01 Jan 2001

Dynamic Resource Allocation for Network Echo Cancellation

T. Gansler

J. Benesty

M. Mohan Sondhi

Steven L. Grant
Missouri University of Science and Technology, sgrant@mst.edu

Recommended Citation

T. Gansler et al., "Dynamic Resource Allocation for Network Echo Cancellation," *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001. ICASSP '01*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2001.
The definitive version is available at https://doi.org/10.1109/ICASSP.2001.940347

DYNAMIC RESOURCE ALLOCATION FOR NETWORK ECHO CANCELLATION

Tomas Gänsler, Jacob Benesty, M. Mohan Sondhi, and Steven L. Gay

Bell Laboratories, Lucent Technologies
600 Mountain Avenue
Murray Hill, New Jersey 07974-0636
{gaensler,jbenesty,mms,slg}@bell-labs.com

ABSTRACT

Network echo canceler chips are designed to handle several channels simultaneously. With the processing speeds now available, a single chip might handle several hundred channels. In current implementations, however, the adaptation algorithm is designed for a single channel, and the computations are replicated N_c times, where N_c is the number of channels. With such an implementation, the computational requirement is N_c times the *peak* load for a single channel. The number of computations required in each channel, however, varies widely over time. Therefore, a considerable reduction in computational load can be achieved by designing the system for the *average* load plus a margin to account for load variations. The reduction in complexity is achieved by exploiting three features: (a) the inherent pauses in conversations, (b) the sparseness of network echo paths, and (c) the fact that an adaptive filter does not need to be updated when the error signal is small. In this paper it is shown that, in principle, such a design can reduce the computational load by a very large factor – perhaps as large as thirty. It remains to be seen whether a customized hardware architecture can be implemented to fully take advantage of the proposed algorithm.

1. INTRODUCTION

Current algorithms for network echo cancelers are designed without regard to the fact that, invariably, a single canceler chip handles many conversations simultaneously. This implies that for N_c channels, the processor must handle N_c times the *peak* computational load of a single channel. If the number of channels is large, however, it should be possible to reduce the demands on the processor to something close to N_c times the *average* load. Some additional computational capacity would, of course, be necessary to take care of statistical fluctuation in the requirements, but the required safety margin becomes smaller as N_c becomes larger. (With the speed and memory now available on a chip, the number of channels can be several hundred, so the safety margin might not have to be large.) Once the problem is looked upon as that of dealing with a large number of channels, it is also possible to take advantage of other knowledge about speech patterns and characteristics of long distance circuits to further reduce the computational load. In this paper, we show how the computational requirement can, in principle, be reduced by a very large factor – perhaps as large as thirty.

Basically we capitalize on three facts. First, during a telephone conversation, there are many pauses in each speech signal. These pauses have been exploited to decrease the idle time in telephone connections, so called TASI (time assignment speech interpolation) networks, since the 1960s [1]. An echo canceler too can take advantage of these pauses. Second, network echo paths are sparse, i.e. only a few coefficients are nonzero. By utilizing this sparseness property of the responses, it is possible to increase the convergence rate and decrease the complexity of adaptive filters [2], [3], [4], [5]. Finally, echo paths do not change much during a conversation. Hence the adaptive filter need not be updated continuously. It is estimated that it needs to be updated perhaps only 10% of the time. These three features can be exploited to design an efficient algorithm.

The paper is organized as follows: In Section 2 the proportionate normalized least mean squares (PNLMS) algorithm is briefly presented. Section 3 gives some proposals on how to simplify the PNLMS algorithm, and shows the reduction in complexity that may be achieved. Performance simulations are presented in Section 4, and Section 5 gives a discussion of the results and problems associated with this approach.

2. THE PNLMS ALGORITHM

In this section, we give a brief description of the PNLMS algorithm; for more details see [6, 7]. In derivations and descriptions, the following notation is used:

$$\begin{aligned}
x(n) &= \text{Far-end signal,}\\
y(n) &= \text{Echo and background noise possibly including near-end signal,}\\
\mathbf{x}(n) &= [x(n)\ \cdots\ x(n-L+1)]^T, \text{ Excitation vector,}\\
\mathbf{h} &= [h_0\ \cdots\ h_{L-1}]^T, \text{ True echo path,}\\
\hat{\mathbf{h}}(n) &= [\hat{h}_0(n)\ \cdots\ \hat{h}_{L-1}(n)]^T, \text{ Estimated echo path.}
\end{aligned}$$

Here L is the length of the adaptive filter, and n is the time index.

The PNLMS algorithm was proposed in [6]. In this algorithm, an adaptive individual step-size is assigned to each filter coefficient. The step-sizes are calculated from the last estimate of the filter coefficients in such a way that a larger coefficient receives a larger increment, thus increasing the convergence rate of that coefficient. This has the effect that active coefficients are adjusted faster than non-active coefficients (i.e. small or zero coefficients). Hence, PNLMS converges much faster than NLMS for sparse impulse responses (i.e., responses in which only a small percentage of coefficients is significant). Most impulse responses in the telephone network have this characteristic.

0-7803-7041-4/01/$10.00 ©2001 IEEE

The PNLMS algorithm is described by the following equations:

$$e(n) = y(n) - \hat{\mathbf{h}}^T(n-1)\mathbf{x}(n), \quad (1)$$

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \frac{\mu \mathbf{G}(n)\mathbf{x}(n)}{\mathbf{x}^T(n)\mathbf{G}(n)\mathbf{x}(n) + \delta} e(n), \quad (2)$$

$$\mathbf{G}(n) = \mathrm{diag}\{g_0(n), \ldots, g_{L-1}(n)\}. \quad (3)$$

$\mathbf{G}(n)$ is a diagonal matrix which adjusts the step-sizes of the individual taps of the filter, μ is the overall step-size parameter, and δ is a regularization parameter which prevents division by zero and stabilizes the solution when speech is used as the input (far-end) signal. The diagonal elements of $\mathbf{G}(n)$ are calculated as follows [6]:

$$\gamma_l(n+1) = \max\{\rho \max\{\delta_p, |\hat{h}_0(n)|, \ldots, |\hat{h}_{L-1}(n)|\}, |\hat{h}_l(n)|\}, \quad (4)$$
$$0 \le l \le L-1,$$

$$g_l(n+1) = L\gamma_l(n+1) / \sum_{i=0}^{L-1} \gamma_i(n+1), \quad (5)$$
$$0 \le l \le L-1.$$

Parameters δ_p and ρ are positive numbers with typical values $\delta_p = 0.01$, $\rho = 5/L$. ρ prevents coefficients from stalling when they are much smaller than the largest coefficient and δ_p regularizes the updating when *all* coefficients are zero at initialization.

3. PROPOSALS FOR REDUCED COMPLEXITY

In this section, we outline the principles of a simplified algorithm that exploits properties of network echo cancellation described in the previous section. We also present the theoretical complexity gains one can achieve. It is assumed that one or more computation engines serve the channels and some logic has been designed to control and distribute the resources. Decisions, e.g., which channels should be updated, are based on results from voice activity and double-talk detection.

An algorithm that takes advantage of the sparseness of the impulse response to improve convergence rate is the PNLMS algorithm [7]. However, its complexity is greater by a factor of 2 compared to that of the standard NLMS algorithm. In the following, we show that we can do much better than these algorithms from a complexity point of view.

3.1. A simple algorithm to update only active channels and coefficients

In a two-way conversation each talker is active only about half of the time; additionally, there are pauses between sentences and syllables. During these inactive time slots no coefficient updating is needed. Furthermore, since network echo path responses are sparse, we can focus computations on only the active (non-zero) coefficients. The following algorithm saves a large number of multiplications at the expense of some additional overhead compared to current implementations of NLMS and PNLMS. The key features are: (1) No coefficient is updated if the channel is inactive or double-talk has been detected. (2) No coefficient is updated if the residual error is sufficiently small. (3) Step-sizes for the active taps can all be made equal instead of the step-sizes specified by the matrix $\mathbf{G}(n)$. (4) All coefficients are updated (i.e., an NLMS iteration is made), every Mth iteration. Only the active coefficients are updated at all other iterations. (5) The index of active coefficients is updated every Mth iteration.

Periods of inactivity are easily identified with a look-ahead of one, or a few, samples at the outputs of the voice activity and double-talk detectors. Hence the first item is easily implemented. The implementation of item 2 is explained in the next subsection. Items 3–5 are implemented as follows:

Let us first define an "active set," i.e., the set of active tap weights. To this end, define a threshold T, and sort the tap weights in descending order of absolute value. Then define the active set A_s as the first L_a weights in this list, such that their cumulative magnitude just exceeds T times the cumulative magnitude of *all* the L taps of the filter. In symbols

$$A_s = \{l : \min_{l \in A_s}\{|\hat{h}_l(n)|\} > \max_{l \notin A_s}\{|\hat{h}_l(n)|\}, \; T\sum_{l=1}^{L-1}|\hat{h}_l(n)| \le \sum_{l \in A_s}|\hat{h}_l(n)| \le T\sum_{l=1}^{L-1}|\hat{h}_{l,n}| + \max_{k \notin A_s}\{|\hat{h}_k(n)|\}\}. \quad (6)$$

The threshold T is selected in the range $T = [0.9, 1)$. From an implementation point of view it may be appropriate to limit the maximum size of the set to L_{max} thus $L_a \le L_{max} < L$. The maximum load of a channel can thereby be limited. For $M-1$ consecutive iterations the coefficients of the active set are updated as follows:

$$e(n) = y(n) - \sum_{l \in A_s} \hat{h}_l(n-1)x(n-l), \quad (7)$$

$$\hat{h}_l(n) = \hat{h}_l(n-1) + \frac{\mu x(n-l)}{\sum_{l \in A_s} x^2(n-l) + \delta} e(n), \; l \in A_s. \quad (8)$$

Every Mth iteration a full NLMS iteration is made, i.e.,

$$e(n) = y(n) - \hat{\mathbf{h}}^T(n-1)\mathbf{x}(n), \quad (9)$$

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \frac{\mu \mathbf{x}(n)}{\mathbf{x}^T(n)\mathbf{x}(n) + \delta} e(n). \quad (10)$$

Increasing M will reduce the average complexity but also worsen tracking performance. Moreover, with $T = 1$ the active set covers all the taps, and we get the standard NLMS algorithm. In the simulations described in Section 4 we used $M = 10$.

3.2. Stopping adaptation when small residual error is detected

Echo paths on the network vary slowly, in general. Hence adaptation is needed only at a small percentage of iterations, perhaps no more than 10%. This would yield a huge reduction in computations, since on a vast majority of iterations we need to compute only the convolution with the small number (L_a) of coefficients in the set A_s. Therefore we propose that when the error signal is sufficiently small, we do not update or sort the tap weights. Asymptotically, for network echo cancelers, the complexity of this algorithm would thus be reduced essentially to the computation of a convolution on the active taps only.

A good decision variable, to decide if the residual error is small enough, is the normalized mean square error (in dB) defined as follows:

$$\xi_r(n) = 10\log_{10}\left(\frac{< e^2(n) >_{N_r}}{< y^2(n) >_{N_r} + \delta_y}\right), \quad (11)$$

where

$$< e^2(n) >_{N_r} = \frac{1}{N_r}\sum_{n-N_r+1}^{n} e^2(n) \quad (12)$$

is the mean square error and $< y^2(n) >_{N_r}$ is analogously defined. The regularization parameter δ_y prevents division by zero during silences between words, and N_r is the length of the window used to estimate energy. N_r should preferably be chosen small in order not to degrade the tracking performance of the adaptive algorithm when the echo path changes.

The adaptive algorithm proceeds as follows: at each iteration n, (11) is computed and $\xi_r(n)$ is compared to a threshold T_r (a typical range is $T_r = [-40, -30]$). The decision rule is simple: if $\xi_r(n) \geq T_r$, then the residual error is not considered small enough and the algorithm continues to update; If $\xi_r(n) < T_r$, then the residual error is considered negligible and the algorithm neither updates nor sorts the taps of the filter. However, the convolution on the active taps is still performed.

3.3. Theoretical reduction in complexity

The complexity of the proposed algorithm is compared to a PNLMS implementation with respect to multiplications and other required computations. Computations required for the various steps of the proposed algorithm are: Eq. (6): $k_0 L\log_2(L) + 2L + L_a$ (sorting instructions) [1], Eq. (7): L_a (multiplications), Eq. (8): $2L_a$ (multiplications), Eq. (9): L (multiplications), Eq. (10): $2L$ (multiplications), where k_0 is a proportionality constant for the sorting algorithm. Let the probability of active speech be denoted by p_s and the probability of active adaptation by p_a. Then, assuming equal weight for multiplications and sorting instructions, the average required number of computations is

$$C_0 = p_s p_a \frac{3L}{M}, \text{ full NLMS update,} \quad (13)$$

$$C_1 = p_s \frac{(M-1)(1+2p_a)L_a}{M}, \quad (14)$$
update of active coefficients,

$$C_2 = p_s p_a \frac{L + L_a + k_0 L\log_2(L)}{M}, \quad (15)$$
update of the active set,

and the average total number of computations is

$$C_{p.alg} = C_0 + C_1 + C_2. \quad (16)$$

For comparison, note that the implementation of NLMS and PNLMS requires (with equal weight for multiplications and comparisons):

$$C_{NLMS} = 2L + 1, \quad (17)$$

$$C_{PNLMS} = 6L. \quad (18)$$

[1] This assumes that an algorithm like *quicksort* is used.

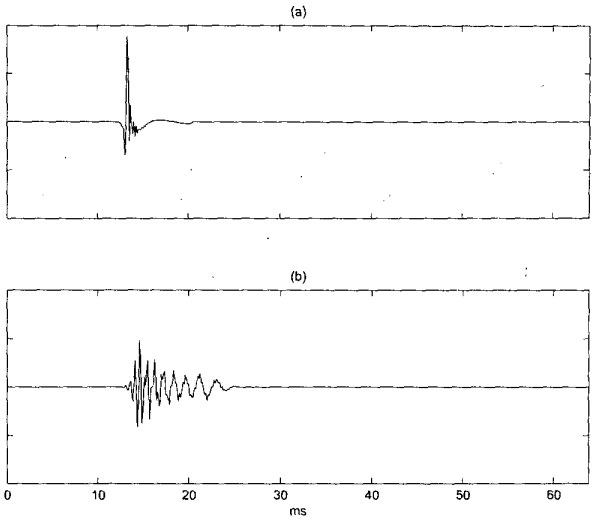

Fig. 1. Impulse responses of the two hybrids used in our simulations.

As an illustration, assuming the expected typical values: $p_s = 0.5$, $p_a = 0.1$, $L_a = 100$, $k_0 = 1$, $M = 10$, $L = 768$, we find that

$$C_{p.alg} \approx 70, \quad (19)$$

$$C_{NLMS} = 1537, \quad (20)$$

$$C_{PNLMS} = 4608, \quad (21)$$

which shows that the average complexity of the proposed algorithm could be drastically less than that of NLMS or PNLMS. However, for various reasons, the estimate in (19) should not be taken literally. First, it does not allow for a safety margin, and it is not yet clear how large that needs to be made. Second, at present there is no hard evidence to justify the estimate $p_a \approx 0.1$. Third, the estimate of complexity of the sorting algorithm is not rigorous. Finally, the choice of $M = 10$, that controls the initial convergence rate has not yet been optimized. Nevertheless, a reduction in complexity by a factor as large as 20 or 30 appears to be possible.

4. SIMULATIONS OF THE PROPOSED ALGORITHM

In this section we compare the performance, in terms of convergence rate and tracking, of our proposed algorithm vs. NLMS and PNLMS. We also compare the difference in performance for the proposed algorithm when the adaptation is halted according to the criterion in Section 3.2. Figure 1 shows impulse responses of the hybrids used in our simulations. These represent two generic types of responses that can be expected in practice. Speech is used as excitation signal.

The performance of the algorithms is evaluated by using the misalignment (MIS) which is given by,

$$\text{MIS}(n) = \|\hat{\mathbf{h}}(n) - \mathbf{h}\|/\|\mathbf{h}\|, \quad (22)$$

where $\mathbf{h}$ is the impulse response of the true echo path.

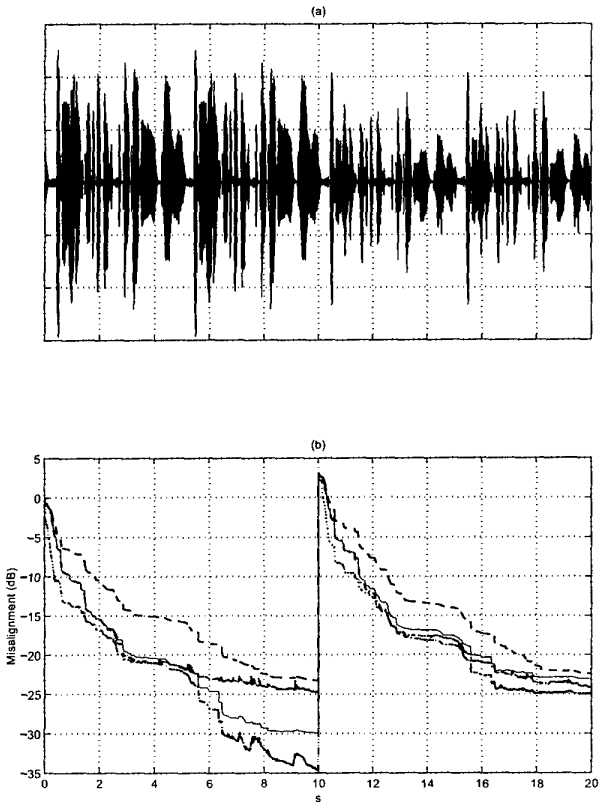

Fig. 2. (a) Echo and background noise, $y(n)$. (b) Behavior of the misalignment of the proposed algorithm, with adaptation stopped according to decision variable (11) (solid), and without stopping adaptation (dash-dotted), NLMS (dash), and PNLMS (dotted). The echo path changes at time 10 s from the one in Fig. 1a to the one in Fig. 1b.

All algorithms are tuned to achieve approximately the same minimum mean square error in order to fairly compare convergence rate.

The following parameters are used: $\mu = 0.2$, $L = 512$ (64 ms), $\delta = 4 \cdot 10^6$. $T_{\mathrm{r}} = -38.5$ dB, $N_{\mathrm{r}} = 40$, $\delta_y = 1 \cdot 10^6$. $T = 0.98$, $L_{\max} = 200$. $\sigma_x = 1900$, SNR $\approx$ 39 dB. Hybrid attenuation: 6 dB. $\mathbf{h}(-1) = \mathbf{0}$.

Figure 2 shows the misalignment for three different algorithms when the input is speech. In order to study tracking of the algorithm, i.e., how the algorithm behaves when the echo path changes, we swap echo path at 10 seconds from that of Fig. 1a to the one in Fig. 1b. We find for both initial convergence and tracking, the proposed algorithm is considerably faster than NLMS but somewhat slower than PNLMS. With longer echo paths, the performance improvement compared to NLMS will be even greater.

Also seen in Fig. 2, is that when we compare the proposed algorithm without stopping adaptation during periods of small residual error (i.e. with T_{r} set to $-\infty$) with the same algorithm when adaptation is indeed stopped, i.e. T_{r} selected such that the adaptation was halted more than 45% of the time, there is not a great difference in performance. We can see that the proposed algorithm (whether or not adaptation is stopped) outperforms the NLMS algorithm.

5. DISCUSSION

In this paper, a number of proposals have been made for decreasing the complexity of the adaptive algorithm in a multi echo canceler system. Emphasis has been placed on finding simple procedures for choosing active regions of the impulse response and halting adaptation when the residual error is small. Though more careful analysis and development of the algorithms are needed, these proposals give some idea of what can be done from an algorithm point of view in order to improve the efficiency of the implementations.

An important aspect of the problem that we have not discussed here is the possibility of reducing the requirement of storage capacity. Reduction in storage requirements is necessary if the reduction of computational complexity is to be fully exploited.

One possibility for reducing storage requirements is to store the L_{a} coefficients in the set A_{s} with full precision, and the rest with reduced precision. Another possibility is to store $L_{\max}$ coefficients with full precision and the rest with reduced precision. Since the inactive coefficients are, in general, much larger in number, this procedure can significantly reduce the memory requirement. What remains to be seen is how few bits can be used for the inactive taps without degrading performance.

6. REFERENCES

[1] K. Bullington and J. M. Fraser, "Engineering aspects of TASI," *Bell Syst. Tech. J.*, pp. 353–364, Mar. 1959.

[2] S. Kawamura and M. Hatori, "A tap selection algorithm for adaptive filters," in *Proc. IEEE ICASSP, 1986*, pp. 2979–2982.

[3] V. Madisetti, D. Messerschmitt, and N. Nordström, "Dynamically reduced complexity implementation of echo cancelers," in *Proc. IEEE ICASSP*, 1986, pp. 1313–1316.

[4] A. Sugiyama et al., "A fast convergence algorithm for adaptive FIR filters under computational constraint for adaptive tap-position control," *IEEE Trans. Circuits Syst. II*, vol. 43, pp. 629–636, Sept. 1996.

[5] J. Homer, I. Mareels, R. R. Bitmead, B. Wahlberg, and A. Gustafsson, "LMS estimation via structural detection," *IEEE Trans. Signal Processing*, vol. 46, pp. 2651 –2663, Oct. 1998.

[6] D. L. Duttweiler, "Proportionate normalized least mean squares adaptation in echo cancelers," *IEEE Trans. Speech Audio Processing*, vol. 8, pp. 508–518, Sept. 2000.

[7] T. Gänsler, S. L. Gay, M. M. Sondhi, and J. Benesty, "Double-talk robust fast converging algorithms for network echo cancellation," *IEEE Trans. Speech Audio Processing*, Nov. 2000.