

On the LP Which Finds a MMAE Stack Filter

Moncef Gabbouj, *Member, IEEE*, and Edward J. Coyle, *Member, IEEE*

Abstract—Two methods are proposed to modify the linear program (LP) developed in [1] to find a stack filter which minimizes the mean absolute error (MAE).

In the first approach, the number of constraints is substantially reduced at the expense of requiring a zero-one LP to solve for an optimal filter. This scheme reduces the number of constraints from $O(n2^n)$ to $O(2^n)$, which is exactly the cardinality of the set of possible binary vectors which can appear in the window of the filter.

In the second approach, the LP is transformed into a max-flow problem. This guarantees that the problem can be solved in time which is a polynomial function of the number of variables in the LP, as opposed to the worst case exponential time that may occur with the simplex method. It also allows the many fast algorithms for the max-flow problem to be used to find an optimal stack filter.

Recursive algorithms for the construction of the window width n constraint matrix for both the original LP and the max-flow modification are also provided.

I. INTRODUCTION

IT WAS shown in [1] that a stack filter which minimizes the mean absolute error (MAE) can be found via the following linear program (LP):

$$\min \sum_{j=1}^{2^n} [\alpha_j P_j(0|x_j) + \beta_j P_j(1|x_j)] \quad (1.1)$$

subject to the decision constraints: For each vertex of the unit n cube

$$P_f(1|x) + P_f(0|x) = 1 \quad (1.2)$$

and for each edge ($r0s$, $r1s$) of the n cube

$$P_f(1|r1s) \geq P_f(1|r0s). \quad (1.3)$$

The α_j 's and the β_j 's in (1.1) are constant cost coefficients and $P_f(0|x_j)$ ($P_f(1|x_j)$) is the unknown filter decision probability and is equal to the probability that the filter outputs a 0 (a 1) when it observes x_j in its window. These are referred to as the stack filter decision variables later in this paper. This LP contains 2^{n+1} variables and $(n2^{n-1} + 2^n)$ constraints when the optimization is performed over all stack filters of window width n . As shown in [1, appendix] the number of variables in the LP can be reduced to 2^n and the number of constraints can be reduced to $n2^{n-1} + 1$.

Manuscript received January 1, 1990; revised September 21, 1990.

M. Gabbouj is with the Research Institute of Information Technology, Tampere University of Technology, SF-33101 Tampere, Finland.

E. J. Coyle is with the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.

IEEE Log Number 9102462.

One purpose of this paper is to further investigate this LP to determine if the number of constraints can be further reduced. Note that the number of variables cannot be reduced below 2^n since that is the number of different binary vectors that can be seen in a window of size n ; the output of the positive Boolean function on each level of the stack filter must be specified for each input it might observe.

We will first show that, except for the nonnegativity constraints, the reduced set of constraints obtained in [1, appendix] is the smallest set of constraints that may be used in any LP to find an optimal stack filter.

Under certain conditions, though, this set of constraints can be reduced to

$$2^n - C_{\lfloor n/2 \rfloor}^n + 1$$

where C_k^n is the usual binomial coefficient, and $[x]$ is the integer part of x . While this results in a significant savings in storage space for the constraint matrix, it is achieved at the expense of increased execution time. The increased time complexity is due to the fact that a zero-one LP must be used to find the optimal filter under the reduced set of constraints. The reduction in storage space can, however, make this a very reasonable tradeoff.

Another problem with the LP is that its worst case time complexity, assuming a standard simplex algorithm is used, is doubly exponential in the window width n : $O(a^{2^n})$, where a is some constant. We will show that this LP can be transformed into a max-flow problem, which reduces its time complexity to $O(n^3 2^{3n})$. This time complexity is still exponential, but since the number of variables must be 2^n it appears unavoidable. This reduction in time complexity is achieved with only a small increase in storage requirements over the original LP.

As we prove the space and time complexity results discussed above, algorithms for the construction of the constraint matrix for both the LP and the max-flow approach to finding an optimal stack filter will be provided. The algorithms are recursive in n , the filter window width.

II. FIRST-ORDER REDUCTION OF THE NUMBER OF CONSTRAINTS

In this section, we will show that if an LP is used to find an optimal stack filter, then the set of constraints obtained in [1, appendix] is indeed the smallest set except for the nonnegativity constraints, which can be reduced to a single constraint.

The LP for finding an optimal stack filter of window width n described in (1.1)–(1.3) can be rewritten in matrix form as

$$\begin{aligned} \min c^T x \\ \text{subject to } A_n x \leq \mathbf{1}, \quad x \geq 0. \end{aligned}$$

The meanings of the symbols in the above LP are defined below.

First, the ordering of the stack filter decision variables (which were defined earlier) in the vector x is as follows: the i th entry in x (the indexing starts at 1) is the filter decision variable $x_i = P_f(1|B_n(i-1))$, where $B_n(m)$ means the binary string of length n whose value in base 10 is m . For instance, x_{14} for a window width 5 filter is specified as follows:

$$x_{14} = P_f(1|B_5(13)) = P_f(1|01101).$$

It specifies the filter output when the vector 01101 is in the window of the filter.

The set of inequalities $A_n x \leq \mathbf{1}$ are known as the stacking constraints. These equations may be rewritten as follows:

$$x_i - x_j \leq 0 \quad \text{for all } x_i \leq x_j \quad \text{and} \quad d_H(x_i, x_j) = 1 \quad (2.1)$$

$$x_{2^n} \leq 1 \quad (2.2)$$

in which x_k is a binary window vector of length n ; the k th entry of the vector of decision variables x is $x_k = P_f(1|x_k)$, the decision variable associated with the window vector x_k ; and, $d_H(x_i, x_j) = 1$ means x_i differs from x_j in exactly one entry, such as the m th (so we might have $x_i(m) = 0$ and $x_j(m) = 1$).

The notation $x_i \leq x_j$ means that each entry of the binary vector x_i is less than or equal to the corresponding entry in x_j . Note that “ \leq ” specifies only a partial ordering on the set of vectors of length n . The vectors x_i and x_j are said to be comparable if either $x_i \leq x_j$ or $x_j \leq x_i$; otherwise, they are said to be incomparable. If we consider the cube formed in n -dimensional space by the 2^n binary window vectors of length n , the edges of this cube (which is also known as a Hasse cube) specify exactly the set of inequalities in (2.1), if vertices x_i and x_j are connected by an edge in this diagram, then either $x_i \leq x_j$ or $x_j \leq x_i$. Since there are exactly $n2^{n-1}$ edges in an n cube, (2.1) specifies $n2^{n-1}$ inequalities.

Finally, the vector c^T contains the cost coefficients corresponding to the stack filter decision variables; and $\mathbf{1}$, which is a vector of all zeros except for a one in the last entry, is the requirement vector of the LP.

To show that all of these $(n2^{n-1} + 1)$ inequalities are required for the LP, it suffices to show that if any one of them is removed, then there exists a vector x of decision variables which violates the stacking constraints but is feasible under the reduced set of constraints.

It is clear that (2.2) is required; otherwise, some or even all of the variables may become unbounded.

Now, suppose that the inequality

$$x_i - x_j \leq 0$$

is removed. Then one stacking constraint will be violated if there exists a feasible point (with respect to the reduced set) where $x_i = 1$ and $x_j = 0$.

Set the decision variables for the binary window vectors which are comparable to x_i and x_j as follows:

$$x_k = 1 \quad \text{for all } x_k \geq x_i, \quad k \neq j \quad (2.3)$$

and

$$x_k = 0 \quad \text{for all } x_k \leq x_j, \quad k \neq i. \quad (2.4)$$

Recall that the reduced set of constraints is

$$\begin{aligned} x_l - x_k \leq 0 \quad \text{for all } x_l \leq x_k \quad \text{and} \quad d_H(x_l, x_k) = 1 \\ \text{except } l = i \quad \text{and} \quad k = j. \end{aligned} \quad (2.5)$$

From (2.3), we get $x_i - x_k = 0$ for all k such that $x_k \geq x_i$, which satisfies (2.5). Equation (2.4) gives $x_k - x_j = 0$ for all k such that $x_k \leq x_j$, which also satisfies (2.5).

This procedure does not assign values to all the x_k 's. For the ones which are left, any of the many assignments which satisfy (2.5) can be chosen. Such a decision vector x constructed from (2.3) and (2.4) is feasible. However, since $x_i - x_j \leq 0$ does not hold, x violates the stacking property and hence, the inequality $x_i - x_j \leq 0$ is not redundant.

The above procedure can be applied to any single inequality constraint or any set of inequality constraints to show that none of them is redundant. Therefore, the set containing those $(n2^{n-1} + 1)$ constraints is a minimal set of constraints for the LP.

On the other hand, the set of nonnegativity constraints

$$x_i \geq 0, \quad i = 1, 2, \dots, 2^n$$

can be reduced to just one constraint, $x_1 \geq 0$, because the stacking property will then ensure the nonnegativity of all remaining decision variables.

The above analysis proves the following theorem.

Theorem 1: The smallest constraint matrix that can be used with an LP to find an optimal stack filter has $(n2^{n-1} + 1)$ rows and the LP has one nonnegativity constraint. \square

III. SECOND-ORDER REDUCTION IN THE NUMBER OF CONSTRAINTS

Here, we will show that a further reduction in the number of constraints is possible if a zero-one LP is used instead of an LP. This approach is preferred, for instance, when computer storage is a harder constraint than execution time.

To show the reduction in the number of constraints, we will make use of the stacking diagram defined below.

Definition 1: A stacking diagram for a window width n stack filter is an undirected graph $G = (V, E)$ where V is the set of all n -tuples and $E = \{(u, v): u, v \in V, d_H(u, v) = 1\}$. \square

Note that G is just the graph of the nearest neighbor partial ordering between n -tuples. A suitable representa-

tion (for this application) of this graph is when it is drawn as $n + 1$ levels each containing $C_k^n, k = 0, 1, \dots, n$, incomparable n -tuples. Now, it is clear that the edges in E are just those links between n -tuples in neighboring levels of the graph. For an illustration, see Fig. 1.

One can easily verify that the number of edges in G is equal to $n2^{n-1}$. Each one of these edges describes an inequality constraint in the LP. Starting from the top of the graph, there is one n -tuple at level 0 (the n -tuple of all 1's, call it x^0). There are n connections between this level and the level below (level $k = 1$). Let the n -tuples on level 1 be called $x_1^1, x_2^1, \dots, x_n^1$. Previously, in order to account for the stacking constraints, we required the following:

$$\begin{aligned} x_1^1 - x^0 &\leq 0 \\ x_2^1 - x^0 &\leq 0 \\ &\vdots \\ x_n^1 - x^0 &\leq 0. \end{aligned}$$

The claim is that the above n inequalities are equivalent (in the sense that they have the same effect on the optimization problem) to a single inequality. Let us just add these inequalities:

$$\sum_{i=1}^n x_i^1 - nx^0 \leq 0.$$

Now, if any of the x_i^1 's is a 1, it will force x^0 to be 1; hence, the stacking constraints are fulfilled. (Keep in mind the assumption that all variables are 0-1 variables.)

Example 1: Consider the case $n = 2$. In the top level, level 0, we have $x^0 = 11$. In the level below, level 1, we have $x_1^1 = 10$ and $x_2^1 = 01$. The stacking constraints are then

$$\begin{aligned} x_1^1 - x^0 &\leq 0 \\ x_2^1 - x^0 &\leq 0. \end{aligned}$$

Adding these two inequalities, we get

$$x_1^1 + x_2^1 - 2x^0 \leq 0.$$

Now it is clear that if x_1^1 or x_2^1 is equal to 1, then x^0 is forced to be 1. \square

Proceeding along this path, take x_1^1 from level 1. It has $n - 1$ connections to the level below (level 2). Group the corresponding $(n - 1)$ inequalities into a single one, as was done above. Perform the same manipulation on x_2^1, \dots, x_n^1 . Repeat the above procedure for all n -tuples in levels 2, 3, $\dots, \lfloor n/2 \rfloor - 1 = \text{mid}$. When manipulations on level mid are completed, move to level $k, k = n, n - 1, \dots, \lfloor n/2 \rfloor + 1$, and repeat the above procedure, except that this time, look for connections between the current level and the level above, instead.

After the above procedure has been completed, every node on each level has been visited once and only once, yielding one inequality constraint, except for those nodes

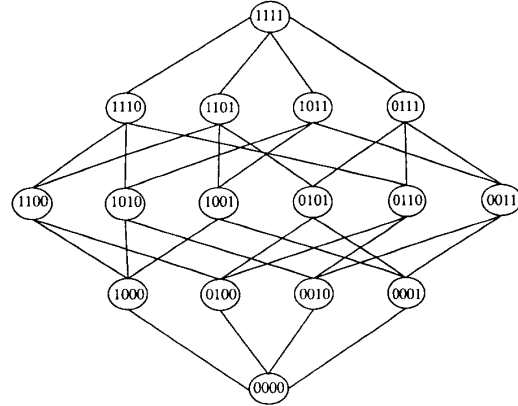


Fig. 1. Stacking diagram for window width four stack filters.

TABLE I
NUMBER OF CONSTRAINTS FOR LP AND ILP FOR VARIOUS WINDOW SIZES

| n | $n2^{n-1} + 1$ | $2^n - C_{\lfloor n/2 \rfloor}^n + 1$ |
|-----|----------------|---------------------------------------|
| 3 | 13 | 6 |
| 5 | 81 | 23 |
| 7 | 449 | 94 |
| 9 | 2305 | 387 |
| 11 | 11 265 | 1587 |
| 13 | 53 249 | 6477 |
| 15 | 245 761 | 26 334 |
| 25 | 419 430 401 | 28 354 133 |

in level $\lfloor n/2 \rfloor$, which have not been visited. There are $C_{\lfloor n/2 \rfloor}^n n$ -tuples on this level, and hence, the above procedure reduces the number of constraints to

$$2^n - C_{\lfloor n/2 \rfloor}^n + 1.$$

(The additional constraint comes from (2.2).) This conclusion is stated in the following theorem.

Theorem 2: Only $(2^n - C_{\lfloor n/2 \rfloor}^n + 1)$ constraints are required for a zero-one LP to solve the optimization problem. \square

This scheme produces an 80% reduction in the number of constraints when optimizing over a 3×3 window. Table I gives the number of constraints required by each approach as a function of the window width n of the filter.

Remark: Note that by using an ILP rather than an LP, we can solve a larger optimization problem (by increasing the filter's window width from 9 to 13) without increasing the order of the number of constraints required.

However, there are two drawbacks associated with this method. The first is that the new problem must be solved using a zero-one LP rather than a regular LP, the former being harder than the latter. However, for small window widths (less than nine), simulation results show no significant difference between CPU time required to solve both types of problems, that is, both problems are solved very quickly. A significant difference in time requirement is expected in the case of larger window widths (larger than nine).

The second drawback of this method is that the reduced constraint matrix is no longer totally unimodular (defined later). Total unimodularity is a very desirable property which we will exploit in the next section to reduce the time complexity of the LP.

IV. TRANSFORMATION TO A NETWORK PROBLEM

In this section, we will present a method to reduce the complexity of the original LP by exploiting the structure of its constraint matrix. This approach significantly reduces the time complexity with only a minor increase in space requirements.

Recall that [3] the constraint matrix of the original LP was stated to be totally unimodular. In this section, we will show that this is the case and then, using this property, we will transform the LP problem into a network problem. This will result in a reduction in the complexity of the LP from a worst case exponential time algorithm using the standard simplex method to an algorithm which is a polynomial time algorithm in the number of decision variables.

First we present a definition and some theorems which will be used in the sequel.

Definition 2: [4, p. 316]: A square, integer matrix B is called unimodular (UM) if its determinant $\det(B) = \pm 1$. An integer matrix A is called totally unimodular (TUM) if every square, nonsingular submatrix of A is UM. \square

Theorem 3: [4, p. 317]: An integer matrix A with $a_{ij} = 0, \pm 1$ is TUM if no more than two nonzero entries appear in any column, and if the rows of A can be partitioned into two sets I_1 and I_2 such that 1) if a column has two entries of the same sign, their rows are in different sets; 2) If a column has two entries of different signs, their rows are in the same set. \square

The above theorem is equivalent to the following.

Theorem 4: [5, p. 269]: An integer matrix A is TUM iff each collection of columns of A can be split into two parts so that the sum of the columns in one part minus the sum of the columns in the other part is a vector whose entries are only 0, +1, and -1. \square

Total unimodularity is preserved under certain operations, some of which are listed in the following theorem.

Theorem 5: [5, p. 280]: The following statements are equivalent:

- i) The matrix A is TUM.
- ii) The matrix A^T is TUM.
- iii) The matrix \bar{A} obtained from A by appending a row or a column with one nonzero entry of ± 1 is also TUM.
- iv) The matrix \hat{A} obtained from A by permuting some columns of A is again TUM. \square

Now, we will show that the constraint matrix of the original LP is TUM. The structure of the constraint matrix will be made very clear in the proof of this theorem. The proof also provides a recursive algorithm for constructing the constraint matrix for window width n from the constraint matrix for window width $n - 1$.

Theorem 6: Let A_n be the constraint matrix of the original LP for a window width n stack filtering problem. Then A_n is TUM for all $n \geq 2$.

Proof: For $n = 2$, define

$$A_2 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} W_2 & \mathbf{0} \\ \mathbf{0} & W_2 \\ I_2 & -I_2 \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

where $W_2 = [1 \ -1]$, I_k is a $k \times k$ identity matrix, $\mathbf{0}$ is a matrix (or vector) of zeros of appropriate dimension, and $\mathbf{1}$ is a row vector of all zeros except the last entry, which is a 1. The constraint equations for this LP for the window width 2 case are then

$$A_2 x \leq \mathbf{1}$$

where the requirement vector $\mathbf{1}$ is a length 5 column vector which is all zeros except the last entry, which is a 1. The ordering of the stack filter decision variables in the vector x was described in Section II.

By enumeration, it is possible to show that A_2 satisfies the conditions in Theorem 4 and that it is therefore TUM. Alternatively, one can take the transpose of A_2 , which is

$$A_2^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 1 \end{bmatrix}$$

and apply Theorem 3, with $I_2 = \emptyset$. Then, by Theorem 5, parts i) and ii), A_2 is TUM. The constraint matrix A_n , $n \geq 3$, can be obtained from W_n of dimensions $((n - 1)2^{n-2} \times 2^{n-1})$, which is defined recursively as follows:

$$W_n = \begin{bmatrix} W_{n-1} & \mathbf{0} \\ \mathbf{0} & W_{n-1} \\ I_{2^{n-2}} & -I_{2^{n-2}} \end{bmatrix}$$

For general n

$$A_n = \begin{bmatrix} W_n & \mathbf{0} \\ \mathbf{0} & W_n \\ I_{2^{n-1}} & -I_{2^{n-1}} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Note that W_n is just the matrix A_{n-1} without its last row. A_n is a $(n2^{n-1} + 1) \times (2^n)$ matrix.

The constraint equations for the LP which finds an optimal window width n stack filter are

$$A_n x \leq \mathbf{1}$$

where the length $n2^{n-1} + 1$ column vector $\mathbf{1}$ has a one in its last entry and zeros everywhere else, and where the i th

entry of x (the indexing is started at 1) corresponds to the filter decision variable $P_f(1|B_n(i-1))$.

Suppose that A_{n-1} is TUM. We will show that A_n is also TUM. Recall that W_{n+1} is the matrix A_n without its last row. Since the last row of A_n is $\mathbf{1}$ (a row of all zeros except the last one, which is a 1), then, by Theorem 5 iii), if W_{n+1} is TUM, then A_n is also TUM.

Next, we will show that W_{n+1} is TUM, again by induction. $W_2 = [1 \ -1]$ is clearly TUM (the determinant of every square nonsingular submatrix of W_2 is ± 1). Assume that W_n is TUM, we shall show that W_{n+1} is also TUM.

Let's examine W_{n+1}^T , instead. It is easy to show (by induction, also) that every column of W_{n+1}^T contains only two nonzero entries: +1 and -1. Now, apply Theorem 3, with $I_2 = \emptyset$. This shows that W_{n+1}^T is indeed TUM and hence, by Theorem 5 i) and ii), W_{n+1} is TUM. Therefore, we have, by induction, that A_n is TUM for all n . \square

A_n will now be transformed into a network matrix using operations ii) and iii) of Theorem 5. First, we obtain the dual of the original LP problem. The primal is

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & A_n x \leq \mathbf{1} \quad \text{and} \quad x \geq 0 \end{aligned}$$

where all the symbols have been defined in Section II.

The dual of this LP is

$$\begin{aligned} \min \quad & \bar{\mathbf{1}}^T \lambda \\ \text{subject to} \quad & A_n^T \lambda \geq c \quad \text{and} \quad \lambda \geq 0. \end{aligned}$$

Put the dual in standard canonical form by adding 2^n surplus variables as follows:

$$\sum_{j=1}^M a_{ij} \cdot \lambda_j - s_i = c_i, \quad s_i \geq 0$$

where a_{ij} is the i, j th entry of A_n^T . Let $\bar{A}_n = [A_n^T \ -I_{2^n}]$, which is $(2^n) \times (n2^{n-1} + 2^n + 1)$, denote the new constraint matrix for the dual in standard canonical form. Note that \bar{A}_n is TUM since A_n is and because of parts ii) and iii) of Theorem 5. Note also that every column of A_n^T (except the last) has 2 nonzero entries: 1 and -1.

Add one transshipment node (a node with zero requirement) and an arc from the node corresponding to the last row of A_n^T to this new node. This adds one row to \bar{A}_n to produce

$$\tilde{A}_n = \begin{bmatrix} A_n^T & -I_{2^n} \\ -\mathbf{1} & \mathbf{0} \end{bmatrix}$$

where $-\mathbf{1}$ is a vector whose entries are all zeros except the last one, which is -1. This is a $(2^n - 1) \times (n2^{n-1} + 2^n + 1)$ matrix; and it is TUM.

To every node corresponding to a row in A_n^T , attach a transshipment node through an arc leaving the transshipment node. This will transform \tilde{A}_n into

$$\hat{A}_n = \begin{bmatrix} \tilde{A}_n \\ \mathbf{0} \ I_{2^n} \end{bmatrix} = \begin{bmatrix} A_n^T & -I_{2^n} \\ -\mathbf{1} & \mathbf{0} \\ \mathbf{0} & I_{2^n} \end{bmatrix}$$

which is a $(2^{n+1} + 1) \times (n2^{n-1} + 2^n + 1)$ matrix.

It is straightforward to show that every column in \hat{A}_n has exactly one +1 and one -1 entry. Thus, \hat{A}_n is an arc-node incidence matrix and the dual has been transformed into a max-flow problem that can be solved using an $O(|V|^3)$ max-flow algorithm [4, p. 211], where $|V|$ is the number of nodes in the network.

The way $|V|$ is related to the window width of the filter is discussed next. Recall that the dual was

$$\begin{aligned} \min \quad & \mathbf{1}^T \lambda \\ \text{subject to} \quad & A_n^T \lambda \geq c \quad \text{and} \quad \lambda \geq 0. \end{aligned}$$

After the above manipulations, the LP problem will be

$$\begin{aligned} \min \quad & \mathbf{1}^T \pi \\ \text{subject to} \quad & \hat{A}_n \pi = \bar{c} \quad \text{and} \quad \pi \geq 0 \end{aligned}$$

where $\pi = [\lambda^T | s_1 \ s_2 \ \dots \ s_{2^n}]^T$ and $\bar{c} = [c^T \ \mathbf{0}]^T$. Therefore,

$$|V| = n2^{n-1} + 2^n + 1 = \left(1 + \frac{n}{2}\right) 2^n + 1.$$

The time complexity of the max-flow algorithm is

$$O(|V|^3) = O(n^3 \cdot 2^{3n})$$

which is significantly better than the worst case complexity of the simplex algorithm when it is applied to the primal problem. In other words,

$$O(n^3 \cdot 2^{3n}) \ll O(a^{2^n})$$

for $n > 3, a = e$. However, one must remember that this comparison is against the worst case performance of the simplex algorithm. This makes it very difficult to see the actual difference in the computational requirement of both approaches in the feasible case of small window widths (feasibility here applies to the use of the LP for window widths less than or equal to five). Beyond this window size, the use of the LP becomes completely impractical in both computation and storage requirements and the use of the max-flow formulation becomes the alternative.

Finally, rather than giving an interpretation of the max-flow problem, we shall consider the interpretation of its dual since it is the problem we started with. The dual of the max-flow problem is a min-cut problem. At optimality, the solution of the min-cut problem cuts the decision space into two disjoint sets, W and \bar{W} , such that $x_i = 0$ for all $x_i \in W, x_j = 1$ for all $x_j \in \bar{W}$ and the capacity $C(W, \bar{W})$ of the cut has the maximum value. Fig. 2 shows an instance of an optimal solution.

W and \bar{W} are the largest sets of window vectors for which the optimal stack filter will decide a 0 and a 1, respectively, while minimizing the overall cost of making an error.

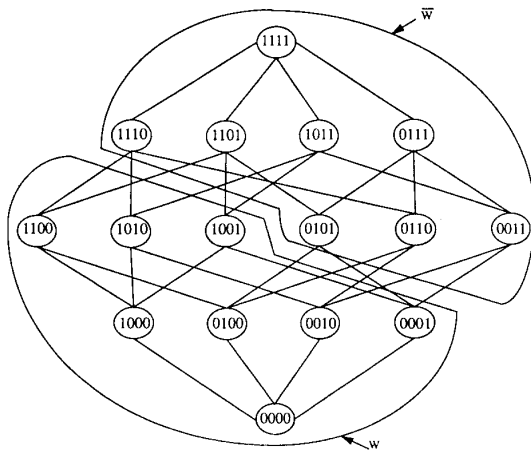


Fig. 2. An instance of an optimal solution.

Note that when the sets W and \bar{W} are separated by a horizontal line in the stacking diagram of Fig. 1, the optimal stack filter is a rank-order filter.

V. CONCLUSIONS AND FURTHER RESEARCH

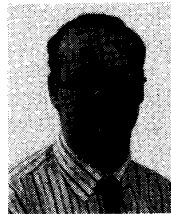
We have presented two approaches to modify the LP problem that finds the optimal stack filter for an MMAE stack filtering problem. The first dealt with reducing the number of constraints at the expense of requiring a 0-1 LP to solve for the optimal point; while the second transformed the problem to a network problem and thereby reduced its time complexity from being doubly exponential in the window width n (in the worst case) to being just exponential.

As for future research, sparsity and special structures in the constraint matrix could still be exploited to further reduce the time required to solve for the optimal point.

Another approach is to use the Dantzig-Wolfe decomposition principle [2] to decompose the constraint matrix and thereby reduce the large problem to a set of smaller and more manageable problems.

REFERENCES

- [1] E. J. Coyle and J.-H. Lin, "Stack filters and the mean absolute error criterion," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp. 1244-1254, Aug. 1988.
- [2] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Amer. Math. Soc. Not.*, Abstr. 553-559, vol. 5, no. 7, p. 811, Dec. 1958.
- [3] M. Gabbouj and E. J. Coyle, "Minimum mean absolute error stack filtering with structural constraints and goals," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 955-968, June 1990.
- [4] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
- [5] A. Shrijiver, *Theory of Linear and Integer Programming*. New York: Wiley, 1986.

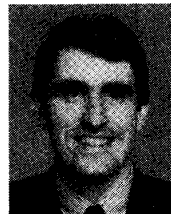


Moncef Gabbouj (S'85-M'90) was born in Monastir, Tunisia, in 1962. He received the B.S. degree in electrical engineering from Oklahoma State University, Stillwater, in 1985, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1986 and 1989.

Since 1990, he has been with the Research Institute of Information Technology, Tampere, Finland, where he is currently a Senior Research Scientist. He also holds a teaching position in the

Signal Processing Laboratory at Tampere University of Technology, Tampere, Finland. His research interests include nonlinear signal and image processing, mathematical morphology, neural networks, and artificial intelligence.

Dr. Gabbouj was a corecipient of the Myril B. Reed Best Paper Award from the 32nd Midwest Symposium on Circuits and Systems. He is currently the Secretary of the Technical Committee on Digital Signal Processing of the IEEE Circuits and Systems Society.



Edward J. Coyle (S'79-M'82) received the Bachelor of Electrical Engineering degree from the University of Delaware in 1978, and the master's and Ph.D. degrees in electrical engineering and computer science from Princeton University in 1980 and 1982, respectively.

Since 1982, he has been with the School of Electrical Engineering at Purdue University, West Lafayette, IN, where he is an Associate Professor. During the 1990-1991 academic year, he was a Visiting Professor of Electrical Engineering at

the University of Delaware. His research interests lie in the areas of digital signal and image processing, and performance analysis of computer communication networks.

Dr. Coyle was a corecipient of the 1986 Best Paper Award for authors under the age of 30 from the IEEE Acoustics, Speech, and Signal Processing Society. He was also a corecipient of the Myril B. Reed Best Paper Award from the 32nd Midwest Symposium on Circuits and Systems. He has served as an Associate editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS and was a member of the technical program committee of ISCAS'91.