) : (\text{lower},n+1,k,\text{even ceil}(Lb(n,k))) \) (upper,n-1,k,even floor(Ub(n,k)))
- \( \text{P3}(n,k) : (\text{lower},n-1,k,Lb(n,k)) \) (upper,n+1,k,Ub(n,k))
- \( \text{P4}(n,k) : (\text{lower},n-1,k-1,Lb(n,k)) \) (upper,n+1,k-1,Ub(n,k))
- \( \text{A}(n,k) : (\text{lower},n-Lb(n,k),k-1,ceil(Lb(n,k)/2)) \) (upper,n+2*Ub(n,k)+1,k+1,2*Ub(n,k))
- \( \text{E}(n,k) : (\text{upper},n+2*Ub(n,k)+1,k+1,2*Ub(n,k)) \) (lower,n-Lb(n,k),k-1,ceil(Lb(n,k)/2))
- \( \text{B}(n,k,s) : (\text{lower},n-\text{Ub}(n,n-k),k-\text{Ub}(n,n-k)+1,Lb(n,k)) \) (upper,n+s,k+s-1,Ub(n,k)) if \( \text{Ub}(n+s,n-k+1) \leq s \)
- \( \text{C}(n,m,k) : (\text{lower},n+m,k,Lb(n,k)+Lb(m,k)) \) (upper,n-m,k,Ub(n,k)+Lb(m,k))
- \( \text{D}(n,k,j) : (\text{lower},2*n,k+j,\text{MIN}(Lb(n,k),2*\text{Lb}(n,j)))) \)

REMARKS:

-- Invariance with respect to \( R_n(j,\ldots,s) \) is defined as:

For all sensible \( j,\ldots,s \): not \( \text{IsImprovement}(R_n(j,\ldots,s)) \)

-- Each of these quadruples has a length-dimension pair differing from that of all bounds (Lb or Ub) involved in its construction, whatever the values of the free variables.

-- Invariance with respect to \( \text{P2}_{lower} \), \( \text{P3}_{lower} \) and \( \text{P4}_{lower} \) implies monotonicity of Lb similar to that of \( d_{\max} \) as in (17), (18) and (19). The same holds for Ub in case of
invariance with respect to $P_{2\text{upper}}$, $P_{3\text{upper}}$ and $P_{4\text{upper}}$. In fact $d_{\text{max}}$ is possibly equal to $L_b$ or $U_b$ for all $n$ and $k$, as far as we know. When the table is closed, therefore, $L_b$ and $U_b$ share those properties with $d_{\text{max}}$ that depend on the closing constructions.

$\text{--Upper = Eizer.}$

$\text{--Upper has an exceptional formulation.}$

If the table is not closed under a set of constructions, then there is at least one construction $R_e$ and a set of values $j, s$ for its free variables such that $\text{IsImprovement}(R_e(j, s))$ holds. That is, an internal improvement is possible, it will be called a variance. Applying $MM(R_e(j, s))$ improves the table, and has as a consequence: not $\text{IsImprovement}(R_e(j, s))$ (this particular variance is removed, see second remark above), although it may cause other new variances. Any finite part $T$ of the table, however, can only take a finite number of improvements, since these are monotonic and reduce

\[ \text{SUM( (n,k) in T: Ub(n,k) - Lb(n,k) )}, \]

which is a non-negative integer. When the sum equals zero, this means that $d_{\text{max}}$ is completely known for that part $T$ of the table.

2.4. Disturbing Invariance and Restoring it

We aim at a table closed under a particular set of constructions (the choice is still to be made). How can invariance be obtained? The method suggested above is repeatedly finding a variance and applying $MM$. For a finite table this process will terminate, and will result in a closed table. The removal of one variance, however, may produce (many) others. A straightforward way of removing all variances is scanning the whole table, making whatever internal improvements that are possible, and rescanning the table until none are found for one complete scan. For a fairly large table this will mean that many constructions are tried without avail every time over and over again. What we need, is a systematic way of keeping track of all variances. This should not be too difficult if it is known what new variances can be introduced by a removal.

Once we have a closed table, invariance can only be disturbed by bound improvements from "outside" (i.e. not based on any of the constructions under which the table is closed), these improvements will be called external. We seek to restore invariance immediately after making a single external improvement. When the bounds in the table are not very tight, it could be more efficient to make all possible external improvements before restoring invariance. But only in the very beginning do we have loose bounds and consequently the opportunity for many improvements. We suggest the following recursive algorithm to restore invariance. It is only an informal attempt, so the terms occurring in it are not well-defined, but it has the right structure.
PROCEDURE Restore(t: triple) ;
( Remove all variances involving t as "input" )
VAR v: "vector" ; q: quadruple ;
BEGIN
FOR "all constructions R" DO BEGIN
  FOR "all occurrences of t in R" DO BEGIN
    v := "vector of values for free variables in R corresponding to this occurrence of t" ;
    q := R(v) ;
    IF IsImprovement(q) THEN BEGIN
      AssignBound(q) ;
      Restore(q.t)
    END { if }
  END { for }
END { for }
END { Restore } ;

When q is a possible external improvement, the calling sequence will be:

( IsBound(q) )
MM(q) ;
( all variances--if any--involve q.t as "input" )
Restore(q.t)
( the table is invariant )

Notice that MM cannot be used in the body of Restore. The calling sequence can be rewritten as:

( IsBound(q) )
IF IsImprovement(q) THEN BEGIN
  AssignBound(q) ;
  ( all variances--if any--involve q.t as "input" )
  Restore(q.t)
END { if }
( the table is invariant )

We see that this sequence also occurs inside the body of Restore. It can be eliminated in the following way, combining the implied one-point monotonic modification with the restoration of invariance. The resulting recursive procedure is called Update.
PROCEDURE Update(q: quadruple) ;
 ( Pre-condition: IsBound(q).
 Post-condition: q is incorporated in the table, AND
 no variances were added, i.e. the ones that were
 introduced have all been removed as well.
 )
 VAR v: "vector" ;
 BEGIN
 IF IsImprovement(q) THEN BEGIN
 AssignBound(q) ;
 FOR "all constructions R" DO BEGIN
 FOR "all occurrences of q.t in R" DO BEGIN
 v := "vector of values for free variables in R
 corresponding to this occurrence of q.t" ;
 Update(R(v))
 END ( for )
 END ( for )
 END ( if )
 END ( Update ) ;
 Its calling sequence is:

 ( the table is closed AND IsBound(q) )
 Update(q)
 ( bound q has been incorporated AND the table is closed )

 REMARKS:
 --The stacking mechanism involved in the execution of these
 recursive procedures keeps track of all potential variances.
 "IF IsImprovement" detects true variances (except for the
 outermost invocation of Update, where it checks the usefulness
 of the new bound), AssignBound eliminates one (again
 excepting the outermost invocation of Update, where it makes
 the external improvement, thereby possibly introducing the
 first variances), and the FOR-loops (with recursive call)
 "extend" the list of potential variances with the ones that
 may have been introduced by AssignBound. The initial list is
 trivial.
 --The post-condition of Update is to be proved under the
 assumption that each recursive invocation in the body termi-
nates in the given post-condition (like the step of an
 induction argument); that is why it is somewhat stronger
 than what the outermost invocation is to accomplish.
 --Because the table is considered unbounded, these procedures
 may never terminate. But they do not leave any variances,
 and they do not cycle (get stuck), the only problem is that
 they may "run away" (see section 3.2 for more on this
 problem). This can be understood by considering any finite
 part T of the table, where IsImprovement is modified to
 return false if q.t lies outside T. A termination argument
 for this situation was given at the end of section 2.3.
2.5 Propagation Rules

Some constructions have more than one input (in particular: B, C and D). When restoring invariance they have to be considered more than once (that is what the second FOR-loop in Update does). In constructions C and D the inputs are almost independent, so that the second input can be chosen in many ways even if the first is fixed. The notion of a propagation rule (or function) is a refinement of that of a construction, such that it has only one input. This makes no difference for P2, P3, P4, A, and E. B, C, and D, however, decompose into several propagation rules. The advantage of propagation rules is that they are all alike, this simplifies reasoning about, for example, the effect of applying constructions one after the other (see section 2.6).

We define a propagation rule to be any partial function \( P \), mapping bound quadruples into bound quadruples, possibly involving values of the table Lb and Ub, and which satisfies:

\[
\text{for all } q \in \text{dom}(P): \text{IsBound}(q) \implies \text{IsBound}(P(q)) \quad (26)
\]

where \( \text{dom}(P) \) is \( P \)'s domain. Since there is only one table, we shall not explicitly write \( Lb \) and \( Ub \) as arguments or parameters to the propagation function. The function argument will often be called input, the function value sometimes its consequence. By the application of the propagation rule \( P \) at the location \( s \) we mean the invocation \( MM(P(Bound(s))) \).

Propagation functions can be derived from construction techniques. We use the construction's identifier also as a name for the related propagation function (when there are more functions related to one construction, the distinction will be made by subscripting or priming ('')). The if-then formulation of the construction techniques is most suitable for the derivation, because of the implication in (26), and the occurrence of \( d_{\max} \) in the definition of IsBound. The derivation is straightforward. For instance, \( C'M \) was obtained by taking \( d = Ub(n,k) \) in the fifth proposition for the effect of construction C (see section 2.1), and afterwards substituting \( n \) and \( m \) for resp. \( m \) and \( n \).

\[
P2(b,n,k,d) := \begin{cases} 
\text{lower}, n+1, k, \text{even/ceil}(d) & \text{case } b=\text{lower} \\
\text{upper}, n-1, k, \text{even/loor}(d) & \text{case } b=\text{upper} 
\end{cases}
\]

\[
P3(b,n,k,d) := \begin{cases} 
\text{lower}, n-1, k, d-1 & \text{case } b=\text{lower} \\
\text{upper}, n+1, k, d+1 & \text{case } b=\text{upper} 
\end{cases}
\]

\[
P4(b,n,k,d) := \begin{cases} 
\text{lower}, n-1, k-1, d & \text{case } b=\text{lower} \\
\text{upper}, n+1, k+1, d & \text{case } b=\text{upper} 
\end{cases}
\]

\[
A(b,n,k,d) := \begin{cases} 
\text{lower}, n-d, k-1, \text{ceil}(d/2) & \text{if } b=\text{lower} 
\end{cases}
\]

\[
E(b,n,k,d) := \begin{cases} 
\text{upper}, n+2#d+1, k+1, 2#d & \text{if } b=\text{upper} 
\end{cases}
\]
2. Theoretical Discussion

Using propagation rules the notion of invariance can be redefined in an obvious way. Invariance of lower bounds w.r.t. construction D corresponds to invariance w.r.t. propagation rules D~ and D'~ for all j. With the new notion it is easier to pinpoint what a variance is, it is completely specified by a propagation rule R and a triple s, for which we have IsImprovement(R(Bound(s))). allowing an internal improvement. Using propagation rules, the procedure Update can be rewritten as follows.

\[ B_1(b,n,k,d) := (lower, n - Ub(n,n-k), k - Ub(n,n-k) + 1, d) \] if \( b = \text{lower} \)
\[ B_2(b,n,k,s) := (lower, n - s, n - k - s + 1, Ub(n,n-k)) \] if \( b = \text{upper} \)
\[ B_3(b,n,k,d) := (upper, n + s, k + s - 1, d) \] if \( b = \text{upper} \)

where \( s = \min(t : Ub(n+t,n-k+1) \leq t) \)
\[ B_4(b,n,k,s) := (upper, n - k, Ub(n - s, n - k + 1)) \] if \( b = \text{upper} \)

\[ C_m(b,n,k,d) := (lower, n + m, k, d + Ub(m,k)) \] case \( b = \text{lower} \)
\[ (upper, n - m, k, d - Ub(m,k)) \] case \( b = \text{upper} \)
\[ C'_m(b,n,k,c) := (upper, n - m - k, Ub(m,k) - c) \] if \( b = \text{lower} \)

\[ D_1(b,n,k,d) := (lower, 2n, k + j, \min(d, 2\times LB(n,j))) \] if \( b = \text{lower} \)
\[ D'_1(b,n,k,d) := (lower, 2n, k + j, \min(LB(n,j), 2\times d)) \] if \( b = \text{lower} \)

REMARKS:

-- "Case" indicates a definition by cases, "if" indicates a restriction on the function's domain.
-- The domains of these functions were not clearly defined; \( b, n, k \) and \( d \) are to be restricted to "values that make sense." The domain may be quite irregular, since it can depend on the actual values in the table.
-- \( B_3 \) requires explanation. Actually \( s \) should have been a parameter (so \( B_{3,s} \)), satisfying \( s \geq Ub(n+s,n-k+1) \). But since \( B_{3,s} \) follows from \( B_{3,s} \) when \( s < t \) (by P4), we might as well restrict ourselves to the minimal \( s \).
-- Notice that almost always the the bound kind of the consequence is the same as that of the input. There are two exceptions: \( B_2 \) (propagation from upper bound to lower bound) and \( C' \) (propagation from lower to upper bound).
-- In the propagation functions \( P_2, P_3, P_4, A \) and \( E \) no values of the table occur; they work directly from input to consequence.
-- Propagation functions \( A \) and \( E \) have disjoint domains and could therefore have been combined into one function.
-- For the same reason \( B_1 \) could have been combined with any of the other \( B_i \). This was not done for the sake of clarity.
-- \( B_1(b,n,k,lb(n,k)) = B_2(b,n,n-k,Ub(n,n-k)) \)
\( B_3(b,n,k,lb(n,k)) = B_4(b,n+s,n-k+1,s) \) (s as in \( B_3 \))
\( C_m(upper,n,k,Ub(n,k)) = C'_m(upper,m,k,Lb(m,k)) \)
\( D_1(b,n,k,lb(n,k)) = D'_m(b,n,j,lb(n,j)) \)

Using propagation rules the notion of invariance can be redefined in an obvious way. Invariance of lower bounds w.r.t. construction D corresponds to invariance w.r.t. propagation rules \( D_1 \) and \( D'_1 \), for all \( j \). With the new notion it is easier to pinpoint what a variance is, it is completely specified by a propagation rule \( R \) and a triple \( s \), for which we have IsImprovement(R(Bound(s))), allowing an internal improvement. Using propagation rules, the procedure Update can be rewritten as follows.
2. Theoretical Discussion

PROCEDURE Update(q: quadruple) {
( Pre-condition: IsBound(q).
Produce a table that incorporates the bound q, and
that has no new variances.
)
BEGIN
IF IsImprovement(q) THEN BEGIN
AssignBound(q) ;
FOR "all propagation rules R" DO
IF "q in dom(R)" THEN Update(R(q))
END ( if )
END ( Update )

A new problem is that propagation rules are not independent. That is, in order for Update—as defined above—to work correctly (indeed restoring invariance), the set of propagation rules cannot be chosen arbitrarily. Invariance w.r.t. B₁ is equivalent to invariance w.r.t. B₂, as is readily seen from the last REMARK above. But if invariance w.r.t. B₁ is desired, then B₂ has to be included as well. We need a complete decomposition of a construction into propagation rules. It is obtained by deriving a propagation rule for each occurrence of Lb (or Ub) in the construction, by performing a coordinate transformation such that we get Lb(n,k) (or Ub(n,k)), and then replacing it by d. Construction C, for example, gives two propagation rules for lower bounds that happen to be equivalent. The propagation rules above form complete decompositions.

2.6 Combining Propagation Rules

Propagation rules can be combined by simple functional composition. We shall write \( P;R \) to denote the function that maps q onto \( R(P(q)) \) (the semicolon was inspired by the sequential composition operator for statements as used in many programming languages). When P and R are propagation functions, then so is \( P;R \), i.e. \( P;R \) satisfies (26). We shall now discuss some relationships between the propagation rules given above and some of their composites. We introduce the binary relation \( \rightarrow \) on bound quadruples, indicating that one bound is at least as useful as another.

\[
q₁ \rightarrow q₂ \text{ is defined as } q₁.t = q₂.t \text{ AND } \begin{cases} q₁.d < q₂.d & \text{case } q₁.t.b = \text{lower} \\ q₁.d > q₂.d & \text{case } q₁.t.b = \text{upper} \end{cases}
\]  \hspace{1cm} (27)

For example, a post-condition of \( MM(q) \) is: \( Bound(q.t) \rightarrow q \), expressing that \( MM \) does not weaken the table. If \( q₁ \rightarrow q₂ \) holds, then \( IsBound(q₁) \) implies \( IsBound(q₂) \) (but not necessarily the other way round; for instance, take \( q₁ = (24,15,5) \) and \( q₂ = (25,15,6) \)). We define the identity propagation function \( \text{Id} \) by: \( \text{Id}(q) := q \). We shall list a number of relationships between propagation functions in the following format: \( P \circ R \), where \( r \) is \( \rightarrow, \leftarrow, \text{ or } = \) (equality of quadruples). This is meant to be read as: for all \( q = (b,n,k,d) \) in the intersection of \( \text{dom}(P) \) and \( \text{dom}(R) \), \( P(q) r R(q) \) holds. The postfix operator \# on a propagation
function is used to indicate "a sufficient number of times (20) composed with itself," it is an existential quantification inside the universal quantification over the argument q, i.e. the number may depend on q. Of course, proofs are required, but in most cases they are trivial, and have been omitted. For lines with # in front a proof is given below.

\[
P3 \circ P2 = \text{Id} \quad \text{if even}(d)
P3 \circ P2 \leftarrow \text{Id} \quad \text{in general}
P2 \circ P3 = \text{Id} \quad \text{if odd}(d)
P2 \circ P3 \leftarrow \text{Id} \quad \text{in general}
P2 \circ P4 = P4 \circ P2
P3 \circ P4 = P4 \circ P3
\]

\# P2 \circ A = A \quad \text{if odd}(d)
\# P2 \circ A \leftarrow A \circ P2 \quad \text{if even}(d)
P2 \circ A \leftarrow A \circ P2 \# \quad \text{in general}
P3 \circ A \leftarrow A
P4 \circ A = A \circ P4

\[
E \rightarrow P3 \circ P4
P2 \circ E = E \circ P2 \quad \text{if even}(d)
P3 \circ E \rightarrow E \circ P3 \circ P3
P4 \circ E = E \circ P4
\]

\[
B_1 \rightarrow P4 \circ P3
P2 \circ B_1 = P4 \circ P3 \quad \text{if even}(d)
P3 \circ B_1 = B_1 \circ P3 \circ P4 \# \quad \text{if P4-closed}
P4 \circ B_1 = B_1 \circ P4 \# \quad \text{if P3-closed}
P2 \circ B_2 = B_2 \quad \text{if odd}(d) \text{ and P4-closed}
P2 \circ B_2 = B_1 \circ P4 \quad \text{if even}(d) \text{ and P4-closed}
P2 \circ B_2 = B_1 \circ P4 \# \quad \text{if P4-closed, in general}
P3 \circ B_2 \leftarrow B_2 \quad \text{if P4-closed}
P4 \circ B_2 \rightarrow B_1 \circ P2 \quad \text{if P2-closed}
B_3 \rightarrow P4 \circ P3
\]

\# P2 \circ B_3 = B_1 \circ P2 \circ P4 \# \quad \text{if P4-closed}
P2 \circ B_3 = P4 \circ P3 \quad \text{if even}(d)
P3 \circ B_3 \circ P2 \circ P4 \# \rightarrow B_3 \quad \text{if even}(d) \text{ and}
P4 \circ B_3 = B_3

P2 \circ B_4 \text{ not reachable from } B_4 \text{ by } P2, P3, P4
P3 \circ B_4 = B_4 \circ P3
P4 \circ B_4 \rightarrow B_4 \circ P3 \quad \text{if P3 closed}

\[
P2 \circ C_4 = C_4 \circ P2 \quad \text{if even}(Lb(m, k))
P2 \circ C_4 \leftarrow C_4 \cdot 1 \quad \text{if odd}(Lb(m, k)) \text{ and P2-closed}
P3 \circ C_4 = C_4 \circ P3
P4 \circ C_4 \rightarrow C_4 \circ P4 \quad \text{if P4-closed}
C_4 \leftarrow C_4 \cdot 1 \circ P2 \quad \text{if } \text{Lb}(m, k) = \text{Lb}(m-1, k)
C_4 \leftarrow C_4 \cdot 1 \circ P3 \quad \text{if odd}(\text{Lb}(m, k)) \text{ and P2-closed}
\]
2. Theoretical Discussion

\[P2D_j \rightarrow D'_j;P2P2 \quad \text{if P2-closed} \]
\[P3D_j \rightarrow D'_j;P3P3 \quad \text{if P3-closed} \]
\[P4D_j \text{ no relation given} \]
\[P2D'_j \rightarrow D'_j;P2P2 \quad \text{if P2-closed} \]
\[P3D'_j \rightarrow D'_j;P3P3 \quad \text{if P3-closed} \]
\[D_j \rightarrow D'_j \quad \text{if } \text{Lb}(n,j) \leq d \]
\[D'_j \rightarrow D_j \quad \text{if } \text{Lb}(n,j) \geq d \]

Proofs for relationships prefixed with \#:

\[\text{(P2'A)} (\text{lower},n,k,d)\]
\[\{ \text{def. of } \text{ and of P2 } \}
\[== \text{A}(\text{lower},n+1,k,\text{even floor}(d))\]
\[\{ \text{def. of A } \}
\[== (\text{lower},n+1,\text{even floor}(d),k-1,\text{ceil}(\text{even floor}(d)/2))\]
\[\{ \text{use (5) and (4) (idempotency of ceil) } \}
\[== (\text{lower},n+1,\text{even floor}(d),k-1,\text{ceil}(d/2))\]
\[\{ \text{case analysis: even/odd d, and (7) } \}
\[== (\text{lower},n+d+1,k-1,\text{ceil}(d/2)) \quad \text{case even}(d)\]
\[== (\text{lower},n-d-1,k-1,\text{ceil}(d/2)) \quad \text{case odd}(d)\]
\[\{ \text{def. of A } \}
\[== \text{A}(\text{lower},n,k,d) \quad \text{case odd}(d)\]

\[\text{(A1P2)} (\text{lower},n,k,d)\]
\[\{ \text{def. of } \text{ and of A } \}
\[== \text{P2}(\text{lower},n-d,k-1,\text{ceil}(d/2))\]
\[\{ \text{def. of P2 } \}
\[== (\text{lower},n-d+1,k-1,\text{even floor}(\text{ceil}(d/2)))\]
\[\{ \text{8 and def. of } \rightarrow \}
\[== (\text{lower},n-d+1,k-1,\text{ceil}(d/2))\]
\[\{ \text{above } \}
\[== (\text{P2'A}) (\text{lower},n,k,d) \quad \text{case even}(d)\]

\[\text{(P2'Ba)} (\text{upper},n,k,d)\]
\[\{ \text{def. of } \text{ and of P2 } \}
\[== \text{B}\text{a}(\text{upper},n-1,k,\text{even floor}(d))\]
\[\{ \text{def. of B}\text{a } \}
\[== (\text{upper},n-1,k+1,\text{even floor}(d))\]
\[\text{where } s := \text{MIN } S, S := ( t : \text{Ub}(n+t-1,n-k) \leq t )\]

\[\text{(B}\text{a};P2P4*) (\text{upper},n,k,d)\]
\[\{ \text{def. of } \text{ and of B}\text{a } \}
\[== (\text{P2P4*}) (\text{upper},n+u,k+u-1,d)\]
\[\text{where } u := \text{MIN } U, U := ( t : \text{Ub}(n+t,n-k+1) \leq t )\]
\[\{ \text{def. of } \text{ and of P2 } \}
\[== (\text{P4*}) (\text{upper},n+u-1,k+u-1,\text{even floor}(d))\]
\[\{ \text{def. of P4, taking * as i times } \}
\[== (\text{upper},n+u+i-1,k+u+i-1,\text{even floor}(d)), \text{ where } i \geq 0\]

If the table is P4-closed, then \(\text{Ub}(n+t,n-k+1) \leq \text{Ub}(n+t-1,n-k)\). So \(S\) is contained in \(U\), and hence \(u \leq s\). By taking \(i = s-u\), we get \(\text{P2'Ba} == \text{B}\text{a};P2P4* \quad \text{if P4-closed.}\)

(End of Proofs)
2. Theoretical Discussion

REMARKS:

- Let \( R(i,j,k) \) stand for a composition of \( P2 \), \( P3 \), and \( P4 \) occurring resp. \( i \), \( j \), and \( k \) times in any order. All \( P4 \)'s can be shifted to the right since they commute with both \( P2 \) and \( P3 \). By repeatedly "cancelling" adjacent \( P2 \)'s and \( P3 \)'s we get

  \[
  R(i,j,k) \leftarrow i-j \text{ times } P2 \quad i \text{ times } P4 \quad \text{if } \ i \geq j
  \]

  \[
  R(i,j,k) \leftarrow j-i \text{ times } P3 \quad i \text{ times } P4 \quad \text{if } \ j \geq i
  \]

2.7 Associating References

With each bound we associate a reference, indicating the bound's justification. A bound is said to be internally justified, if it can be derived by a construction, possibly involving other bounds in the table (it is the result of a propagation rule or \( P1 \)). The construction's name may serve as reference in this case. Other (ad hoc) proofs of bounds will be called external justifications: the article containing the proof, or its originator's name, can be used as reference. A bound can, of course, be justified in many ways, even internal and external justifications do not exclude each other. Circular justifications are not acceptable. For example, \([8,4,5]\) and \([9,4,6]\) justify each other by the addition of a parity check (\( P2 \)) and by puncturing (\( P3 \)), hardly satisfactory since neither exists.

The procedure \( \text{Update} \) can be given an extra parameter representing the reference for the bound. When the table is changed (by \( \text{AssignBound} \)), the associated reference can be updated as well, reflecting the justification of the new bound. \( \text{Update} \) will never give rise to internal justification cycles. Cycles involving an external justification are of course still possible: for instance, two articles each referring to the other for some intermediate results. Or even more dangerous: an article that constructs bound \( u \) via an ingenious method using, it says, bound \( v \) from the table. What if \( u \) is incorrect because \( v \) does not occur in the table, but the incorporation of \( u \) has as an indirect consequence \( v \), so that after the update \( v \) does occur in the table? The program can not be expected to detect these flaws. There are, however, some additional problems with references. These will be discussed below and in section 3.2.

We prefer the internal justifications \( P1 \) through \( P4 \) to others, because they are simpler. The corresponding construction is easier to carry out, and the other bound involved— if any— has length and dimension differing by at most one from those of the bound justified. In the table as presented in appendix 1, the references \( P1 \) through \( P4 \) have been omitted, they are called implicit references, the others explicit. This was done in order to make the table better readable. There are many of these, and \( P4 \) "commutes" with \( P2 \) and \( P3 \) (see section 2.6), so that in most cases the choice would seem arbitrary. E.g., \([6,3,3]\) can be derived from \([8,4,4]\) by \( P3 \text{IP4} \) (via \([7,4,3]\)), or by \( P4 \text{IP3} \) (via \([7,3,4]\)), so both \( P3 \) and \( P4 \) are possible as reference on \([6,3,3]\). The interpretation of an implicit reference requires a little more attention, see section 4.1 for a recipe. The preference of
implicit references complicates the updating process, because now references may have to be changed not only when a bound is improved, but also when it is equalled by a P2, P3, or P4 consequence. In the latter case we shall want to replace the existing reference by an implicit reference (called *wiping* a reference for short). Some care is needed with reference wiping, in order not to wipe the external reference that started the update (see section 3.2 Ad (iii)).

Although we are anxious to avoid circular justifications, it appears that reference cycles may have to be accepted. Starting from a bound, repeatedly following its reference to another bound (any of the others, if more than one is referenced, as with B, C and D), should ideally end at a bound that is justified externally, or by P1. Imagine, however, a table that allows an improvement at location s, which by a propagation chain improves itself during the restoration of the table's invariance. What should be the reference at s, the external reference, or the last reference of the improving chain? Both are required to reconstruct what has happened. So the history of a table of bounds cannot be neglected. Self-improvement can be detected, but it is only one manifestation of the problem. What about an improvement u that together with bound v constructs bound w, where a consequence of w makes an improvement (or wipes a reference) somewhere in the justification chain for v. This is much harder to detect dynamically during updating. As far as we know (by visual inspection aided with the reports, like in Appendix III), there are no reference cycles in our table. A program could be written to verify this.
3. Pragmatic Considerations

In this section we shall deal with more practical issues, while postponing implementation details to section 4.

3.1. Our Choice of Propagation Rules

As already mentioned in the introduction, invariance under more propagation rules generally implies tighter bounds, but also requires more effort to compute the closure. It does not pay to include a propagation rule that is no stronger (in the sense of (a) having a larger domain, or (b) the relation \( \rightarrow \)) than all the others and their composites.

We used the following propagation functions in the construction of our tables: \( P_2, P_3, P_4, A, B, C \) (restricted to \( b = \text{lower} \)), \( D, D', E \). The upper bound consequences for constructions C and D were omitted, because they did not seem to be very effective, but would give rise to many additional propagation rules; checking all of them for every improved upper bound seemed a waste of computing time. Their ineffectiveness was not proved, but some unsuccessful attempts at internal improvements using them on tables generated without them convinced us. By the way, inspection of the updated table also reveals that constructions C and D have lost (most of) their power for improving lower bounds as well, since even the maximal increase of a lower bound (by making it equal to the upper bound) does not introduce a variance w.r.t. constructions C and D. The contribution to lower bounds by construction B is not very impressive, neither that of A when compared to the amount of upper bound consequences for B and E.

3.2. The Updating Algorithm Revisited

When we designed the procedure \( \text{Update} \) in section 2, there were several aspects that we ignored, and that turn out to be problematical. (i) The table that we intend to maintain is finite, so the artificial boundary comes into play. (ii) \( \text{Update} \) is inefficient. (iii) The updating of associated references has to be incorporated.

Ad (i).

The finiteness of the table is demanded by our storage facilities, and is required for the termination of \( \text{Update} \). Therefore we introduce an artificial boundary by restricting the length parameter \( n \) to values less than, say, 128; this brings along some new problems. Bounds (codes) that fall outside the artificial boundary can be ignored as far as invariance is concerned, although we should not totally neglect them if we want the best possible bounds. Especially not if they are close to the boundary, because by using a construction technique we might very well derive a bound improvement within the table. How far outside does it make sense to keep this up? The problem is that it may take
several applications of propagation rules to get inside; twice
puncturing, for instance, seems worth the trouble.

The artificial boundary also causes a referencing dilemma. What
reference do we associate with a bound u that is the result of
propagation rule P applied to a bound v outside the table
(justified by ZZ)? Neither 'P' nor 'ZZ' alone is sufficient to
trace u to its origins, since its justification chain can never
be followed further than v (because bounds outside the artificial
boundary are not stored, there is no reference attached to them).

Ad (ii).

_Update_ considers all possible chains of improvements, it
attempts to extend the chain by using a propagation rule. This
means that starting from the place where it makes its first
improvement, all possible composites of one or more propagation
functions are applied. A composite P;R is not considered if P did
not result in an improvement, since the table is assumed invar-
iant before the outermost invocation of _Update_, hence R cannot
improve if P didn't. This already cuts the number of interesting
composites considerably, but in the light of the properties in
section 2.6 many more composites can be dropped. For example,
one of P2*P3*P4*IA need be considered, if all of AiP2*P4* are
investigated as well.

Ad (iii).

Our preference for implicit references and abhorrence of
(too obvious) reference cycles poses another problem. The fol-
lowing version of _Update_ is unacceptable.

PROCEDURE _Update2_(q: quadruple; r: reference) { ( no good )
BEGIN
   IF IsImprovement(q) THEN BEGIN
      AssignBound2(q,r) { ( make improvement )
      FOR "all propagation rules R" DO { remove variance )
         IF "q in dom(R)" THEN _Update2_(R(q),'R')
      END { improvement )
   ELSE IF "q == Bound(q,t)" THEN
      IF "r is an implicit reference" THEN
         AssignBound2(q,r) { replace reference )
   END { Update2 } ;

   n,k: 6 | 7 |
   | 6 | | |
   | 1 | |
   | 1 | |
   22 | 9 | 8 |
   | 1 | |
   | 1 | |
   23 | 10 | 9-9 |
   | 1 | |
   | 1 | |
   | 1 | |
   24 | 10 | 9-10 |
   | 1 | |
   | 1 | |

   n,k: 6 | 7 |
   | 6 | | |
   | 1 | |
   | 1 | |
   22 | 9 | 8 |
   | 1 | |
   | 1 | |
   23 | 10 | 9 |
   | 1 | |
   | 1 | |
   24 | 10 | 10 |
   | 1 | |
   | 1 | |

25

j. Pragmatic Considerations

3.
It is not acceptable because the following scenario is possible. The bound \( u = (\text{lower}, 24, 7, 10) \) is to be incorporated with reference 'T3' in the table shown on the left (it was taken from the table in [A1]; for an explanation of the table's format see section 4.1). This bound \( u \) has as P3-consequence \( v = (\text{lower}, 23, 7, 9) \), which is also an improvement. The P4-consequence of \( v \) is \( (\text{lower}, 22, 6, 9) \), which only equals the bound with reference 'G.' The scoundrel is, however, \( v \)'s P2-consequence \( (\text{lower}, 24, 7, 10) \), which equals the original bound \( u \). When \( \text{Update2}(u, 'T3') \) is invoked, the reference 'G' is wiped as desired, but the reference 'T3' soon also disappears after being entered correctly. The table shown on the right is what we would have liked. In its zeal to apply all propagation rules at each improvement \( \text{Update2} \) may introduce an obvious two-reference cycle (involving P2 and P3); notice that the bounds are correctly justified, only the associated references will be circular. In order to prevent this, \( \text{Update} \) will have to know something about its calling history, for instance via an additional parameter, but we propose a different solution.

Our approach consists of restoring invariance w.r.t. P2, P3, and P4 nonrecursively prior to the recursive application of other propagation rules. This is fairly simple owing to their composition properties (see section 2.6), and has several advantages. P234 will be used to stand for P2, P3, and/or P4. P234 invariance can be established much more efficiently when done nonrecursively, furthermore it allows P234 composition properties of other propagation rules to be taken into account, thereby further increasing efficiency. This is all we shall do about (iii). Two-reference cycles can also be easily avoided without strengthening the obligations of \( \text{Update} \), thereby solving (iii). And it enables us to take a fair stand on bounds outside the artificial boundary: to get inside, P234 will be applied repeatedly (composing them nonrecursively), and of the other propagation rules those that decrease the bound's length will be applied recursively; this deals with one of our concerns expressed in (i).

The obligations of this new procedure \( \text{Update} \) have to be formulated carefully, its implementation is a delicate affair. Instead of giving a fully detailed description of \( \text{Update} \) with all assertions required for a correctness proof, we shall discuss some of the problems encountered, and our way of attacking them. \( \text{Update} \) roughly gets the following structure, which can be viewed as a transformed version of the original procedure.
3. Pragmatic Considerations

PROCEDURE Update(q: quadruple; r: reference); 
(a) {pre-condition: ..., post-condition: ... }
VAR V: SET OF quadruple; { see (c) and (e) }
BEGIN
  V := []; { V becomes the empty set }
(b) IF "q is interesting" THEN BEGIN
  (c) "make improvements on account of P234, seeing to it that
      P234 invariance is finally restored again, thereby
      possibly introducing other variances; add each such
      improvement to V; also take care of the proper
      references, possibly wiping some" ;
  (d) FOR "all non-P234 propagation rules R" DO
      (e) FOR "all quadruples p in V" DO
      (f) IF "R is applicable to p and required"
      (g) THEN Update(R(p), 'R') ;
  END (if)
END (Update);

Ad (a).
Update's pre-condition is changed to "the table is closed under
P234" AND "(IsBound(q) on account of r" OR "q lies outside
the natural boundaries"), bounds outside the table simply being dis-
carded. The reason for doing so is that q does not belong to the
domain of propagation function R if R(q) falls outside the natu-nal boundaries, part of the test "q in dom(R)" can now be defer-
ed to a recursive call on Update, so that propagation functions
can be applied naively. Update's post-condition is strengthened
to "incorporate q" AND "do not introduce new variances" AND "the
table is closed under P234." Closure of the table under P234 is
now an invariant of the procedure Update, this is helpful in (c)
and (f).

Ad (b).
Because of (i) Update only takes action if "q is within all
boundaries" AND "q is an improvement, OR "q is within the natural
boundaries, but outside the artificial boundary." In the latter
case an attempt will be made to derive bounds closer to or within
the artificial boundary. No constructions are considered that
first lead away from the artificial boundary, even though it is
possible that this would later result in something interesting;
these detours can not be done for lack of a good criterium to
select a finite number of promising composites (termination
requirement).

Ad (c).
Due to the composition properties for P234 (see section 2.6, esp.
the closing REMARK) and the assumed closure under P234 (see (a))
it is fairly easy to find all interesting P234* consequences of q
(* indicating repetition), but the artificial boundary requires
some care. We have chosen the following strategy. P3 is applied
zero or more times to q until it is useless, to each of these
results P4 is applied zero or more times (until useless). P2 is
applied once or more until useless, to each result P4 is again
applied repeatedly. In this way all interesting P234* conse-
quences of q (q's P234* area for short) are generated, including
3. Pragmatic Considerations

q itself. Each improving P234* consequence is to be added to the set V, so that all remaining variances can later on be removed recursively (see (e)). This set V is an admistration local to each invocation of Update. Its representation in PASCAL is somewhat problematic, since sets of records are not allowed. Actually, maintaining a set of triples instead of quadruples suffices, the table can be consulted for the associated distance. Such a set of triples can be represented by its characteristic function. Under (e) we discuss a method of doing this in one global variable; it introduces the possibility of interference between these local administrations, but this turns out to be beneficial.

If a P234* consequence is useless but gives the same bound as in the table, then the associated reference can be wiped (CondWipeRef in appendix IV; special care is needed when q lies outside the artificial boundary). P2 is never applied after P3, nor the other way round, so the two-reference cycles mentioned in (ii) will not occur. The reference associated with q will be r, q's P234* consequences get an implicit reference. If, however, q lies outside the artificial boundary, but (some of) its P234* consequences improve the table, then we want an explicit reference somewhere, giving at least a hint about how the improvement arose (see (i)). The current version of the updating algorithm is rather naive in this respect, and puts far too many explicit references on the artificial boundary, superfluous ones being removed manually.

In order to deal with the artificial boundary, it is good to have an idea of the P234* area's anatomy. On all the diagonals generated by P4 the bound is constant (equal to q.d); in P2's direction it is also constant (but possibly differing by one from q.d); in the direction of P3 it is steadily decreasing (increasing) by one on each step for lower (cq. upper) bounds. It is not necessary to generate that part of the P234* area outside the artificial boundary, so in our program is incorporated a straightforward method to skip that part.

Ad (d).
This FOR-loop can be unrolled and each non-P234 propagation rule dealt with separately in an ad hoc fashion.

Ad (e).
All the variances introduced by (c) have their inputs in q's P234* area, as registered in the set V. Previously there was only one such input (viz. q itself), therefore the stacking mechanism kept track of the location of all (inputs of) variances during the recursive calls (g), obviating the set V. We want to represent V without introducing too much overhead, and such that it can be easily traversed. Is it feasible to reconstruct q's P234* area from q alone? Well, q is indeed kept intact by the stacking mechanism, but the table has possibly undergone radical changes as a result of the recursive calls. It would be fairly easy to find an encompassing area; it might, however, be much too large, giving rise to many unneeded applications of propagation rules.
We have chosen to mark q's P234* area in the table while doing (c), this amounts to representing V by its characteristic function. With each bound in the table there is--besides the current bound on the minimum distance--a reference, and a mark. At first one might think of a boolean mark, but the recursive nature of Update introduces enough problems to drop this idea (the q's alone do not give enough clues to distinguish between the local V's, which may be adjacent). We use integer marks: an invocation of Update marks with its recursion depth (1 and up), 0 indicating unmarked. That overlapping V's cannot be represented in this way is no problem, a variance needs to be removed but once. V's grow only at (c) on the deepest recursion level, which takes precedence in case of overlap. The traversal of the set V at (e) is accomplished by starting from q (itself a member of V), using the P234 rules for stepping, and using the marks to decide when to stop. In a certain sense a P234* area is convex and, hence, connected. With the way we finally implemented marking, it is not necessary to remove the marks afterwards (a selected few beyond the P234* area's boundary are removed while marking), but one restriction is that the P234* area has to be traversed in the same way as it was generated under (c). When q lies outside the artificial boundary, no marking is done. Due to our choice of propagation rules only q and not its entire P234* area is required for the application of rules that decrease the length parameter (and for the removal of variances introduced by q's P234* consequences). The associated reference will be that of the propagation rule, the reference with the bound outside the artificial boundary is lost.

Ad (f).
Applicability of a propagation rule R to bound p should together with (b) cover the text "p in dom(R)." Only undefined computations have to be prevented by (f), like accessing a bound outside the table; resulting in a negative dimension is alright, and is eliminated by (b) one recursion level deeper. The properties of composition (see 2.6) allow us to reduce the number of applications, because Update (on the next level) always applies P234 repeatedly. For example, rule A need only be applied to q and not to the other bounds in its P234* area. It is also nice to know that the table is closed under P234 prior to the application (see (a) and (c)), so that even more properties can be used. Invariance w.r.t. Besides, P234 is a more natural state of the table since it expresses certain monotonicity properties. For instance, in the light of the properties for Cm and closure under P234, Cm only has to be applied to q and its P4 consequences, and only for those m that satisfy \( \text{odd}(Lb(m-1,k)) \).

Ad (g).
We have usually put (e), (f) and (g) together in one (or more) ad hoc procedure(s) for each non-P234 rule, the supplied reference in (g) is simply a constant, and is always an explicit reference.
3.3. Generating an Initial Invariant Table

At first we thought of using a trivial initial table (one that is easily generated), and using Update to turn it into a table closed under our selection of propagation rules. Update as described above requires the table to be closed under P234, so that is also a prerequisite of any initial table. Two trivial tables suggest themselves. The most trivial table has not even incorporated P1: \( L_b(n,k) = 0 \), and \( U_b(n,k) = \text{"infinite,"} \) for all \( n \) and \( k \). This table is also invariant under all the other propagation rules. It should be turned into an initial table by incorporating P1, which can be done by using Update to introduce the bounds: \( (\text{lower,MaxN,1,MaxN}), (\text{lower,MaxN,MaxN,1}), \) and \( (\text{upper,1,1,1}) \). The representation of \"infinite\" is troublesome, but the representation for the table as used in the programs makes it even less interesting (see section 4.2), since the bounds on the natural boundary \( n = k \) would initially not be equal. Another quite trivial table is:

\[
\begin{align*}
L_b(n,k) &= \begin{cases} 
/ n & (k = 1) \\
2 & (1 < k < n) \\
\setminus 1 & (k = n)
\end{cases} \\
U_b(n,k) &= n-k+1 \quad (1 \leq k \leq n)
\end{align*}
\]

It is closed under P234, even P1 is incorporated, and it does not have the representation problems of the previous one, but it takes a careful analysis to find out which improvements have to be made for it to become closed under the other propagation rules. Since we did not use this table either, we shall not go through the search for a (minimal) set of improvements by considering all the variances in this table. The reason for discarding this alternative as well is that it takes Update an enormous amount of time to compute the closure. This is not so strange because the table allows for a lot of improvement, the point is that each tiny improvement causes an avalanche of other small improvements. Update does not wait for the big catch, and consequently is running through a huge number of propagation chains, piling one improvement up another, the final value at a particular location being obtained only after what seems to be the maximal number of minimal improvements.

So we abandoned the idea of a trivial initial table. We would have to invest some effort in the initial table itself, instead of letting Update do all the work. We came up with a one-pass dynamic initialization, determining once for each location in the table what would be the best value from the other values that were already available, by applying the constructions at hand. Any variances--when known--can be removed afterwards by Update.

The order of this single pass over all locations leaves some freedom. We have chosen for the natural writing order, starting with length equal 1 and working downwards (increasing length), while for each length doing all dimensions in increasing order; all the lower bounds are done first. For each location all bounds with the same length and smaller dimension, and all bounds with
smaller length (any dimension) are available. So, for lower bounds we can consider the constructions P2, C, and D (followed by P3 or P4 if the length is odd); for upper bounds P3, P4, B (partly), and E. This implies invariance of the initial table w.r.t. these constructions. It is not so difficult to prove (by induction on the number of computed entries) that the lower bounds will be closed under P3 and P4 as well, similarly for upper bounds under P2.

While filling in the lower bounds any variances with respect to A can be detected and written to a file, so that they can be subjected to an Update afterwards. During the generation of upper bounds variances under B for lower bounds and some upper bounds are detectable in the same way. It turns out, however, that the table generated in this way is immediately closed under our selection of propagation rules, so the separate Update-pass is not needed. We have not proved that this is generally the case.
4. Implementation Details

4.1 Interpreting a Table (and other user aspects)

Since the table is fairly large, it is impossible to find out by simple visual inspection what has and what has not been changed as result of a series of updates. As an aid to the interpretation of an updated table, some statistics are maintained, and during each update session a report is generated, indicating major events. These events include: a bound is completely known for a particular parameter pair, along an entire column, or along an entire row; previous bound with external reference has been indirectly equalled (the reference is wiped out), has been superseded. See Appendix III for the report generated while producing the updated table.

Each new bound is checked before it is incorporated in the table. This is done to safeguard the table's integrity, because it is next to impossible to undo a faulty update. It is always possible, of course, to corrupt the table by feeding invalid but likely bounds. Ideally IsBound should hold before AssignBound is done, but in its stead we enforce the more computable NOT IsViolation. In case of a violation the program is aborted if there is reason to believe the internal table to be corrupted (i.e. if the table was modified since the outermost invocation of the Update that caused the violation), otherwise the violating bound is discarded.

FUNCTION IsViolation(q: quadruple): boolean;
( Is bound quadruple q in conflict with the other bounds? )
BEGIN
WITH q, t DO CASE b OF
  lower: IsViolation := (d > Ub(n,k));
  upper: IsViolation := (d < Lb(n,k));
END ( case )
END ( IsViolation );

The table is partitioned in blocks for printing. Each block has 25 (or 27) rows and 20 columns of bound pairs. This is about the maximum amount that fits on one page of our lineprinter (66 lines of 132 columns), when a block has been formatted with horizontal and vertical separating lines every 5 bound pairs and each
bound is accompanied by its reference. After an update session there is an option to print all modified blocks with modifications "highlighted." A block is identified by a pair of block coordinates, these are shown in the upper and lower right hand corners. The code's length is plotted vertically, its dimension horizontally.

A bound pair is printed as follows. If the lower and upper bound are equal then d_max is known, its value is shown centered; otherwise both lower and upper bound are printed separated by a dash (-). A reference consists of a two-character string, an implicit reference is shown as two blanks. The reference associated with the lower bound is shown over the bound's value (and possibly a bit to the left); the upper bound's reference is similarly shown as a (right-hand) superscript. Between these references there is a highlighting symbol, a blank indicates that neither bound was modified during the update session; a less-than (greater-than) sign («,» think of arrows) indicates that only the lower (upper) bound was modified; an asterisk (*) means that both bounds were improved.

How can the references in a table of bounds be interpreted? An external reference (an explicit reference other than A through E) directly refers to the REFERENCE section. Keep in mind that any reference on the artificial boundary (n = 127) may have been produced by a P234 consequence of a bound outside that boundary (see section 3.2 (c)). The list below should help in interpreting internal references. Notice that such a reference does not always uniquely determine the other bound(s) involved, furthermore, it may require a search to find them. An implicit reference--one of P1 through P4--has to be reconstructed since it was omitted from the table. We now give a recipe for the reconstruction of an implicit reference. It is not unique, indeed that was one of the reasons for their omission. If the bound lies on a natural boundary (n = 1, or n = k), its missing reference is P1. Otherwise, first try P4, next try P3 (if the bound is odd it must work, since P2 always results in an even bound), then try P2; take whichever worked first. If none worked, then the table is not closed under P234.

Once the internal reference is known it can be traced to other bounds. If this tracing is repeated--by taking any of the other bounds involved as new bound--it should always lead to P1, an explicit reference, or outside the artificial boundary. If it does not, then there seems to be something amiss with the references. In fact, the tracing process can only cycle for ever in that case, so there is a reference cycle. In general, these are hard to avoid, but in practice they have--as far as we know--not appeared in our tables. For an example of a reference cycle try to trace the bound (upper,63,40,20) in the table in [A], actually the explicit reference 'K' is missing there. In our table such a two-reference cycle cannot occur, implicit references can always be traced to P1 or an explicit reference.
4. Implementation Details

Internal reference

---

**at (lower, n, k)**

<table>
<thead>
<tr>
<th>Locations of bounds referred to</th>
<th>at (lower, n, k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>(lower, n-1, k)</td>
<td>(lower, n-1, k)</td>
</tr>
<tr>
<td>(lower, n+1, k)</td>
<td>(lower, n+1, k)</td>
</tr>
<tr>
<td>(lower, n+1, k+1)</td>
<td>(lower, n+1, k+1)</td>
</tr>
<tr>
<td>(lower, n+2, k, n+k+1)</td>
<td>(lower, n+2, k, n+k+1)</td>
</tr>
<tr>
<td>(lower, n+S, k+S-1)</td>
<td>for some S with S ≥ Ub(n+S, k+S-1)</td>
</tr>
<tr>
<td>(lower, m, k), (lower, n-m, k)</td>
<td>for some m ≥ k</td>
</tr>
<tr>
<td>(lower, ceil(n/2), j), (lower, ceil(n/2), k-j)</td>
<td>for some j &lt; n</td>
</tr>
</tbody>
</table>

---

**at (upper, n, k)**

<table>
<thead>
<tr>
<th>Locations of bounds referred to</th>
<th>at (upper, n, k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>(upper, n+1, k)</td>
<td>(upper, n+1, k)</td>
</tr>
<tr>
<td>(upper, n-1, k)</td>
<td>(upper, n-1, k)</td>
</tr>
<tr>
<td>(upper, n-1, k-1)</td>
<td>(upper, n-1, k-1)</td>
</tr>
<tr>
<td>(upper, n-Ub(n, k), n-k-1)</td>
<td>(upper, n-Ub(n, k), n-k-1)</td>
</tr>
<tr>
<td>(upper, n-k), (upper, n-s, k-s+1)</td>
<td>where s = Ub(n,n-k)</td>
</tr>
<tr>
<td>(lower, m, k), (upper, m+n, k)</td>
<td>for some m</td>
</tr>
</tbody>
</table>

---

Examples of interpretation based on the table in Appendix I

- (upper, 12, 5, 4) H: see [H] for nonexistence of [12, 5, 3]
- (lower, 127, 16, 51) We: punctured [128, 16, 52] from [We]
- (lower, 38, 9, 15) A: residual code of [68, 10, 30]
- (lower, 85, 8, 40) B: construction B using [89, 11, 40] & (upper, 89, 78, 4)
- (lower, 90, 7, 44) D: (u, u+v) on [45, 1, 45] & [45, 6, 22]
- (upper, 65, 8, 30) E: Griesmer bound using (upper, 34, 7, 15)
- (upper, 18, 10, 4) B: via construction B using (upper, 18, 8, 6) & (upper, 12, 5, 4)
- (lower, 63, 12, 24): recipe (repeated) leads to [66, 18, 23] 0, equally good are [66, 14, 25] L, and [63, 16, 23] I (their P234* areas overlap)
- (upper, 23, 12, 7): recipe (repeated) leads to (upper, 16, 6, 6) E, equally good is (upper, 18, 10, 4) B

4.2 Inside Details of Programs

The programs BoundUpdater and InitialBt are listed in Appendix IV. A table of bounds is stored in two ways: during program execution it is in an (internal) ARRAY (TYPE BtArray), for more permanent storage a FILE is used (TYPE BtFile). BoundUpdater reads in a BtFile, prompts via standard OUTPUT, reads commands
4. Implementation Details

via standard INPUT, performs the specified operations on its internal copy of the table, and afterwards writes it out to a BtFile; it also produces a report via the FILE Report and a list of applied updates via the FILE List; table printout is via the FILE Pr. InitialBt generates a BtFile and a list of improvements on it, that can be used by BoundUpdater to obtain a closed table of bounds. BoundUpdater interprets commands one line at a time, the first non-blank on the line determines the type of operation requested, the rest of the line constitutes a parameter list. The following commands are available:

L: incorporate specified lower bound and restore invariance
U: idem for upper bound
S: save internal table on file and print statistics
P: print the block with specified coordinates
A: print all blocks
M: print all modified-but-not-yet-printed blocks
W: print all blocks in Wallpaper format
Q: do S, finish off, and quit program

The limitations of PASCAL and the limited resources available on the computer system running the software compelled us to do some things in a less straightforward way. In what way this influenced the design will be discussed next, along with other aspects that are in want of explanation. To begin with, the source text of BoundUpdater is divided in three parts, viz. BoundUpdater, UpdateBt, and PrintBt; the latter two are textually included in the first by the compiler. With each bound in the table there is associated a two-character reference, an integer mark, and a boolean indicating whether the bound was changed during the update session (see the TYPE BTElement). The latter two are only used during program execution, but are nonetheless saved on the BtFile as well. Notice that the reference is not of TYPE Reference, for otherwise BTElement would not be packed in one word (6 bytes) because of alignment restrictions.

The TYPES triple and quadruple were not used since PASCAL lacks a RECORD constructor and does not allow a RECORD to be returned as function value. Hence the function Bound becomes useless, and was accordingly eliminated. The function IsBound is only a conceptual function, never intended to be put in the program. The procedure AssignBound does not explicitly occur by that name, it was swallowed by Improve, a procedure that also does some other things besides modifying the bound (e.g., checking for violation, updating the statistics, and reporting some major events).

PASCAL does not know triangular arrays, and two square arrays each half-filled require too much space (for \( n < 128 \), there are 16256 bounds, each occupying one word (6 bytes) of storage, that is about 92K bytes; doubling this would go too far). We decided to put upper and lower bounds in one square array Bt, lower bounds in the lower left triangle, upper bounds in the upper right triangle. That is, for \( 1 \leq k \leq n \) we have \( Bt[n,k].dist = Lb(n,k) \), and \( Et[k,n].dist = Ub(n,k) \). The diagonal
4. Implementation Details

is shared by both triangles, this is possible since the value of \( d_{\text{max}} \) is known on the boundary \( n = k \) (viz. equal 1), so upper and lower bounds are the same. This sharing does call for extra care when using the array \( B_t \).

Several nonstandard PASCAL features have been used. These will now be listed accompanied by a short comment.

Compiler Directives: a line starting with a dollar sign ($) is an instruction for the compiler, and not part of the program as such; these are used to enable or disable certain compile time options (like listing or range checking), and such things as conditional compilation ($ SET OMIT = and $ POP OMIT) and source file inclusion ($ INCLUDE).

Identifiers: upper and lower case are equivalent; all characters are significant, may contain (significant) underscores.

PACKED: our program relies on effective packing; without it the table would increase in size appreciably; this is of course implementation dependent, the (?) Pascal standard does not prescribe anything on packing.

FILE: associated with a file are a number of attributes (like its title if it is a disk resident file), the values of which are determined by: defaults, the program heading, file attribute equations at program invocation, and special procedures (see below).

STRING(n): predefined type; a variable length string of at most \( n \) characters.

MIN, MAX: functions with a variable number of parameters all of the same scalar type, returning the minimum resp. maximum.

RUNTIME: parameterless function returning the amount of process time (in seconds) elapsed since the program started; result is of type real.

IORES: pseudo function that can serve as a constant indicating an I/O result; its actual parameter is restricted to a number of identifiers that only have a special meaning in this context (like Ok and DataErr).

ABORT: parameterless procedure that immediately aborts the program's execution.

GETATTRIBUTE, SETATTRIBUTE: procedures to inspect and modify file attributes; they have a special syntax.

CLOSE: procedure to close a file; its first parameter is the file to be closed; its second parameter is optional and indicates a special action, it is restricted to a number of identifiers that only have this special meaning in this context (like SAVE and CRUNCH).

READ: when used as procedure an I/O error aborts the program; can optionally be used as a function that returns an integer indicating the I/O result without aborting.

OTHERWISE: an extension of the CASE statement; this reserved word is not followed by a colon; it is syntactically the last alternative; it can be followed by a statement list that will be terminated by the standard END; this alternative is semantically chosen iff none of the others apply.
5. Concluding Remarks

We used our bound table maintenance package to reconstruct Helgert and Stinaff's table in [A], which was to serve as the starting point for a more up to date table of bounds. For the reconstruction we extracted all bounds from their table that had external references (i.e. 'F' through 'Z', see Appendix II), and used these to update our initial invariant table (which is based solely on P1 and all possible constructions by P2, P3, P4, and A through E). We did not recheck the correctness of all these externally determined bounds, but we could not trace their lower bound 26 on $d_{max}(59,8)$, which has reference 'U.' It is the only 'U' in their table, and it refers to construction Y1, which is the basis of the general construction B; it might be that there is a [63,11,26]-code with a dual code of type [63,52,4] (we know $4 \leq d_{max}(63,52) \leq 5$, but that is not enough). Two external references are missing in their table: the lower bound 23 on $d_{max}(127,57)$ is on account of 'I,' and the upper bound 20 on $d_{max}(83,40)$ is from 'K.'

Our reconstruction of the table in [A] revealed some of its shortcomings. First, our program showed that the table in [A] is not closed under any of the constructions A through E, substantiating the concluding remark in [A] that the effects of B had not been fully evaluated. Using the same data, the program found the following (internal) improvements, written as (bound kind, length, dimension, bound on distance); their P2, P3 and P4 consequences are not mentioned, (**) indicates a second order improvement:

- (lower, 38, 9, 15) A
- (upper, 62, 12, 26) E
- (upper, 66, 9, 30) E
- (upper, 68, 17, 26) E
- (upper, 71, 13, 30) E
- (lower, 72, 14, 28) A
- (upper, 76, 24, 26) E
- (lower, 79, 7, 37) C
- (upper, 81, 16, 32) B
- (lower, 82, 7, 39) C
- (upper, 84, 16, 34) B

Second, there are some minor differences in the references, that were associated with each bound. Most remarkable are: (a) the omission of the superfluous references 'C' and 'I' for the lower bounds on resp. $d_{max}(105,18)$ and $d_{max}(31,21)$ (both bounds are also justifiable by P3 and P4, going back to resp. (lower, 117, 20, 43) 'X,' and (lower, 33, 22, 6) 'N'); (b) the avoidance of external justification for a number of bounds (see list below: the lower bounds were justified by D and P3; the upper bound by E). It must be noted that a different order of incorporating external improvements will, generally, result in different references, so (b) is not very telling.
5. Concluding Remarks

In our initial invariant table there is no contribution to lower bounds by constructions A and B, nor to upper bounds by construction B with $2k < n$. As a result the initial table generated in one pass from top to bottom (increasing length), and left to right (increasing dimension) is immediately invariant. We have no proof for this, it just happened to be the case.

The new table of upper and lower bounds on the minimum-distance of binary linear codes, as it appears in appendix I, is based on many improvements found in the literature (see REFERENCES and Appendix II). The updates on the table were made in chronological order of appearance of the improvement. The results in [Su] were improved by those in [PT1], the reference 'Su,' therefore, disappeared again after a short while. It was impossible to give full credit to all researchers involved, many results were found independently. At least one source is mentioned for each improvement incorporated. It appeared that some results claimed to be new, were derivable from older improvements by constructions P2 thru E, this may explain some unexpected or missing references.

Most noticeable in the updated table is that $d_{max}$ has been completely determined for dimensions 6 and 7 (due to [T31]). The new table is the result of 89 single updates, of which about 80 lasted. This resulted in 110 additional parameter pairs for which $d_{max}$ is now known. It is remarkable that (roughly) in the range $8 \leq k \leq 20$, $d_{max}(n,k)$ is most often known, when its value is a multiple of 4 (ranging from 8 to 40). For comparison we give the

**Computing times** (in seconds on Burroughs B7900)
- 3 for initial invariant table
- 264 for reconstruction of table in [A]
- 19 for updating table (includes conversion for print-out)

Now that the table of bounds is available in computer readable form, it can be used for other purposes than automatic updating and printing.

**Acknowledgments**

Chris Vos did most of the searching through the available literature of the past ten years. Henk van Tilborg dug up some interesting articles from his files, that might otherwise have been overlooked. Cees Hemerik suggested some improvements in the presentation of the programs, any imperfections, however, fall under my responsibility.
REFERENCES


[B] Dual code construction Y1 (see [M], and [MS] p.592), followed by repeated shortening.

[C] Concatenation (see [MS] p.76).

[D] The u,u+v construction (see [MS] p.76).


[P2] Parity check (see [MS] p.2/).

[P3] Puncturing (see [MS] p.28).


Selected references from [A]:


[F] Griesmer codes, see [E].


[J] Improvement on Johnson bound: S.M. Johnson, private communication, see [A] [27].


[R] Improvement on Johnson bound: R.J. McEliece and L.R. Welch, private communication, see [A] [29].


[U] (?) Construction Y1, see [M].


[Z] M. Karlin, private communication, see [A] [30].

Additional references:


INDEX

Auxiliary ceil/floor functions  6
Binary linear [n,k,d]-code  2
   parameters: length, dimension, minimum-distance  2
   type  2, 5
   existence, nonexistence  2, 5
   \(d_{\max}(n,k)\)  2
   upper and lower bounds  2, 5
   monotonicity  8
General code construction technique  5, 9
   input code(s), effect  5, 17
Improvement  10, 11, 12
   internal, external  13, 14
Justification of bound: internal, external, circular  22
P234# area  27
   anatomy  28, 29
   marking  29
Propagation rule or function  17, 24
   input, consequence, application  17
   complete decomposition  19
   composition, relation \(\rightarrow\)  19
   P234  26
Reference associated with bound  22, 24
   implicit, explicit, internal, external  23
   wiping  23, 28
   interpretation, tracing, cycle  23, 26, 33, 34
Report  23, 32, 35
Table of bounds on \(d_{\max}\)  2, 10, 34
   \(Lb\) and \(Ub\)  10, 11, 13, 35
   natural and artificial boundary  10, 24, 25, 30, 36
   bound kind: lower, upper  10, 11
   location triple, bound quadruple  10, 11, 35
   one-point (monotonic) modification  10, 12, 35
   closure, invariance, variance  13, 14, 18, 31
   initial table  6, 30
   violation  32
   format, block  32, 33, 35
Update  16, 19, 22, 24, 25, 27

41
### Upper and Lower Bounds on $d_{\text{ex}}(n,k)$ for Binary Linear Codes

<table>
<thead>
<tr>
<th>$n \times k$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>0,0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>17</td>
<td>11</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>12</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>12</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>13</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>21</td>
<td>14</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>22</td>
<td>14</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>23</td>
<td>15</td>
<td>12</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>24</td>
<td>16</td>
<td>13</td>
<td>12</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>8-9</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>16</td>
<td>14</td>
<td>12</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9-10</td>
<td>8-9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Appendix I. The Updated Table of Bounds*
| \( n,k \) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 30 |
|       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 30 |
| 2     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 3     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 4     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 5     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 6     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 7     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 8     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 9     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 10    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

Appendix I: The Updated Table of Bounds
| 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 0.1 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 0.1 |

Appendix I. The Updated Table of Bounds
<table>
<thead>
<tr>
<th>n-k</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>38</th>
<th>39</th>
<th>40</th>
<th>41</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>33</td>
<td>6</td>
<td>6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>34</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>35</td>
<td>6-8</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>36</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>37</td>
<td>7-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>38</td>
<td>8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>39</td>
<td>8-9</td>
<td>8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>8-10</td>
<td>8-9</td>
<td>8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>9-10</td>
<td>8-10</td>
<td>8-9</td>
<td>8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>10-11</td>
<td>8-10</td>
<td>8-9</td>
<td>8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>11-12</td>
<td>8-10</td>
<td>8-9</td>
<td>8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>11-12</td>
<td>10-12</td>
<td>9-11</td>
<td>8-10</td>
<td>8-10</td>
<td>8-10</td>
<td>8-9</td>
<td>8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>11-12</td>
<td>10-12</td>
<td>9-11</td>
<td>8-10</td>
<td>8-10</td>
<td>8-10</td>
<td>8-9</td>
<td>8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>11-12</td>
<td>10-12</td>
<td>9-11</td>
<td>8-10</td>
<td>8-10</td>
<td>8-10</td>
<td>8-9</td>
<td>8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>11-12</td>
<td>10-12</td>
<td>9-11</td>
<td>8-10</td>
<td>8-10</td>
<td>8-10</td>
<td>8-9</td>
<td>8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>12-14</td>
<td>12-15</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>12-14</td>
<td>12-15</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>12-14</td>
<td>12-15</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

Appendix 1: The Updated Table of Bounds
<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>38</th>
<th>39</th>
<th>40</th>
<th>41</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>22-28</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>22-28</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>22-28</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>22-28</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>22-28</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>22-28</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>22-28</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>22-28</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>22-28</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>22-28</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>22-28</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
</tbody>
</table>

Appendix 1: The Updated Table of Bounds
<table>
<thead>
<tr>
<th>n, k</th>
<th>41</th>
<th>42</th>
<th>43</th>
<th>44</th>
<th>45</th>
<th>46</th>
<th>47</th>
<th>48</th>
<th>49</th>
<th>50</th>
<th>51</th>
<th>52</th>
<th>53</th>
<th>54</th>
<th>55</th>
<th>56</th>
<th>57</th>
<th>58</th>
<th>59</th>
<th>60</th>
<th>1,2</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>33</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>35</td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>36</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>37</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>39</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>41</td>
</tr>
<tr>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>42</td>
</tr>
<tr>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>43</td>
</tr>
<tr>
<td>44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>45</td>
</tr>
<tr>
<td>46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>46</td>
</tr>
<tr>
<td>47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>47</td>
</tr>
<tr>
<td>48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>49</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
</tbody>
</table>

Appendix 1: The Updated Table of Bounds
| n, k, l | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 4,2 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

Appendix I: The Updated Table of Bounds
<table>
<thead>
<tr>
<th>n, k</th>
<th>61</th>
<th>62</th>
<th>63</th>
<th>64</th>
<th>65</th>
<th>66</th>
<th>67</th>
<th>68</th>
<th>69</th>
<th>70</th>
<th>71</th>
<th>72</th>
<th>73</th>
<th>74</th>
<th>75</th>
<th>76</th>
<th>77</th>
<th>78</th>
<th>79</th>
<th>80</th>
<th>2,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td></td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>4-6</td>
<td>4-6</td>
<td>4-6</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
</tr>
<tr>
<td>n, k</td>
<td>51</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
<td>2,3</td>
</tr>
<tr>
<td>n, k</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
<td>81</td>
</tr>
<tr>
<td>------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>76</td>
<td>6</td>
<td>5-6</td>
<td>4-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
</tr>
<tr>
<td>77</td>
<td>6</td>
<td>5-6</td>
<td>4-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>78</td>
<td>79</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>5-6</td>
<td>4-6</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>79</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>6-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>6-8</td>
<td>6-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>7-8</td>
<td>6-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6</td>
<td>6-8</td>
<td>6-5</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>81</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>7-8</td>
<td>6-8</td>
<td>6-8</td>
<td>6-8</td>
<td>6-7</td>
<td>6-8</td>
<td>6-5</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>82</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>7-8</td>
<td>6-8</td>
<td>6-8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-8</td>
<td>6-8</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>83</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>7-8</td>
<td>6-8</td>
<td>7-8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-8</td>
<td>6-8</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>84</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>7-8</td>
<td>6-8</td>
<td>7-8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-8</td>
<td>6-8</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>85</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>7-8</td>
<td>6-8</td>
<td>7-8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-8</td>
<td>6-8</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>86</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>7-8</td>
<td>6-8</td>
<td>7-8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-8</td>
<td>6-8</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>87</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>7-8</td>
<td>6-8</td>
<td>7-8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-8</td>
<td>6-8</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>7-8</td>
<td>6-8</td>
<td>7-8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-8</td>
<td>6-8</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>89</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>7-8</td>
<td>6-8</td>
<td>7-8</td>
<td>7-8</td>
<td>6-8</td>
<td>6-8</td>
<td>6-8</td>
<td>4-5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Appendix I: The Updated Table of Bounds
<table>
<thead>
<tr>
<th>α₁</th>
<th>61</th>
<th>62</th>
<th>63</th>
<th>64</th>
<th>65</th>
<th>66</th>
<th>67</th>
<th>68</th>
<th>69</th>
<th>70</th>
<th>71</th>
<th>72</th>
<th>73</th>
<th>74</th>
<th>75</th>
<th>76</th>
<th>77</th>
<th>78</th>
<th>79</th>
<th>80</th>
<th>4,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>12-18</td>
<td>12-18</td>
<td>12-18</td>
<td>12-17</td>
<td>12-16</td>
<td>11-16</td>
<td>10-15</td>
<td>10-14</td>
<td>10-13</td>
<td>9-12</td>
<td>8-11</td>
<td>8-10</td>
<td>8-9</td>
<td>7-8</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>12-19</td>
<td>12-18</td>
<td>12-18</td>
<td>12-17</td>
<td>12-16</td>
<td>11-16</td>
<td>10-15</td>
<td>10-14</td>
<td>10-13</td>
<td>9-12</td>
<td>8-11</td>
<td>8-10</td>
<td>8-9</td>
<td>7-8</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>12-20</td>
<td>12-19</td>
<td>12-18</td>
<td>12-17</td>
<td>12-16</td>
<td>11-16</td>
<td>10-15</td>
<td>10-14</td>
<td>10-13</td>
<td>9-12</td>
<td>8-11</td>
<td>8-10</td>
<td>8-9</td>
<td>7-8</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-19</td>
<td>12-18</td>
<td>12-17</td>
<td>12-16</td>
<td>11-16</td>
<td>10-15</td>
<td>10-14</td>
<td>10-13</td>
<td>9-12</td>
<td>8-11</td>
<td>8-10</td>
<td>8-9</td>
<td>7-8</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-19</td>
<td>12-18</td>
<td>12-17</td>
<td>12-16</td>
<td>11-16</td>
<td>10-15</td>
<td>10-14</td>
<td>10-13</td>
<td>9-12</td>
<td>8-11</td>
<td>8-10</td>
<td>8-9</td>
<td>7-8</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-19</td>
<td>12-18</td>
<td>12-17</td>
<td>12-16</td>
<td>11-16</td>
<td>10-15</td>
<td>10-14</td>
<td>10-13</td>
<td>9-12</td>
<td>8-11</td>
<td>8-10</td>
<td>8-9</td>
<td>7-8</td>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-19</td>
<td>12-18</td>
<td>12-17</td>
<td>12-16</td>
<td>11-16</td>
<td>10-15</td>
<td>10-14</td>
<td>10-13</td>
<td>9-12</td>
<td>8-11</td>
<td>8-10</td>
<td>8-9</td>
<td>7-8</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-19</td>
<td>12-18</td>
<td>12-17</td>
<td>12-16</td>
<td>11-16</td>
<td>10-15</td>
<td>10-14</td>
<td>10-13</td>
<td>9-12</td>
<td>8-11</td>
<td>8-10</td>
<td>8-9</td>
<td>7-8</td>
<td>101</td>
</tr>
<tr>
<td>119</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-19</td>
<td>12-18</td>
<td>12-17</td>
<td>12-16</td>
<td>11-16</td>
<td>10-15</td>
<td>10-14</td>
<td>10-13</td>
<td>9-12</td>
<td>8-11</td>
<td>8-10</td>
<td>8-9</td>
<td>7-8</td>
</tr>
<tr>
<td>120</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-20</td>
<td>12-19</td>
<td>12-18</td>
<td>12-17</td>
<td>12-16</td>
<td>11-16</td>
<td>10-15</td>
<td>10-14</td>
<td>10-13</td>
<td>9-12</td>
<td>8-11</td>
<td>8-10</td>
<td>8-9</td>
</tr>
</tbody>
</table>

**Appendix I. The Updated Table of Bounds**
| n x k | 81  | 82  | 83  | 84  | 85  | 86  | 87  | 88  | 89  | 90  | 91  | 92  | 93  | 94  | 95  | 96  | 97  | 98  | 99  | 100 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|       | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 |
|       | 6-8 | 7-8 | 8-9 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 | 8-10 |
|       | 5-8 | 5-8 | 7-8 | 7-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 | 8-8 |
|       | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 | 6-7 |
|       | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|       | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|       | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 | 4-5 |
|       | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|       | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|Appendix I: The Updated Table of Bounds |
\begin{table}
\centering
\begin{tabular}{|c|cccccccccccccccccccccc|}
\hline
\textbf{\textit{n+k}} & \textbf{101} & \textbf{102} & \textbf{103} & \textbf{104} & \textbf{105} & \textbf{106} & \textbf{107} & \textbf{108} & \textbf{109} & \textbf{110} & \textbf{111} & \textbf{112} & \textbf{113} & \textbf{114} & \textbf{115} & \textbf{116} & \textbf{117} & \textbf{118} & \textbf{119} & \textbf{120} & \textbf{4.5} \\
\hline
\hline
\end{tabular}
\end{table}
| \( n, k \) | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 |
|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 101  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 101 |
| 102  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 102 |
| 103  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 103 |
| 104  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 104 |
| 105  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 105 |
| 106  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 106 |
| 107  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 107 |
| 108  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 108 |
| 109  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 109 |
| 110  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 110 |
| 111  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 111 |
| 112  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 112 |
| 113  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 113 |
| 114  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 114 |
| 115  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 115 |
| 116  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 116 |
| 117  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 117 |
| 118  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 118 |
| 119  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 119 |
| 120  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 120 |

Appendix 1. The Updated Table of Bounds
Table of bounds read from file: (WSADD101)~ELGERSTINAFF/BOUNDS ON USER2
5990 unsolved cases

All blocks printed.

Considering bound (lower, 45, 6,22) "SS"
\[ d_{\text{max}}(45, 6) = 22 "SS" \]
\[ d_{\text{max}}(44, 6) = 21 "SS" \]
\[ d_{\text{max}}(77, 6) = 28 "SS" \]
\[ d_{\text{max}}(76, 6) = 32 "SS" \]
\[ d_{\text{max}}(90, 7) = 44 "SS" \]
(PT = 0.10)

Considering bound (lower, 75, 6,36) "SS"
\[ d_{\text{max}}(75, 6) = 36 "SS" \]
\[ d_{\text{max}}(72, 6) = 35 "SS" \]
(PT = 0.02)

Considering bound (lower, 77, 6,36) "SS": USELESS (PT = 0.00)

Considering bound (lower, 105, 7,32) "SS"
\[ d_{\text{max}}(105, 7) = 52 "SS" \]
\[ d_{\text{max}}(104, 7) = 51 "SS" \]
(PT = 0.04)

Considering bound (lower, 109, 7,54) "SS"
\[ d_{\text{max}}(109, 7) = 54 "SS" \]
\[ d_{\text{max}}(108, 7) = 53 "SS" \]

Considering bound (upper, 28, 6,12) "GW"
\[ d_{\text{max}}(28, 6) = 12 "GW" \]
\[ d_{\text{max}}(29, 6) = 13 "GW" \]
\[ d_{\text{max}}(53, 7) = 24 "GW" \]
\[ d_{\text{max}}(102, 8) = 48 "GW" \]
\[ d_{\text{max}}(57, 10) = 49 "GW" \]
\[ d_{\text{max}}(36, 11) = 49 "GW" \]
\[ d_{\text{max}}(65, 15) = 48 "GW" \]
\[ d_{\text{max}}(64, 16) = 48 "GW" \]
\[ d_{\text{max}}(32, 9) = 12 "GW" \]
\[ d_{\text{max}}(33, 10) = 12 "GW" \]

Bound (upper, 34,11,12) "K": external reference removed
(PT = 0.52)

Considering bound (upper, 40, 6,18) "GW"
\[ d_{\text{max}}(40, 6) = 18 "GW" \]
\[ d_{\text{max}}(41, 6) = 19 "GW" \]

* DIMENSION 6 HAS BEEN COMPLETELY DETERMINED FOR LENGTH <= 127

Considering bound (lower, 26, 9, 9) "Ha"
\[ d_{\text{max}}(27, 9) = 10 "Ha" \]
\[ d_{\text{max}}(26, 8) = 10 "Ha" \]
(PT = 0.06)

Considering bound (lower, 30, 12, 9) "Ha"
\[ d_{\text{max}}(31, 12) = 10 "Ha" \]
(PT = 0.06)

Considering bound (lower, 34, 15, 9) "Ha"
Bound (lower, 35, 15, 10) "K": external reference removed
(PT = 0.07)

Considering bound (lower, 47, 25, 9) "Ha": useful (PT = 0.04)

Considering bound (lower, 72, 53, 7) "Ha"

Bound (lower, 70,51,7) "O": external reference removed
\[ d_{\text{max}}(73, 55) = 8 "O" \]
\[ d_{\text{max}}(72, 52) = 8 "O" \]
(PT = 0.03)

Considering bound (lower, 86, 66, 7) "Ha"
Bound (lower, 83,63,7) "Q": external reference removed
\[ d_{\text{max}}(87, 66) = 8 "Q" \]
\[ d_{\text{max}}(86, 65) = 8 "Q" \]
\[ d_{\text{max}}(85, 64) = 8 "Q" \]
(PT = 0.03)

Considering bound (lower, 92, 7,45) "Br" *
\[ d_{\text{max}}(92, 7) = 45 "Br" \]
\[ d_{\text{max}}(93, 7) = 46 "Br" \]
(PT = 0.03)

Considering bound (upper, 61, 7,29) "Lc" *
\[ d_{\text{max}}(61, 7) = 29 "Lc" \]
\[ d_{\text{max}}(60, 7) = 28 "Lc" \]
(PT = 0.05)

Considering bound (upper, 29, 6,13) "Lc": USELESS (PT = 0.00)

Considering bound (upper, 103, 8,49) "Lc": USELESS (PT = 0.00)

Considering bound (upper, 106, 8,53) "Lc": USELESS (PT = 0.00)

Considering bound (upper, 110, 8,51) "Lc": USELESS (PT = 0.00)

Considering bound (upper, 113, 8,53) "Lc": USELESS (PT = 0.00)

Considering bound (upper, 118, 8,57) "Lc": USELESS (PT = 0.00)

Considering bound (upper, 121, 8,59) "Lc": USELESS (PT = 0.00)

Considering bound (upper, 125, 8,61) "Lc"
\[ d_{\text{max}}(125, 9) = 61 "Lc" \]
\[ d_{\text{max}}(125, 4) = 60 "Lc" \]
(PT = 0.02)

Considering bound (lower, 80, 47,11) "Su"
Bound (lower, 79,46,11) "K": external reference removed
(PT = 0.05)

Considering bound (lower, 56, 46,13) "Su": useful (PT = 0.22)

Considering bound (lower, 76, 50, 9) "Ka": useful (PT = 0.12)

Considering bound (lower, 76, 44,11) "Ka"
Bound (lower, 74,42,11) "K": external reference removed
(PT = 0.03)

Considering bound (lower, 73, 38,13) "Ka": useful (PT = 0.03)

Considering bound (lower, 72, 17,25) "Ka"
Bound (lower, 73,17,25) "K": improved to 26
(PT = 0.02)

Considering bound (lower, 73, 9,30) "Ka": useful (PT = 0.02)

Considering bound (lower, 76, 17,27) "Ka": useful (PT = 0.03)

Considering bound (lower, 80, 15,29) "Ka"
Bound (lower, 78,15,29) "K": external reference removed
(PT = 0.03)

Considering bound (lower, 76, 9,33) "Ka": useful (PT = 0.07)

Considering bound (lower, 45, 6,22) "AL": USELESS (PT = 0.00)

Considering bound (lower, 103, 7,52) "AL": USELESS (PT = 0.00)

Considering bound (lower, 109, 7,54) "AL": USELESS (PT = 0.00)

Considering bound (lower, 93, 7,46) "AL": USELESS (PT = 0.00)

Considering bound (lower, 87, 7,42) "AL"
\[ d_{\text{max}}(87, 7) = 42 "AL" \]
(PT = 0.03)
Considering bound (lower, 24, 7, 10) "T3"
Bound (lower, 24, 7) = 10 "T3" ± 8
d_max = 24, 7 = 9 ± 8  
Length 23 has been completely determined

Considering bound (lower, 32, 7, 14) "T3"
Bound (lower, 32, 7) = 14 "T3" ± 8  
PT = 0.03
Considering bound (upper, 31, 7, 13) "T3"
Bound (upper, 31, 7) = 13 "T3" ± 8  
PT = 0.03
Considering bound (upper, 34, 7, 15) "T3"
Bound (upper, 34, 7) = 15 "T3" ± 8  
PT = 0.49
Considering bound (lower, 39, 7, 17) "T3"
Bound (lower, 39, 7) = 17 "T3" ± 8  
PT = 0.03
Considering bound (lower, 47, 7, 22) "T3"
Bound (lower, 47, 7) = 22 "T3" ± 8  
PT = 0.04
Considering bound (upper, 55, 7, 25) "T3"
Bound (upper, 55, 7) = 25 "T3" ± 8  
PT = 0.26
Considering bound (upper, 58, 7, 27) "T3"
Bound (upper, 58, 7) = 27 "T3" ± 8  
PT = 0.17
Considering bound (lower, 51, 9, 23) "W1" useful (PT = 0.04)
Considering bound (lower, 45, 10, 17) "Pu" useful (PT = 0.07)
Considering bound (lower, 51, 9, 23) "Pu" USELESS (PT = 0.00)
Considering bound (lower, 52, 10, 21) "Pu"
Bound (lower, 51, 9, 23) "W1" useful (PT = 0.04)
Considering bound (lower, 80, 14, 32) "Pu"
Bound (lower, 80, 12, 32) "X" useful (PT = 0.10)
Considering bound (lower, 79, 9, 35) "Pu" useful (PT = 0.05)
Considering bound (lower, 94, 9, 41) "Pu" useful (PT = 0.02)
Considering bound (lower, 97, 9, 44) "Pu" useful (PT = 0.02)
Considering bound (lower, 120, 9, 56) "Pu" USELESS (PT = 0.00)
Considering bound (lower, 99, 65, 11) "Ro"
Bound (lower, 95, 61, 11) "X" useful (PT = 0.02)
Considering bound (lower, 101, 60, 13) "Ro"
Bound (lower, 96, 55, 13) "X" useful (PT = 0.02)
All (13) blocks with unprinted modifications have been printed.

Cumulative session statistics:
18.68 process time
112 updates
89 useful updates
654 lower bounds improved
370 upper bounds improved
22 external references removed
9 external references superseded
110 cases solved
5480 cases left open
Table of bounds saved on file: (WS00010E1)NEWBOUNDS ON USER2
34 pages saved
Bound updater terminated.

Appendix III: Report for the updated Table
BOUNDUPDATER (11/06/84)

**RESET LIST**

**SET LISTINCL**

**SET WARN5PR**

**SET NOBORDERS**

**RESET TESTOUTPUT**

**RESET LSIPRINTER**

**PROGRAM BoundUpdater (**

**input,**

**output,**

**Pr**

**Report :**

**List :**

**);**

******************************************************************************

**This program maintains a table of known upper and lower bounds on the maximum minu-**

**tum-distance of binary linear codes with word length less than 128.**

**Author: Tom Verhoeff (student)**

**Date: January 1984 (revised September 1984)**

******************************************************************************

**CONST**

**Version**

**MaxBtIndex**

**BtSize**

**NofRef**

**A_Ref**

**B_Ref**

**C_Ref**

**D_Ref**

**E_Ref**

**MaxPage**

**TYPE**

**BoundKind**

**BtIndex**

**Reference**

**BtElementKind**

**BtElement**

**Statistics**

**BoundInfo**

**dist**

**refch1, refch2 :** char ; (not ref: Reference because of packing)

******************************************************************************

**10000** **00010000**

**10010** **00010010**

**10020** **00010020**

**10030** **00010030**

**10040** **00010040**

**10050** **00010050**

**10060** **00010060**

**10080** **00010080**

**11000** **00011000**

**11010** **00011010**

**11020** **00011020**

**11030** **00011030**

**11040** **00011040**

**11050** **00011050**

**11060** **00011060**

**11070** **00011070**

**11080** **00011080**

**11090** **00011090**

**11100** **00011100**

**11110** **00011110**

**11120** **00011120**

**11130** **00011130**

**11140** **00011140**

**11150** **00011150**

**11160** **00011160**

**11170** **00011170**

**11180** **00011180**

**11190** **00011190**

**11200** **00011200**

**11210** **00011210**

**11220** **00011220**

**12000** **00012000**

**12010** **00012010**

**12020** **00012020**

**12030** **00012030**

**12040** **00012040**

**12050** **00012050**

**12060** **00012060**

**12070** **00012070**

**12080** **00012080**

**12090** **00012090**

**12090** **00012100**

**12100** **00012110**

**12110** **00012120**

**12120** **00012130**

**12130** **00012140**

**12140** **00012150**

**12150** **00012160**

**12160** **00012170**

**12170** **00012180**

**12180** **00012190**

**12190** **00012200**

**12200** **00012210**

**12210** **00012220**

**12220** **00012230**

**12230** **00012240**

**12240** **00012250**

**Appendix IV. Program Listings**
FUNCTION IsOpenCase(n,k: BtIndex): boolean;
BEGIN 
  IF 0 < k <= n THEN 
    Bt(n,k).dist = current lower bound, 
    Bt(n,k).refch1..2 corresponding reference, 
    Bt(k,n).dist = current upper bound, 
    Bt(k,n).refch1..2 corresponding reference, 
  ELSE Bt(n,n).dist = 1 = lower = upper
END { IsOpenCase } ;

FUNCTION HasExternalRefCb: BoundKind; n,k: BtIndex): boolean;
BEGIN 
  IF 1 <= k <= n THEN 
    C_SOURCE: WITH Bt[n,k] DO
      HasExternalRef := (refch1 <> "") OR (refch2 <> "") ;
  ELSE Bt[n,n].unsolved = # k: case (n,k) is open,
  ELSE Bt[n,0].unsolved = # n: case (n,k) is open
END { HasExternalRef } ;

FUNCTION HasExplicitRefCb: BoundKind; n,k: BtIndex): boolean;
BEGIN 
  IF 1 <= k <= n THEN 
    CASE n OF 
      lower: WITH Bt[n,k] DO
        HasExplicitRef := (refch1 <> "") OR (refch2 <> "") ;
      upper: WITH Bt[k,n] DO
        HasExplicitRef := (refch1 <> "") OR (refch2 <> "") ;
    END { case } ;
  ELSE Bt[0,0].unsolved = # n,k: case (n,k) is open
END { HasExplicitRef } ;

mark unchanged : boolean ;
END ( BtElement ) ;

BtArray = ARRAY [ BtIndex, BtIndex ] OF BtElement ;
Btfile = FILE OF BtArray ;

VAR 
  Bt : BtArray ;
  Id : ARRAY [ BoundKind ] OF 
    PACKED ARRAY [ 1 .. 5 ] OF char ;
  Pr, Report, List : TEXT ;
  SessionStatistics: ( session statistics )
  BtModified : boolean ; ( Bt modified after last save )
  TotUpdates, 
  TotUseful, 
  TotExternal狸, 
  TotExternalImproved, 
  Bt00 
  TotImproved : ARRAY [ BoundKind ] OF integer ;
  PrevPt : real 
  PagelPrinted : integer ;
  TotalPrinted : ARRAY [0..4,0..6] OF boolean ;
  ( identifies modified unprinted blocks )
END ;
00015210 00015220 00015230 00015240 00015250 00015260 00015270 00015280 00015290 00015300 00015310 00015320 00015330 00015340 00015350 00015360 00015370 00015380 00015390 00015400 00015410 00015420 00015430 00015440 00015450 00015460 00015470 00015480 00015490 00015500 00015510 00015520 00015530 00015540 00015550 00015560 00015570 00015580 00015590 00015600 00015610 00015620 00015630 00015640 00015650 00015660 00015670 00015680 00015690 00015700 00015710 00015720 00015730 00015740 00015750 00015760 00015770 00015780 00015790 00015800 00015810 00015820 00015830 00015840 00015850 00015860 00015870 00015880 00015890 00015900 00015910 00015920 00015930 00015940 00015950 00015960 00015970 00015980 00015990 00016000 00016010 00016020 00016030 00016040 00016050 00016060 00016070 00016080 00016090 00016100 00016110 00016120 00016130 00016140 00016150 00016160 00016170 00016180 00016190 00016200 00016210 00016220 00016230 00016240 00016250 00016260 00016270 00016280 00016290 00016300 00016310 00016320 00016330 00016340 00016350 00016360 00016370 00016380 00016390 00016400 00016410 00016420 00016430 00016440 00016450 00016460 00016470 00016480 00016490 00016500 00016510 00016520 00016530 00016540 00016550 00016560 00016570 00016580 00016590 00016600 00016610 00016620 00016630 00016640 00016650 00016660 00016670 00016680 00016690 00016700 00016710 00016720 00016730 00016740 00016750 00016760 00016770 00016780 00016790 00016800 00016810 00016820 00016830 00016840 00016850 00016860 00016870 00016880 00016890 00016900 00016910 00016920 00016930 00016940 00016950 00016960 00016970 00016980 00016990 00017000 00017010 00017020 00017030 00017040 00017050 00017060 00017070 00017080 00017090 00017100 00017110 00017120 00017130 00017140 00017150 00017160 00017170 00017180 00017190 00017200 00017210

** Appendix IV. Program Listings **
PROCEDURE ProcessInput;
VAR ch, dummy: char;
b: BoundKind;
n, k, d, ior: integer;
r: Reference;
BEGIN
REPEAT until (ch = 'q')
  SayProcThe;
  REPEAT until (ior = IORES(OK))
    Write(Write("Give command: L, U, S, P, M, A, W, Q: ", ch): ToUpper(ch));
    IF eoln THEN readln;
    REPEAT read(ch) UNTIL (ch <> 'l')
      b := Read(n, k, d);
      IF (ior = IORES(OK)) THEN BEGIN
        REPEAT read(dummy) UNTIL (dummy <> 'r');
        r(1) := dummy;
        IF eoln THEN dummy := 'r' ELSE read(dummy);
        r(2) := dummy;
        Write("Bound = ('", Id(b), ",", n, ",", k, ",", d, ",") ", r,");
      END;
      ior := IORES(IORESDATAERR);
    END;
    Write("Invalid bound format");
  END;
  Write("Invalid command: ", ch): ior := IORES(IORESDATAERR);
  Write("Unrecognized command: ", ch);
  ior := IORES(IORESDATAERR);
  Write("Invalid block coordinate format");
  Write("Invalid block format");
  Write("QUIT command");
END (case ch);
UNTIL (ch = 'q');
END (ProcessInput);
END (SaveBit);
 rewite(Pro) ; rewite(Report) ; rewite(List) ;
 rewrite("Bound Updater [",Version=", MaxBtIndex = ",MaxBtIndex:1]");
 write(Report);
 write("Bound Updater [",Version=", MaxBtIndex = ",MaxBtIndex:1]");
 ID(lower) := 'lower'; ID(upper) := 'upper'; ( constant array )
 PrevPt := 0.0 ; PagesPrinted := 0 ;
 BTModified := false ; ( no modifications yet )
 TotUpdates := 0 ; TotUseful := 0 ;
 TotImproved(lower) := 0 ; TotImproved(upper) := 0 ;
 TotImproved := 0 ; TotImproved := 0 ;
 FOR i := 0 TO 6 DO ToBePrinted[i,j] := false ;
 reset(cbf); ut := cbfa ;
 GETATTR(BufEl[TITLE,t]) ;
 writeln(Report,"Table of bounds read from file: ",t) ;
 FOR j := 1 TO MaxBtIndex DO
gt[j].unsolved := true ;
 BtDO := Bt[0,0].unsolved ; ( initial number of open cases )
 writeln(Report,BtDO:8,' unsolved cases'); writeln(Report);
 END ( Initialize ) ;
 PROCEDURE Finalize ;
 BEGIN
eSaveT ;
 writeln(Report) ;
 writeln(Report,PagesPrinted:8,' pages printed');
 writeln(Report,"Bound Updater Terminated.");
 END ( Finalize ) ;
 BEGIN ( BoundUpdater )
 Initialize ;
 ProcessInput ;
 Finalize ;
 END ( BoundUpdater ).
80000 PROCEDURE UpdateBt(b: BoundKind;
80010 len, dim, newdist: integer;
80020 r: Reference);
80030 LABEL 0; ( in case of bound violation )
80040 VAR
80050 NoRemarksYet,
80060 Modified: boolean; ( table modified since last entry of UpdateBt )
80070 pt: real;
80080 depth: integer;
80090
t 01000 $ SET ONLY = NOT TEST OUTPUT
80110 01120 PROCEDURE indent; v: integer;
80130 BEGIN
80140 FOR i := 1 TO Min(25, depth) DO write(Report, " ");
80150 IF (depth > 25) THEN write(Report, " > ");
80160 END ( indent ) ;
80170 01800 $ POP OMIT
80190 02000 FUNCTION IsValid(b: BoundKind;
80210 n,k,d: integer): boolean;
80220 BEGIN
80230 IF IsInTable(n,k) THEN CASE b OF
80240 lower: IsValid := (Bt(n,k).dist = d);
80250 upper: IsValid := (Bt(k,n).dist > d);
80260 END ( case );
80270 ELSE IsValid := false;
80280 END ( IsValid ) ;
80290 03000 FUNCTION IsMarked(b: BoundKind; n,k: integer): boolean;
80310 BEGIN
80320 IF IsInTable(n,k) THEN CASE b OF
80330 lower: IsMarked := (Bt(n,k).mark = depth);
80340 upper: IsMarked := (Bt(k,n).mark = depth);
80350 END ( case );
80360 ELSE IsMarked := false;
80370 END ( IsMarked ) ;
80380 04400 FUNCTION IsNew(b: BoundKind; n,k, newd: integer): boolean;
80450 BEGIN
80460 IF IsInTable(n,k) THEN CASE b OF
80470 lower: IsNew := (Bt(n,k).dist < newd);
80480 upper: IsNew := (Bt(k,n).dist > newd);
80490 END ( case );
80500 ELSE IsNew := false;
80510 END ( IsNew ) ;
80520 05550 PROCEDURE EnsureNewLine ;
BEGIN
IF NoRemarksYet THEN BEGIN
  writeln(Report); NoRemarksYet := false END;
END (EnsureNewLine);

PROCEDURE ToReport(b: BoundKind; n,k: BkIndex);
BEGIN
  EnsureNewLine;
  write('Bound ('',id[b],'',',',n:1,''',',',k:1,''');
CASE b OF
  lower: WITH BtCn,k do
    write('dist:','); refch1,refch2,'',''
  upper: WITH BtCk,n do
    write('dist:','); refch1,refch2,'',''
END ( case );
END ( ToReport );

PROCEDURE CondWipeRef(b: BoundKind; n,k,d: integer);
BEGIN
  IF IsSame(b,n,k,d) THEN BEGIN
    lModified := true;
    SET OMIT = TESTOUTPUT
    IF HasExternalRef(b,n) THEN
      BEGIN
        TotEltWipped := TotEltWipped + 1;
        ToReport(b,n,k);
      END ( IsSame )
  END ( CondWipeRef );

PROCEDURE VioLa ti onExit(b: BoundKind; n,k,d: integer; r: Reference);
BEGIN
  EnsureNewLine;
  write('Bound updater terminated, table not saved.');
  IF LModified THEN BEGIN
    writeln('Internal table assumed corrupted.');
    writeln('Bound updater terminated, table not saved.');
    write('Aborted w/o saving.');
    CLOSE(Pr); CLOSE(Report); CLOSE(output);
  END (ViolationExit );
END
PROCEDURE MakeImprovement(b: BoundKind; n,k,newd: integer; r: Reference) ;

BEGIN ( IsImprovement(b,n,k,newd) assumed )
IF (newd < Bt[n,k].dist) OR (newd > Bt[k,n].dist) THEN
ViolatationE(b,n,k,newd) ;
Modified := true ;
SET OMIT = NOT TESTOUTPUT ;
SET OMIT = TESTOUTPUT ;
END ;
CASE b OF
lower: WITH Bt[n,k] DO BEGIN
dist := newd ;
SetE := NOT SetE ;
END ;
upper: WITH St[k,n] DO BEGIN
dist := newd ;
SetE := NOT SetE ;
END ;
END ;
ToBePrinted := (n,k) DIV (5) ;
AssignRefl(b,n,k,r) ;
END ;
END ;
PROCEDURE Update(b: BoundKind; n,k,newd: integer; r: Reference) ;
FORWARD ;
PROCEDURE #3_E(n,k,newd: integer) ;
(zero or more P3, followed by E )
VAR nn,dd,kk: integer ;
BEGIN
IF od(n,k,newd) THEN BEGIN
  n := n - 1 ;
 :newd := :newd - 1 ;
END ;
BEGIN
  END ;
WHILE (nn <= MaxBIndex+5) AND IsMarkedupper(nn,kk) DO BEGIN
Updateupper(nn,dd,E_Ref) ;
END ;
PROCEDURE zm3(n,k,newd: integer);
VAR s: integer;
BEGIN
  IF odd(newd) THEN BEGIN n := n + 1; newd := newd + 1 END
  WHILE IsMarked(upper,n,k) DO BEGIN
    s := s + 1;
    IF IsInTable(n,k) THEN s := s + 1;
    IF (n < MaxIndex) THEN s := s + 1;
  END;
END;

PROCEDURE zm4(n,k,newd: integer);
VAR m, j: integer;
BEGIN
  IF odd(newd) THEN BEGIN n := n + 1; newd := newd + 1 END;
  WHILE IsMarked(upper,n,k) DO BEGIN
    m := m + 1; j := j + 1;
    m := m + 1; j := j + 1;
    IF (n < MaxIndex) THEN s := s + 1;
  END;
END;

PROCEDURE zm5(n,k,newd: integer);
VAR s: integer;
BEGIN
  IF odd(newd) THEN BEGIN n := n + 1; newd := newd + 1 END;
  WHILE IsMarked(upper,n,k) DO BEGIN
    IF odd(newd) THEN BEGIN n := n + 1; newd := newd + 1 END;
    m := m + 1; j := j + 1;
    IF (n < MaxIndex) THEN s := s + 1;
  END;
END;

PROCEDURE zm6(n,k,newd: integer);
VAR m: integer;
BEGIN
  IF odd(newd) THEN BEGIN n := n + 1; newd := newd + 1 END;
  WHILE IsMarked(upper,n,k) DO BEGIN
    IF IsInTable(n,k) THEN s := s + 1;
    ELSE s := s + 1;
    update(lower,n-s,k-s+1,newd,B_Refer); ( B3 )
    n := n + 1;
  END;
END;

PROCEDURE zm7(n,k,newd: integer);
VAR s: integer;
BEGIN
  IF odd(newd) THEN BEGIN n := n + 1; newd := newd + 1 END;
  WHILE IsMarked(upper,n,k) DO BEGIN
    IF IsInTable(n,k) THEN s := s + 1;
    ELSE s := s + 1;
    update(lower,n-s,k-s+1,newd,B_Refer); ( B3 )
    n := n + 1;
  END;
END;

PROCEDURE zm8(n,k,newd: integer);
VAR s: integer;
BEGIN
  IF odd(newd) THEN BEGIN n := n + 1; newd := newd + 1 END;
  WHILE IsMarked(upper,n,k) DO BEGIN
    IF IsInTable(n,k) THEN s := s + 1;
    ELSE s := s + 1;
    update(lower,n-s,k-s+1,newd,B_Refer); ( B3 )
    n := n + 1;
  END;
END;

PROCEDURE zm9(n,k,newd: integer);
VAR s: integer;
BEGIN
  IF odd(newd) THEN BEGIN n := n + 1; newd := newd + 1 END;
  WHILE IsMarked(upper,n,k) DO BEGIN
    IF IsInTable(n,k) THEN s := s + 1;
    ELSE s := s + 1;
    update(lower,n-s,k-s+1,newd,B_Refer); ( B3 )
    n := n + 1;
  END;
END;
PROCEDURE main(lower,n,k,newd: integer) {
    IF (n+1 <= MaxBtIndex+5) THEN
        UPDATE(lower,n,0,n+1,Dist,Ref); ( Ce )
    n := n + 1;
    UNTIL (n+1 > MaxBtIndex+5) ;
    n := n - 1; k := k - 1;
    END ( while ) ;
    END ( zm3_zm4 ) ;
}

PROCEDURE zm4_D(n,k,newd: integer) ;
    VAR j: integer;
    BEGIN
        WHILE IsMarked(lower,n,k) DO BEGIN
            j := 1;
            REPEAT
                Update(lower,2*n,k+j,IN(2*newd,Bt[n,j].Dist),D_Ref)
                j := j + 1;
            UNTIL (Bt[n,j].Dist <= newd) ;
            REPEAT
                Update(lower,2*n,k+j,IN(newd,2*Bt[n,j].Dist),D_Ref)
                j := j + 1;
            UNTIL (j = n) ;
            n := n - 1; k := k - 1;
        END ( while ) ;
    END ( zm4_D ) ;

PROCEDURE zm5_zm6_D(n,k,newd: integer) ;
    ( zero or more P4, followed by D )
    BEGIN
        WHILE IsMarked(lower,n,k) DO BEGIN
            j := 1;
            REPEAT
                Update(lower,2*n,k+j,MIN(2*newd,Bt[n,j].Dist),D_Ref); ( D1j )
                j := j + 1;
            UNTIL (Bt[n,j].Dist <= newd) ;
            REPEAT
                Update(lower,2*n,k+j,MIN(newd,2*Bt[n,j].Dist),D_Ref); ( Dj )
                j := j + 1;
            UNTIL (j = n) ;
            n := n - 1; newd := newd - 1;
            END ( while ) ;
        END ( zm5_zm6 ) ;

PROCEDURE zm2_zm3_D(n,k,newd: integer) ;
    ( once or more P4, zero or more P4, followed by D )
    BEGIN
        n := n + 1;
        IF odd(newd) THEN newd := newd + 1;
        WHILE IsMarked(lower,n,k) AND (2*n <= MaxBtIndex+5) DO BEGIN
            zm4_D(n,k,newd);
            n := n - 1; newd := newd - 1;
        END ( while ) ;
    END ( zm2_zm3 ) ;

PROCEDURE zm4_Lower(n,k,newd: integer; r: Reference) ;
    ( zero or more P4 (lower bound) )
    BEGIN
        IF NOT IsInTable(n,k) THEN BEGIN
            k := k+MaxBtIndex; n := MaxBtIndex + 5;
            WHILE IsImprovement(lower,n,k,newd) DO BEGIN
                MakeImprovement(lower,n,k,newd,r);
                Bt[n,k].Mark := depth;
                r := NoRef;
                n := n - 1; k := k - 1;
            END ( while ) ;
        END ( if )
        IF (n < MaxBtIndex) THEN BEGIN
            I: 00022330
        22340 00022340
        22350 00022350
        22360 00022360
        22370 00022370
        22380 00022380
        22390 00022390
        22400 00022400
        22410 00022410
        22420 00022420
        22430 00022430
        22440 00022440
        22450 00022450
        22460 00022460
        22470 00022470
        22480 00022480
        22490 00022490
        22500 00022500
        22510 00022510
        22520 00022520
        22530 00022530
        22540 00022540
        22550 00022550
        22560 00022560
        22570 00022570
        22580 00022580
        22590 00022590
        22600 00022600
        22610 00022610
        22620 00022620
        22630 00022630
        22640 00022640
        22650 00022650
        22660 00022660
        22670 00022670
        22680 00022680
        22690 00022690
        22700 00022700
        22710 00022710
        22720 00022720
        22730 00022730
        22740 00022740
        22750 00022750
        22760 00022760
        22770 00022770
        22780 00022780
        22790 00022790
        22800 00022800
        22810 00022810
        22820 00022820
        22830 00022830
        22840 00022840
        22850 00022850
        22860 00022860
        22870 00022870
        22880 00022880
        22890 00022890
        22900 00022900
        22910 00022910
        Appendix IV: Program Listings
PROCEDURE 0.2_zM4_lower(n,k,newd: integer) ;
BEGIN
   IF (n <= MaxBtIndex) THEN BEGIN
      newd := newd - (n <= MaxBtIndex) * k;
      n := k - 1 + MaxBtIndex;
   END;
   WHILE (n > MaxBtIndex) OR IsImprovement(lower,n,k,newd) DO BEGIN
      z4_lower(n,k,newd) := n + 1;
      IF IsInTable(n,k) THEN r := NoRef;
      n := n - 1;
      newd := newd - 1;
   END;
END{if};
PROCEDURE 0.2_zM4_upper(n,k,newd: integer) ;
BEGIN
   IF (n <= MaxBtIndex) THEN BEGIN
      newd := newd - (n <= MaxBtIndex) * k;
      n := k - 1 + MaxBtIndex;
   END;
   WHILE (n > MaxBtIndex) OR IsImprovement(upper,n,k,newd) DO BEGIN
      z4_upper(n,k,newd) := n + 1;
      IF IsInTable(n,k) THEN r := NoRef;
      n := n - 1;
      newd := newd - 1;
   END;
END{if};
PROCEDURE 0.2_zM4_upper(n,k,newd: integer) ;
BEGIN
   IF (n <= MaxBtIndex) THEN BEGIN
      newd := newd - (n <= MaxBtIndex) * k;
      n := k - 1 + MaxBtIndex;
   END;
   WHILE (n > MaxBtIndex) OR IsImprovement(lower,n,k,newd) DO BEGIN
      z4_lower(n,k,newd) := n + 1;
      IF IsInTable(n,k) THEN r := NoRef;
      n := n - 1;
      newd := newd - 1;
   END;
END{if};
PROCEDURE zM3_zM4_upper(n,k,newd: integer) ;
BEGIN
   IF (n <= MaxBtIndex) THEN BEGIN
      newd := newd - (n <= MaxBtIndex) * k;
      n := k - 1 + MaxBtIndex;
   END;
   WHILE (n > MaxBtIndex) OR IsImprovement(upper,n,k,newd) DO BEGIN
      z4_upper(n,k,newd) := n + 1;
      IF IsInTable(n,k) THEN r := NoRef;
      n := n - 1;
      newd := newd - 1;
   END;
END{if};
PROCEDURE zm3_zm4_upper(n,k,newd: integer; r: Reference); 
23520 \ 
23530 BEGIN \ 
23540 WHILE IsImprovement (upper, n, k, newd) DO BEGIN \ 
23550 zm4_upper(n,k,newd,r); \ 
23560 r := Ref; \ 
23570 n := n + 1; \ 
23580 END ( while ); \ 
23590 CondWipeRef (upper, n, k, newd); CondWipeRef (upper, n + 1, k, newd + 1); \ 
23600 WipeMark (upper, n, k); \ 
23610 END ( zm3_zm4_upper ); \ 
23620 \ 
PROCEDURE Update{b: BoundKind; n,k,newd: integer; r: Reference}; \ 
23640 \ 
BEGIN 
23650 \ 
23660 $ \text{SET OMIT = NOT TESTOUTPUT}$; \ 
23670 begin \ 
23680 WRITE(TT, 'Checking bound (', b, ',', n, ',', k, ',', newd, '); r'); \ 
23690 $ \text{POP OMIT}$; \ 
23700 IF (x < k) AND (k < x) AND (n > MaxBlIndex) AND (newd > 1)) \ 
23710 OR IsImprovement (b, n, k, newd) THEN BEGIN \ 
23720 $ \text{SET OMIT = NOT TESTOUTPUT}$; \ 
23730 WRITE(TT, 'interesting'); \ 
23740 $ \text{POP OMIT}$; \ 
23750 depth := depth + 1; \ 
23760 CASE b OF \ 
23770 lower: BEGIN \ 
23780 zm4_lower (n, k, newd, r); \ 
23790 zm2_zm4_lower (n, k, newd); \ 
23800 UpdateLower (n, newd, k, newd + 1) DIV 2, A_Ref); ( A ) \ 
23810 BT (n, k, newd); \ 
23820 IF IsInTable (n, k) THEN BEGIN \ 
23830 zm6 (n, k, newd); \ 
23840 zm3_zm4_B (n, k, newd); \ 
23850 zm2_zm4_B1 (n, k, newd); \ 
23860 END ( if ); \ 
23870 END ( lower bound ); \ 
23880 \ 
upper: BEGIN \ 
23890 zm3_zm4_upper (n, k, newd, r); \ 
23900 zm2_zm4_upper (n, k, newd, r); \ 
23910 IF IsInTable (n, k) THEN BEGIN \ 
23920 zm3 (n, k, newd); \ 
23930 zm2_zm4_B3 (n, k, newd); \ 
23940 zm2_zm4_B2 (n, k, newd); \ 
23950 END ( if ); \ 
23960 END ( upper bound ); \ 
23970 \ 
BEGIN \ 
23980 depth := depth + 1; \ 
23990 $ \text{SET OMIT = NOT TESTOUTPUT}$; \ 
24000 begin \ 
24010 WRITE(TT, 'Exit bound (', b, ',', n, ',', k, ',', newd, '); r'); \ 
24020 $ \text{POP OMIT}$; \ 
24030 END ( then ); \ 
24040 ELSE WRITE(TT, 'useless'); \ 
24050 $ \text{POP OMIT}$; \ 
24060 END ( UpdateBt ); \ 
24070 \ 
24080 BEGIN ( UpdateBt );
write(Report,
'Considering bound ("IDb","",len:3,"",dim:3,"",newdist:2,""),
end);  
writeln(List,("IDb","",len:3,"",dim:3,"",newdist:2,"",r));  
Lmodified := false;
NoRemarksYet := true;  (first remark still has to be made)
pt := RUNTIME;
depth := 0;
Update(b,len,dim,newdist,r);
0:  (in case of violation exit)
pt := RUNTIME - pt;  TotUpdates := TotUpdates + 1;
IF Lmodified THEN BEGIN
  TotUseful := TotUseful + 1;
  Bmodified := true;
  IF NoRemarksYet THEN write(Report,'useful');
END;
ELSE IF NoRemarksYet THEN write(Report,'USELESS');
writeln(Report,' PT=',pt:1:2);  
END  (UpdateBt);

End.
**PROCEDURE StartPage ;**
**BEGIN**
**PagesPrinted := PagesPrinted + 1 ;**
**IF (PagesPrinted MOD MaxPage = 0) THEN BEGIN**
  **CLOSE(Pr) ; rewrite(Pr) END ;**
**END ( StartPage ) ;**

**PROCEDURE PrintBlock(i,j: integer) ; ( i,j assumed in range )**
**CONST SqSize = 5 ;**
**VAR line,col,height,width: integer ;**
**startline,endline,startcol,endcol: integer ;**

**PROCEDURE VertSep ;**
**BEGIN**
  **IF (col MOD SqSize = 0) THEN write(Pr,'/')**
  **ELSE write(Pr,'-')**
**END ( VertSep ) ;**

**PROCEDURE HorSep ;**
**BEGIN**
  **IF (line MOD SqSize = 0) AND (line <> SqSize) THEN BEGIN**
    **writeln(Pr,'--') ;**
    **col := startcol**
    **WHILE (col <> endcol) DO BEGIN**
      **VertSep ;**
      **write(Pr,'--') ;**
      **col := col + 1**
    **END ( while ) ;**
  **END ( if ) ;**
**END ( HorSep ) ;**

**PROCEDURE xValues ;**
**BEGIN**
  **write(Pr,' n,k') ;**
  **col := startcol**
  **WHILE (col <> endcol) DO BEGIN**
    **VertSep ;**
    **IF (col < 100) THEN write(Pr,',',col:3,' ')**
    **ELSE write(Pr,',',col:2,' ')**
  **END ( while ) ;**
  **writeln(Pr,' n,k')**
**END ( xValues ) ;**

**PROCEDURE PrintRef(r1, r2: char; rightjustified: boolean) ;**
**BEGIN**
  **IF rightjustified THEN**
    **IF (r2 = ' ') THEN write(Pr,r2,r1)**
   **ELSE write(Pr,r1,r2)**
  **ELSE**
    **IF (r1 = ' ') THEN write(Pr,r2,r1)**
    **ELSE write(Pr,r1,r2)**
**END ( PrintRef ) ;**

**Appendix IV. Program Listings**
PROCEDURE BtValues;
VAR x: integer;
BEGIN
write(Pr,'*':4);
col := startcol;
WHILE (col <> endcol) DO BEGIN
VertSep; col := col + 1;
IF (col > (line) OR (line > MaxBTIndex THEN write(Pr,'*':5);)
ELSE BEGIN
WHITH Bt(line,col]) DO BEGIN
PrintRef(refch1,refch2,NOT ISOpenCase(line,col) AND (dist < lOll
1:2 ord(unehanged);
END;
WITH Bt(eol,line] DO BEGIN
:= 2*1 ord(unehanged)
CASE I
OF
0: write(Pr,'*');
1: write(Pr,'<');
2: write(Pr,'>');
3: write(Pr,'I')
ELSE BEGIN
Bt(line,eol].dist:2,'-l1
END{ open case}
ELSE write(Pr,Bt[line,eol].dist:3,'I')
END {while}
VertSep
writelnIPr,':35,'Upper and lower Bounds on d_max(n,k) for Binary Linear Codes'
ELSE writeln(Pr1
width := 20, startcol := 20*j, endcol := starcol + width;
line := startline;
 writeln(Pr,1
IF (i < 4) THEN height := 25 ELSE height := 27 ;
writeln(Pr,1
width := 20, startcol := 20*j; endcol := startcol + width;
line := startline;
 kValues; writeln(Pr1
WHILE (line <> endline) DO BEGIN
HorSep; line := line + 1;
PROCEDURE PrintALL;
VAR i,j: Integer;
BEGIN
FOR j := 0 TO 6 DO
  FOR i := 0 TO 4 DO
    If IsBlock(i,j) THEN PrintBlock(i,j);
    If ToBePrinted(i,j) THEN PrintBlock(i,j) ;
END { PrintALL } ;
PROCEDURE PrintModified;
VAR i,j,count: integer;
BEGIN
  count := 0 ; { # blocks printed }
  FOR j := 0 TO 6 DO
    FOR i := 0 TO 4 DO
      If IsBlock(i,j) THEN BEGIN
        PrintBlock(i,j) ;
        count := count + 1 ;
      END { if } ;
    END { for i } ;
  END { for j } ;
  writeln(Report, '"all (+,count)" blocks with ',
     'unprinted modifications have been printed.');
END { PrintModified } ;
PROCEDURE wallpaper;
VAR i,j: Integer;
BEGIN
  FOR j := 0 TO 4 DO
    FOR i := 0 TO 4 DO
      If IsBlock(i,j) THEN PrintBlock(i,j)
        ELSE IF (i = 0) AND (j = 3) THEN PrintBlock(4,j+2)
          ELSE BEGIN
            StartPage;
            writeln(Pr,"FILLER PAGE FOR WALLPAPER") ; page(Pr) ;
          END ;
          writeln(Report,"Wallpaper printed.");
  END { wallPaper } ;
PROGRAM InitialBt (
 output,
  idf : FILE <areasize=1,area=1,title="INITIAL/LISTING">,
  luf : FILE <areasize=1,area=1,title="INITIAL/UPDATES">;
)

FUNCTION IsOpenCase(n,k: BtIndex): boolean;
BEGIN < k <= n assumed >
IsOpenCase := (Bt(n,k).dist < Bt(k,n).dist);
END;

PROCEDURE JenInitialBt;
( Generate initial table of bounds Bt )
VAR n,m,n2,k,k2,c,duald: integer;

PROCEDURE JenUpdate(b: BoundKind; n,k,newd: integer; r: Reference);

PROCEDURE JenMove(b: BoundKind; n,k,newd: integer; r: Reference);
WITH Bt(n,k) DO BEGIN (P2)
 IF odd(newd) THEN newd := newd - 1;
 IF (dist = newd) THEN refch1 := 1;
 END (with);
 END (upper bound);
 END (case);
 END (improve);
 BEGIN (geninitialbt)
 BEGIN (lower bounds)
 FOR n := 1 TO maxindex DO
 FOR k := 1 TO n DO WITH Bt(n,k) DO BEGIN
 refch1 := 1; refch2 := 1;
 IF (k = 1) THEN dist := n;
 ELSE IF (k = n) THEN dist := 1;
 ELSE BEGIN
 IF (odd(dist)) THEN dist := dist + 1;
 END (if);
 END (for k, for n, with Bt[n,k]);
 IF NOT odd(n) THEN BEGIN
 n := n - 1; n2 := n DIV 2;
 WHILE (n <= n2) DO BEGIN
 c := Bt[n,k].dist + Bt[n2,k].dist;
 IF (c > dist) THEN BEGIN
 imove(lower,n,k,c,ref);
 END (imove);
 END (while);
 END (if);
 END (for n, for k, with Bt[n,k]);
 (A: residual code)
c := (dist1) DIV 2 ; (* ceiling(dist1/2) *)
 IF (c > Bt[n-dist1,k-1].dist) THEN
 genupdate(lower,n-dist1,k-1,c,4,ref);
 END (if);
 END (if);
 (P2 (parity check))
 n := n + 1; n2 := n DIV 2;
 (concatenation)
nl := k; n2 := n - k; (n1+n2 = n)
 WHILE (n1 <= n2) DO BEGIN
 c := Bt[n1,k].dist + Bt[n2,k].dist;
 IF (c > dist) THEN BEGIN
 imove(lower,n,k,c,ref);
 END (imove);
 END (while)
 (D: construction u,v+w)
 IF NOT odd(n) THEN BEGIN
 n2 := n DIV 2 ; k1 := 1 ; k2 := k - 1 ; (k1+k2 = k)
 WHILE (k1 <= k2) DO BEGIN
 c := min(Bt[n2,k1].dist, Bt[n2,k2].dist);
 IF (c > dist) THEN BEGIN
 imove(lower,n,k,c,0,ref);
 (check [n-1,k-1] (P4) & [n-1,k] (P3))
 IF (dist > Bt[n-1,k-1].dist) THEN
 imove(lower,n-1,k-1,dist,4,ref);
 IF (dist-1 > Bt[n-1,k].dist) THEN
 imove(lower,n-1,k,dist-1,ref);
 END (imove);
 END (while);
 END (if);
 END (for n, for k, with Bt[n,k]);
 (exceptional case, maxindex is odd: 0 followed by P3, P4 or A)
 IF odd(maxindex) THEN BEGIN
 n := maxindex + 1; n2 := n DIV 2;

FOR k := 2 TO n-2 DO BEGIN
k2 := MIN(k, n-k-1); k1 := k-k2 := (k1+k2 = k)
WHILE (k1 <= k2) DO BEGIN
C := MIN(Bt(n2, k1).dist, 2*Bt(n2, k2).dist) ; (lower, n, k, c)
IF (c > Bt(n1, k-1).dist) THEN (P4)
IF (odd(c)) THEN Improve(upper, lower, n-l, c) ;
ELSE Improve(lower, n-l, k-l, c) ;
END
END (for k)
END (if)

FOR n := 1 TO MaxBtIndex DO BEGIN
FOR k := 1 TO n DO WITH Bt(k, n) DO BEGIN
refch1 := '1'; refch2 := '1';
IF (k = 1) THEN dist := n
ELSE IF (k = n) THEN dist := 1
ELSE BEGIN (1 < k < n)
(P3 and P4)
dist := MIN(Bt(k-1, n-1).dist, Bt(k, n-1).dist+1)
END
WHILE ((dist+1) DIV 2 > Bt(k-1, n-dist).dist) DO BEGIN
Improve(upper, n, k, dist-1, E_Ref)
END (while)

(B: upper bound via "dual code" construction P4*)
IF (n-k <= k) AND (k < n-1) THEN BEGIN
( n, possibly n-k = k)
duald := Bt(n-k, n).dist ; (upper bound for dual of (n, k))
WHILE (dist > Bt(k-duald+1, n-duald).dist) DO BEGIN
Improve(upper, n, k, Bt(k-duald+1, n-duald).dist, B_Ref)
duald := Bt(n-k, n).dist ; (when n-k = k: duald = dist)
END (while)
c := Bt(n-k-dist+1, n-dist).dist
IF (duald > c) THEN
GenUpdate(upper, n, k, c, B_Ref)
END (if)

(B: lower bound via "dual code" construction P4*)
IF (k < n-1) THEN BEGIN
( n-k in fact dist < n-k+1)
c := Bt(n, n-k).dist
IF (c > Bt(n, n-k-dist+1).dist) THEN
GenUpdate(lower, n, n-k, n-dist+1, c, B_Ref)
END (if)
END (for n, for k, with Bt(k, n))
END (for n := 1 TO MaxBtIndex DO Bt(k, n).unsolved := 0)
FOR k := 1 TO MaxBtIndex DO Bt(k, 0).unsolved := 0;
END
FOR n := 1 TO MaxBtIndex DO
  FOR k := 1 TO n DO
    IF isOpenCase(n, k) THEN BEGIN
      WITH Bt[0,0] DO unsolved := unsolved + 1;
      WITH Bt[0,k] DO unsolved := unsolved + 1;
      WITH Bt[n,0] DO unsolved := unsolved + 1;
    END (if);
  END (for k);
END (for n);

BEGIN (InitalBt)
rewrite(output); rewrite(iuf);
writeln('Initial Bound Table Generator, MaxBtIndex = ', MaxBtIndex);
GenInitialBt;
rewrite(iuf, output);
CLOSE(iuf, CRUNCH);
write('write Bt to file');
rewrite(ibf);
ibf := Bt;
put(ibf);
CLOSE(ibf);
END (InitialBt).