Local Search for COM Run Time Optimization

Hansel D'Silva
Amphenol Corporation

IEEE 802.dj joint optical/ electrical/ logic ad hoc

History

Revision	Date	Comment
0.5	19-Jun-26	▪ Initial Draft.

List of Supporters

- Adrien Auge, Qualcomm
- Todd Bermensolo, Independent
- Howard Heck, TE Connectivity

List of Contributors

- Adam Gregory, Samtec
- Richard Mellitz, Samtec

1. Exhaustive and Local Search in the COM tool

J	K	L
Operational		
ERL Pass threshold	11	dB
COM Pass threshold	3	db
DER_0	2.00E-04	
T_r	0.00400	ns
FORCE_TR	1	logical
PMD_type	C2C	for MMSE use C2C only
EW	1	
MLSE	1	logical
ts_anchor	1	
sample_adjustment	[-24 24]	
Local Search	0	
flim	6.70E+10	Hz
zero_pad	1	logical

- If “Local Search” is zero, then Exhaustive Search else greater than one then Local Search.

2. Problem Statement

- Awareness of Local Search in the COM tool in reducing run time.
- With TXLE sweep being enabled, Exhaustive Brute Search takes as much as 4 days while Local Search takes as much as 10 hours.
- How can COM runtime be reduced using Local Search while maintaining accuracy and robustness?
  - Local search is not part of the standard; rather, it is used to make runtime more manageable (less than 15 minutes), particularly when handling the 9.227043 Million equalization combinations generated while sweeping the TXLE.

3. Summary

- Local Search is a useful tool in pruning the equalization search space.
- Adaptive Local Search appears to reduce the run time significantly with nearly negligible loss in accuracy for the value of COM.
  - Compared to the current Local Search, the proposed Adaptive Local Search dynamically adjusts the search radius based on FOM convergence and uses weighted multi-dimensional distance metrics, enabling more aggressive pruning while maintaining focus on the most promising equalization regions.

4. FOM and COM

93A.1.6 Determination of variable equalizer parameters

COM is a function of the variables $c(-2)$, $c(-1)$, $c(1)$, g_{DC}, and g_{DC2}. The following procedure is used to determine the values of these variables that are used to calculate COM.

a) Compute the pulse response $h^{(k)}(t)$ of each signal path k for a given $c(-2)$, $c(-1)$, $c(1)$, g_{DC}, and g_{DC2} using the procedure defined in 93A.1.5.

. . .

$$FOM = 10\log_{10}\left(\frac{A_s^2}{\sigma_{TX}^2 + \sigma_{ISI}^2 + \sigma_J^2 + \sigma_{XT}^2 + \sigma_N^2}\right) \quad \text{(93A–36)}$$

The FOM is calculated for each permitted combination of $c(-2)$, $c(-1)$, $c(1)$, g_{DC}, and g_{DC2} values per Table 93A–1, where any parameters not provided by the clause that invokes this method are set to 0. The combination of values that maximizes the FOM, including the corresponding value of t_s, is used for the calculation of the interference and noise amplitude in 93A.1.7 and the calculation of COM in 93A.1.

- When calculating Channel Operating Margin (COM), there is a callout to Figure Of Merit (FOM) for the determination of the variable equalizer parameters.
- Below is a summary.

  1] FOM is based on an RMS-type metric, which averages error across the waveform.

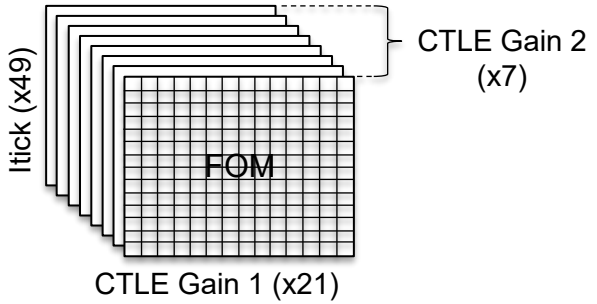
  2] COM is based on a CDF/ statistical tail behavior, meaning it is more sensitive to worst-case or tail events rather than the average.

5. Equalization Search Space

Parameter	Definition	Values	Count
c(-2)	TX equalizer, coeff. -2	[0:0.02:0.14]	8
c(-1)	TX equalizer, coeff. -1	[-0.34:0.02:0]	18
c(0)	TX equalizer, minimum coeff. 0	0.54	N.A.
c(1)	TX equalizer, coeff 1	[-0.2:0.02:0]	11
ctle_index	CTLE Gain 1	[-20:1:0]	21
g_LP_index	CTLE Gain 2	[-6:1:0]	7
itickn	Sampling clock offset	[-24:1:24]	49

```
% Skip combinations with small values of c(0), not guaranteed to be supported by all transmitters.
if txffe_cur<param.tx_ffe_c0_min
    continue;
end
```

TXLE combinations= (8*18*11)- 303= 1281



= 49x21x7
= 7203 combinations without TXLE

= 1281x49x21x7
= 9.227043e6 combination with TXLE
= 9.227043 Million combinations with TXLE

- With TXLE sweep being OFF there are **7,203 combinations**.
- With TXLE sweep being ON there are **9.227043 Million combination**.

6. Results on Run Time and Efficiency

IL_db_die_to_die_at_Fnq = 37.10 dB

0] Run Time

TXLE Sweep	COM [dB]	FOM [dB]	TXLE	Run Time		
				Exhaustive (Brute force)	Existing Local Search= 2	Proposed Local Search= 2
OFF	2.31	11.607	[0 0 1 0]	11.51 minutes	6.92 minutes	0.69 minutes
ON	2.50	11.691	[0 -0.1 0.9 0]	85.04 hours	7.56 hours	7.10 minutes

1] Number of FOM calculations

TXLE Sweep	COM [dB]	FOM [dB]	TXLE	Number of FOM calculations		
				Exhaustive (Brute force)	Existing Local Search= 2	Proposed Local Search= 2
OFF	2.31	11.607	[0 0 1 0]	7,203	5,243	980
ON	2.50	11.691	[0 -0.1 0.9 0]	9.227043e6	0.854609e6	11,907

- The proposed local search reduces the run time significantly.
- Please refer the presentation titled “Commit Request 4p15_2: Adaptive Local Search with Deterministic Zoom-in” for results involving 684 cases of KR and CR channels from the IEEE 802.3dj Public Area.

7. Equalization Search Space Pruning

```
for ctle_index = 1:length(param.ctle_gdc_values)
  %% CTLE Gain
  THIS.ctle_index = ctle_index;
  THIS.g_dc = param.ctle_gdc_values(ctle_index);

  for g_LP_index = 1:length(param.g_DC_HP_values)
    %% Apply CTLE to impulse response
    THIS.g_DC_low = param.g_DC_HP_values(g_LP_index);
    THIS.g_LP_index = g_LP_index;

    % Apply CTLE
    [chdata, THIS.H_ctf, H_low_xc, H_ctf2] = ...
      OptFom_Compute_CTLE(chdata, ctle_gain, THIS, SETTINGS.f_xc, param, OP);

    for TK = 1: size(txffe_matrix,1)
      % Skip combinations with small values of c(0)
      % not guaranteed to be supported by all transmitters.
      txffe_cur = txffe_cursor_vector(TK);
      if txffe_cur < param.tx_ffe_c0_min
        continue;
      end

      THIS.tx_index_vector = FULL_tx_index_vector(TK, :);

      if param.LOCAL_SEARCH > 0 && ~isinf(BEST.FOM)
        skip_it = OptFom_Local_Search(param.LOCAL_SEARCH, BEST, THIS, txffe_sweep_indices);
        if skip_it
          continue;
        end
      end
```

Pruning (skip logic)

```
      %% TXFFE
      % fetch txffe for this iteration
      THIS.txffe = txffe_matrix(TK, :);
      [sbr, chdata, pulse_struc] = .OptFom_Compute_TXFFE(chdata, pulse_struc, THIS.txffe, ...
                  ctle_response_updated, param, OP);

      for itickn = 1:length(full_sample_range)

        THIS.itick = full_sample_range(itickn);

        % Calculate FOM
        % (your FOM calculation logic here)

        if (THIS.FOM > BEST.FOM)
          BEST.txffe_index = THIS.tx_index_vector;
          BEST.ctle        = THIS.ctle_index;
          BEST.gdc         = THIS.g_dc;
          BEST.G_high_pass = THIS.g_LP_index;
          BEST.FOM         = THIS.FOM;
        end

      end % itickn
    end % TK
  end % g_LP_index
end % ctle_index
```

Calculate FOM

Tracking BEST versus THIS

- The equalization optimization performs a nested search across CTLE Gain 1, CTLE Gain 2, TXFFE, and sampling phase (itick).
- Local search prunes the equalization space by skipping CTLE and TXFFE configurations that are farther than a user-defined distance from the current best equalization setting.
- Sampling phase adjustment (itick) is evaluated through an exhaustive sweep, whereas RxFFE coefficients are determined using MMSE-based optimization when MMSE optimization is enabled.

8. Use of Indices for TXLE, CTLE Gain 1 and CTLE Gain 2

```
param.tx_ffe_c0_min= 0.54
param.tx_ffe_cm1_values= [-0.34:0.02:0.00] % [1x18]
param.tx_ffe_cm2_values= [0.00:0.02:0.14] % [1x8]
param.tx_ffe_cm3_values= 0
param.tx_ffe_cm4_values= 0
param.tx_ffe_cp1_values= [-0.20:0.02:0.00] % [1x 11]

param.ctle_gdc_values= [-20:1:0]
param.g_DC_HP_values= [-6:+1:0]

full_sample_range= [-24:1:24]

BEST.FOM = -inf;
```

Parameters to be initialized

```
%% Build txffe values dynamically
[txffe_matrix, cur, txffe_sweep_indices, FULL_tx_index_vector, txffe_cursor_vector]
= OptFom_Build_TXFFE(param);
num_txffe_runs = size(txffe_matrix,1);

%% if LOCAL_SEARCH> 0
if param.LOCAL_SEARCH> 0
   FOM_history = [];
   iter_count = 0;
end
```

1. for TK = 1:size(txffe_matrix,1)

FULL_tx_index_vector=

cm4 cm3 cm2 cm1 cp1

1584x5 double

	1	2	3	4	5
1	1	1	1	1	1
2	1	1	1	1	2
3	1	1	1	1	3
...					
1583	1	1	8	18	10
1584	1	1	8	18	11

2. for ctle_index =1:length(param.ctle_gdc_values)

3. for g_LP_index = 1:length(param.g_DC_HP_values)

- The Local Search uses indices for the TXLE, CTLE Gain 1 and CTLE Gain 2.
- It is important that the matrix columns are ordered as: C-4, C-3, C-2, C-1, C+1, C+2, with the main cursor (C0) intentionally excluded.

9. Current Local Search in the COM tool

```
function skip_it = OptFom_Local_Search(LocalSearch_Value, BEST, THIS, txffe_sweep_indices)

best_txffe_index = BEST.txffe_index;
best_G_high_pass = BEST.G_high_pass;
tx_index_vector = THIS.tx_index_vector;
ctle_index = THIS.ctle_index;
g_LP_index = THIS.g_LP_index;
num_txffe_sweep_indices = length(txffe_sweep_indices);

skip_it=0;

%instead of looping across all taps, only loop across those with length>1 (txffe_sweep_indices)
%It saves time since this block is encountered so often
for kj=1:num_txffe_sweep_indices
   kv=txffe_sweep_indices(kj);
   if kv==1
      previous_loop_val=g_LP_index;
   else
      previous_loop_val=tx_index_vector(kv-1);
   end
   if previous_loop_val>1
      best_index_this_tap=best_txffe_index(kv);
      if abs(tx_index_vector(kv)-best_index_this_tap)>LocalSearch_Value
         skip_it=1;
         break;
      end
   end
end

if ~skip_it && ctle_index>1 && abs(g_LP_index-best_G_high_pass)>LocalSearch_Value
   skip_it=1;
end
```

- For example,
  tx_index_vector = [5 4 3 2 1 ...]
  best_txffe_index = [5 5 3 2 1 ...]
  Hence, distance= abs(current_index - best_index).

- Do not allow LP/CTLE to drift too far from best.
  if abs(g_LP_index - best_G_high_pass) > LocalSearch_Value
  skip_it = 1;

- Authored by Adee Ran and improved by Adam Gregory in making it modular for n-tap TXFFE.
- For each TXFFE tap and CTLE Gain 1 & CTLE Gain 2 index, the algorithm compares the current candidate index to the best-known index. If the difference exceeds a predefined threshold (LocalSearch_Value, let us say 2) in any dimension, the candidate is skipped without evaluation.

10. Proposed Adaptive Local Search

```
%% ------------------------------------------------------
%  Extract tap vectors
%% ------------------------------------------------------
best_taps = BEST.txffe_index(:);
curr_taps = THIS.tx_index_vector(:);

ctle_index = THIS.ctle_index;
lp_curr    = THIS.g_LP_index;
lp_best    = BEST.G_high_pass;

%% ------------------------------------------------------
%  Build weighted vectors
%% ------------------------------------------------------
best_vec = [best_taps; lp_best];
this_vec = [curr_taps; lp_curr];

num_taps = numel(curr_taps);
w_taps = ones(num_taps,1) * edge_weight;

% LP weight (patched)
if THIS.ctle_index > 1
   w_lp = lp_weight * (1 + 0.5 * (THIS.ctle_index - 1));
else
   w_lp = lp_weight;
end
weights = [w_taps; w_lp];
```

```
%% ------------------------------------------------------
%  Compute distances
%% ------------------------------------------------------
diff_vec      = this_vec - best_vec;
weighted_diff = weights .* diff_vec;

L1_w = sum(abs(weighted_diff));
L2_w = sqrt(sum(weighted_diff.^2));

raw_L1_TX = sum(abs(diff_vec(1:num_taps)));
```

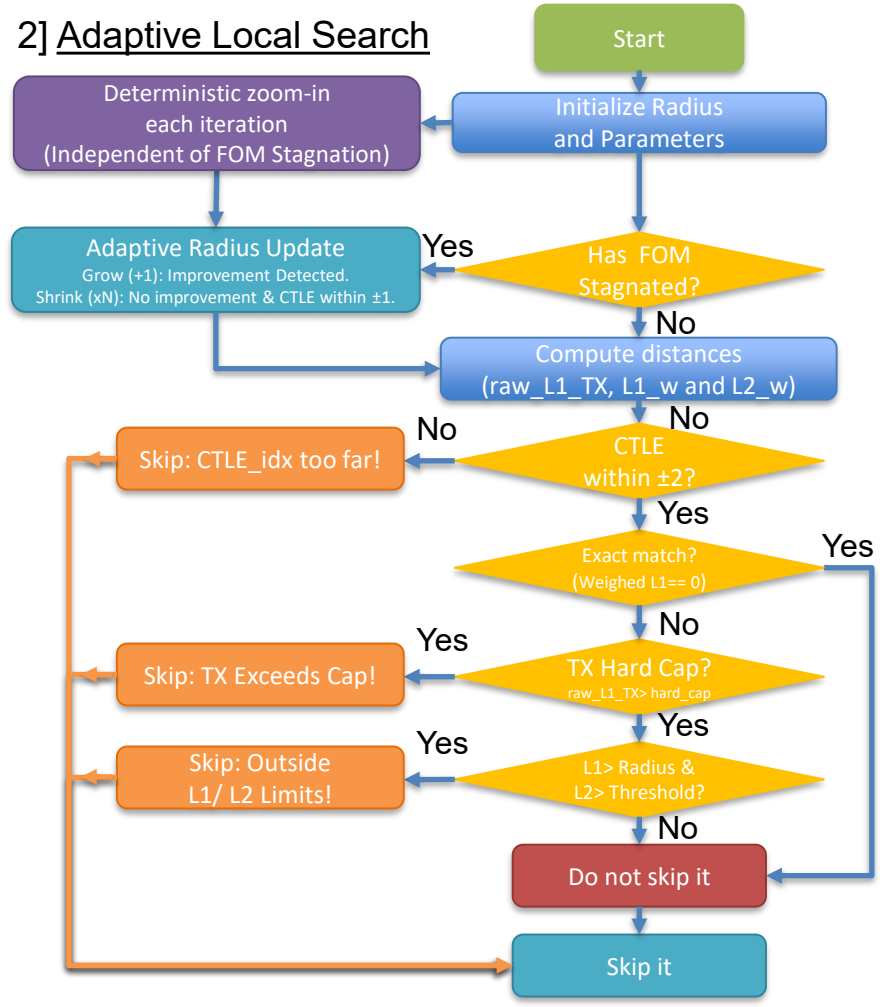
Definitions,
raw_L1_TX: Unweighted Manhattan distance between the current and best TXFFE tap-index vectors.

L1_w: Weighted Manhattan distance between the current equalization setting and the current best equalization setting, including TXFFE tap indices and CTLE Gain 2.

L2_w: Weighted Euclidean distance between the current equalization setting and the current best equalization setting, providing a geometric measure of proximity in the equalization search space.

- Instead of checking each parameter independently, the adaptive local search computes global weighted L1 and L2 distances between the current candidate and the best-known solution.
- It maintains an adaptive search radius that shrinks when improvement stalls and expands when progress is observed.
- If the candidate exceeds this radius, violates a hard cap on total deviation, or fails additional constraints (e.g., CTLE proximity), it is skipped without evaluation.

11. Rough Flowchart of Local Search



Definitions,
raw_L1_TX: Unweighted Manhattan distance between the current and best TXFFE tap-index vectors.

L1_w: Weighted Manhattan distance between the current equalization setting and the current best equalization setting, including TXFFE tap indices and CTLE Gain 2.

L2_w: Weighted Euclidean distance between the current equalization setting and the current best equalization setting, providing a geometric measure of proximity in the equalization search space.

- Compared to the current Local Search, the proposed Adaptive Local Search dynamically adjusts the search radius based on FOM convergence and uses weighted multi-dimensional distance metrics, enabling more aggressive pruning while maintaining focus on the most promising equalization regions.

Thank you