A COMPUTATIONALLY CONSTRAINED OPTIMIZATION FRAMEWORK FOR IMPLEMENTATION AND TUNING OF SPEECH ENHANCEMENT SYSTEMS

Daniele Giacobello, Jason Wung, Ramin Pichevar, Joshua Atkins

Beats Electronics, LLC, 8600 Hayden Place, Culver City, CA 90232
{Daniele.Giacobello, Jason.Wung, Ramin.Pichevar, Josh.Atkins}@beatsbydre.com

ABSTRACT

In this work, we propose an optimization framework for tuning the parameters of a speech enhancement system to maximize its performance while constraining its computational complexity imposed by a target platform. Some parameters allow for enabling or disabling certain algorithmic components of the system, effectively guiding the implementation effort. The speech enhancement system is deployed in a speech recognition front-end and in a full-duplex telephony system. The optimization variables are the parameters of the system and the performance is measured using phone accuracy rate and mean opinion score, respectively. The problem is then a nonlinear program of combinatorial nature which is solved efficiently using a genetic algorithm. The results show improvement in performance over common tuning and implementation strategies.

1. INTRODUCTION

Speech enhancement (SE) algorithms are fundamental to most speech-centric applications due to a plethora of acoustical disturbances that degrade the captured speech signals [1]. The research and development effort in designing SE systems aims at integrating different algorithms and maximizing the performance using objective measures [2]. When the SE systems are used in full-duplex speech communications, the objective is to maximize the perceptual quality using the mean opinion score (MOS) [3], which can be calculated using automated techniques that mimic the human hearing process [4]. The current ITU-T standardized model is the Perceptual Objective Listening Quality Assessment (POLQA) [5], which produces reliable scores for evaluating SE algorithms and overcomes several limitations of its predecessor, the Perceptual Evaluation of Speech Quality (PESQ) [6]. When SE systems are used as a preprocessor for automatic speech recognition (ASR), the objective of the algorithmic design is to maximize the speech recognition accuracy [7]. While model-domain enhancement methods have been shown to better account for the mismatch between the training condition and the application scenario [8], methods relying on fixed acoustic models using the hidden Markov models (HMMs) are still the most common methods for limited-vocabulary recognition on embedded systems [9]. Therefore, these methods rely heavily on the SE algorithms to enhance the speech signals before feature extraction to match the training condition of the ASR [10]. Accurate ways to assess ASR reliability are still a matter of debate since they are heavily application and context dependent [11]. However, for embedded systems, the phone accuracy rate (PAR), or at a higher semantic level the word accuracy rate (WAR), is generally appropriate as a performance measure for the ASR.

During development and prototyping, a commercially viable SE system must take into account the constraints of the target platform [12]. For audio related applications, field-programmable gate arrays (FPGAs) [13] and dedicated digital signal processors (DSPs) are the most common choices since they generally have lower cost, lower latency, and lower energy consumption [14]. However, meeting the computational budget of the target hardware, commonly measured in terms of million cycles per second (MCPS), is generally a non-negotiable condition [15]. The computational complexity of an algorithm is calculated by counting the number of basic mathematical operations, e.g., multiplications, additions, or multiply-accumulations (MACs), as well as the usage of pre-defined, highly-optimized subroutines already embedded in the processor, e.g., the fast Fourier transforms (FFTs) [16].

The objective of maximizing the perceptual quality or the speech recognition accuracy often contradicts the computational constraints imposed by the target platform. While *profiling* each component of a SE system during development is a good practice to avoid overly complex solutions, the *tuning* of the system is often done at an advanced stage of the development and may influence the computational complexity dramatically. Furthermore, the optimization often relies on measures that are easier to handle mathematically, e.g., the mean-squared error (MSE) or the log-spectral distortion (LSD) [2], but may not relate well to the actual goal of the system, i.e., maximizing the perceptual quality or the speech recognition accuracy. In our recent works [17, 18], we formalized the tuning of a SE system for full-duplex communications by casting it as an optimization problem, where the objective function was a perceptual objective measure and the optimization variables were its parameters. The work was then extended to the optimization of a ASR front-end [19], where the objective function was the back-end recognizer accuracy. Similar ideas were used in [20] and in [21], to tune the parameters of a noise reduction system and the parameters of a ASR back-end, respectively.

In previous works, however, the optimization problem was unconstrained. Thus any solution satisfying the maximization of the perceptual objective quality or recognition accuracy could be the solution to our problem. In this work, a nonlinear penalty function accounting for the computational complexity is introduced in the optimization framework. The system to be optimized is comprised of several algorithmic blocks and two large databases of conversational speech, derived from the TIMIT database [22], that cover a wide range of scenarios which are used for training and testing. The system is then optimized for either full-duplex communications or an ASR front-end with the computational complexity constraint specified in terms of MCPS.

2. SPEECH ENHANCEMENT ALGORITHM

Let $y[n]$ be the near-end microphone signal, which consists of the near-end speech $s[n]$ and noise $v[n]$ mixed with the acoustic echo

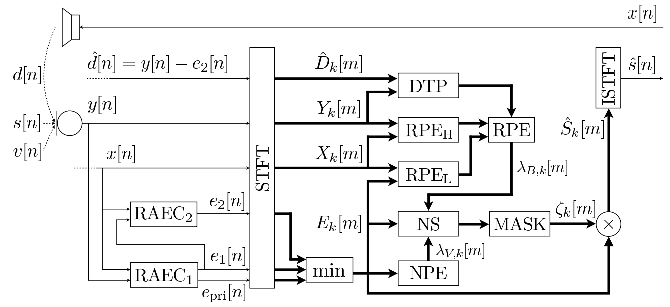

Fig. 1. A block diagram of the speech enhancement system.

$d[n] = h[n] * x[n]$, where $h[n]$ is the impulse response of the system, $x[n]$ is the far-end reference signal, and $*$ is the convolution operator. The overall block diagram of the speech enhancement algorithm is shown in Figure 1, which consists of two robust acoustic echo cancelers (RAECs), a double-talk probability (DTP) estimator, two residual power estimators (RPEs), a noise power estimator (NPE), and a combined noise suppressor (NS) and binary mask. The error signal before the adaptive filter update is $e_{\text{pri}}[n]$, while $e_1[n]$ and $e_2[n]$ are the error signals after the filter update. The noise power estimate and the residual echo power estimate of the m^{th} frame in the k^{th} frequency bin are $\lambda_{V,k}[m]$ and $\lambda_{B,k}[m]$, respectively.

Here we briefly describe the parameters and the computational complexity of the system. A more detailed discussion of the whole system can be found in [19]. The tuning parameters for each of the RAECs consist of the frame size N_{RAEC}, the number of partitioned blocks M_{RAEC}, the number of iterations N_{iter}, the step-size μ_{RAEC}, the tuning parameter γ_{RAEC} for the robust adaptive step-size, and the smoothing factor α_{RAEC} for the power spectral density estimation. The tuning parameters for the DTP consists of the transition probabilities a_{01}, a_{10}, b_{01}, and b_{10}, the smoothing factors α_{DTP} and β_{DTP}, the frequency bin range $[k_{\text{begin}}, k_{\text{end}}]$, the frame duration T_{DTP}, and the adaptation time constants τ. The tuning parameters for the RPE consist of the numbers of partitions M_{RPE_H} and M_{RPE_L} to calculate the coherence and the smoothing factors $\alpha_{\text{RPE}_\text{H}}$ and $\alpha_{\text{RPE}_\text{L}}$ for the power spectral density estimation. The tuning parameters of the NPE consist of the fixed *a priori* speech-to-noise ratio (SNR) ξ_{H_1}, the threshold P_{TH}, and the smoothing factors α_P and α_{NPE} The tuning parameters of the the NS consist of the smoothing factor for the SNR estimator α_{DD}. The tuning parameters for the direct masking consist of the minimum gain G_{min}, the SNR thresholds θ_1 and θ_2, the tuning parameter α, and a binary variable b_m that choses the type of masking applied (based on [25] or *quasi*-binary [19]).

Table 1 shows the computational complexity per sample for each block, where "mply" stands for multiplication, "add" stands for addition, "sqrt" stands for square root, "if-else" stands for the if-else statement, "div" stands for division, "log" stands for the logarithm function, "exp" stands for the exponential function, "MAC" stands for multiply-accumulation, "cplx" stands for complex number, and "pwrSpectr" stands for the square of the magnitude of a complex number. Eventually, the actual complexity is platform dependent, but each of the fundamental operations, such as the FFT, can be estimated in terms of DSP cycles, which in turn allows us to estimate the computation on an actual platform in terms of MCPS. Note that FFT_{RAEC} and FFT_{STFT} represent the FFT cost per sample by dividing the FFT cost by its block size. Also note that some of the tuning parameters, such as the number of partitioned blocks M_{RAEC} and M_{RPE}, the $2N_{\text{RAEC}}$-point FFT of the RAEC, the N_{STFT}-point FFT of the short time Fourier transform (STFT) block, and the number of iterations N_{iter}, will influence directly the complexity. Given the computational complexity of each block, the total computational complexity in terms of MCPS is given by

Table 1. The computational complexity per sample for each block.

$$\begin{aligned}
C_{\text{RAEC}} &= (3N_{\text{iter}}+2)\text{-FFT}_{\text{RAEC}} + (5N_{\text{iter}}+3)\text{-mply} + (3N_{\text{iter}}+1)\text{-MAC} \\
&\quad + (2N_{\text{iter}}+1)\text{-cplx-pwrSpectr} + (2N_{\text{iter}}+1)M_{\text{RAEC}}\text{-cplx-mply} \\
&\quad + N_{\text{iter}}(M_{\text{RAEC}}+1)\text{-add} + N_{\text{iter}}\text{-sqrt} + 2N_{\text{iter}}\text{-div} + N_{\text{iter}}\text{-if-else} \\
&\quad + N_{\text{iter}}M_{\text{RAEC}}\text{-real-cplx-mply} \\
C_{\text{STFT}} &= 2\text{-mply} + \text{FFT}_{\text{STFT}} \\
C_{\text{DTP}} &= 3\text{-cplx-pwrSpectr} + 18\text{-mply} + 12\text{-MAC} + 1\text{-cplx-mply} + 6\text{-div} \\
&\quad + 9\text{-add} + 1\text{-exp} + 1\text{-sqrt} + 1\text{-log} \\
C_{\text{RPE}} &= 1\text{-cplx-pwrSpectr} + 4\text{-mply} + 3\text{-MAC} + (M_{\text{RPE}}+1)\text{-cplx-mply} \\
&\quad + (M_{\text{RPE}}+1)\text{-add} + 1\text{-div} \\
C_{\text{NPE}} &= 1\text{-cplx-pwrSpectr} + 3\text{-div} + 3\text{-add} + 5\text{-mply} + 1\text{-exp} + 3\text{-MAC} \\
&\quad + 2\text{-if-else} \\
C_{\text{NS}} &= 2\text{-cplx-pwrSpectr} + 2\text{-add} + 1\text{-if-else} + 3\text{-mply} + 2\text{-MAC} + 3\text{-div}
\end{aligned}$$

$$\begin{aligned}
\text{C}(\mathbf{p}) = {} & (C_{\text{RAEC}_1} + C_{\text{RAEC}_2} + 7C_{\text{STFT}} + C_{\text{DTP}} \\
& + C_{\text{RPE}_\text{H}} + C_{\text{RPE}_\text{L}} + C_{\text{NPE}} + C_{\text{NS}})\frac{f_s}{10^6}\ \text{[MCPS]}, \qquad (1)
\end{aligned}$$

where $\mathbf{p}$ is the vector of optimization parameters and f_s is the sampling rate. Additionally, there is an on-off flag to either turn on or off the second RAEC block to determine whether using the cascaded structure of two RAEC blocks or running only one RAEC block for a higher number of iterations is more beneficial.

3. OPTIMIZATION FRAMEWORK

3.1. Optimization problem

The SE system tuning can be formulated mathematically as a constrained optimization problem. Let $\hat{s}[n, \mathbf{p}]$ be the SE system output obtained with $\mathbf{p}$, the problem can be written as:

$$\begin{aligned}
&\text{maximize} && \text{Q}(\hat{s}[n, \mathbf{p}]), \\
&\text{subject to} && \text{C}(\mathbf{p}) \leqslant C_{\text{max}}, \qquad (2)
\end{aligned}$$

where $\text{Q}(\cdot)$ is the optimization criterion and C_{max} is the computational complexity constraint. Additionally, we can define $\mathbf{L}$ and $\mathbf{U}$ as the lower and upper bounds of $\mathbf{p}$, i.e., $\mathbf{L} \leqslant \mathbf{p} \leqslant \mathbf{U}$. Since the objective function is nonlinear and not known to be convex, there is no effective method for solving (2). However, the nonlinear programming problem can still be solved by several approaches, each of which involves some compromises [26].

3.2. Optimization algorithm

The genetic algorithms (GAs) have been successfully applied to this type of non-convex mixed-integer optimization problems [27]. The basic idea is to apply genetic operators, such as *mutation* and *crossover*, to *evolve* a set of initial solutions, or *population*, in order to find the solution that maximizes the objective function. The key element of this evolutionary process for dealing with the nonlinear constraints is the so-called *tournament selections*, which allow for several random pairwise comparisons between sets of parameters and quickly determine the boundary of the feasible region [28]. The various steps of the algorithm are outlined below.

Step 1 - An initial population of M solutions is first generated by randomly choosing the values of each set from the feasible region $\mathbf{p}_m^{(0)} \sim \mathcal{U}(\mathbf{L}, \mathbf{U})$. As a general remark, the feasible region determined by the bounds in (2) is larger than the one allowed by the constraint, e.g., the complexity of the $\mathbf{U}$ solution might be much higher than C_{max}. However, a methodology will be used in the evolutionary process to enforce the feasibility of the solution.

Step 2 - The sets that go through crossover or mutation are chosen in a series of tournament selections: a random parameter set ω is extracted from the population, $\Omega \subset \mathbf{\Pi}^{(k)}$, and the set $\mathbf{p}_m^{(k)} \in \Omega$ with the best $\text{Q}(\hat{s}[n, \mathbf{p}_m^{(k)}])$ is then selected. A constraint is imposed in the pairwise comparison of the tournament selection by making sure that when a feasible and an infeasible solutions are compared, the feasible one is chosen, and when two infeasible solutions are compared, the one with smaller constraint violation is chosen [28].

Crossover - This operator allows to combine two sets of parameters with good but not optimum values of their objective function from a previous generation, $\mathbf{p}_n^{(k)}, \mathbf{p}_l^{(k)} \in \mathbf{\Pi}^{(k)}$, through a random weighted mean:

$$\mathbf{p}_m^{(k+1)} = \Phi(\mathbf{p}_n^{(k)}, \mathbf{p}_l^{(k)}) = \boldsymbol{\beta} \odot \mathbf{p}_n^{(k)} + (1 - \boldsymbol{\beta}) \odot \mathbf{p}_l^{(k)}, \quad (3)$$

where $\boldsymbol{\beta} \sim \mathcal{U}(\mathbf{0}, \mathbf{1})$ and $\odot$ denotes element-wise multiplication.

Mutation - The mutation $\mathbf{p}_m^{(k+1)} = \Psi(\mathbf{p}_n^{(k)})$ of the set of values prevents choosing all elements in the population from a local minimum. Different heuristic approaches can be used, often associated with the type of the problem [29, 30]. The uniform perturbation is a simple operator that replaces the value of a l^{th} element with a uniform random value selected between the upper and lower bounds:

$$\Psi_a(p_{n_l}^{(k)}) = \delta, \quad \delta \sim \mathcal{U}(L_l, U_l). \quad (4)$$

Step 3 - When a halting criterion is reached, the set of parameters that maximizes the objective function will be our solution:

$$\hat{\mathbf{p}} = \underset{\mathbf{p}_m^{(K)} \in \mathbf{\Pi}^{(K)}}{\arg\max} \ \text{Q}\left(\hat{s}[n, \mathbf{p}_m^{(K)}]\right) \quad \text{s.t.} \quad \text{C}(\mathbf{p}_m^{(K)}) \leqslant C_{\text{max}}. \quad (5)$$

Note that, given that not necessarily all the solutions in the K^{th} generation might fall within the feasible region [28], we choose the best solution that respects the constraint.

4. EXPERIMENTAL ANALYSIS

4.1. Dataset Generation

A key element to any data driven approach is to have a large and well structured amount of data for training and testing that correlates well to real world scenarios. To properly optimize and evaluate the SE system, two conversational speech databases were generated using the TIMIT database [22] for training and testing. We used the studies presented in [31] and formalized in the ITU-T P.59 standard [32] to generate a full-duplex conversational database composed by near-end (NE) and far-end (FE) speech signals by determining the duration and pattern of talk-spurt (TS), pause (PAU), double-talk (DT), and mutual silence (MS).

An instance of the database was created as follows. We concatenated two sentences, randomly chosen without replacement from a total of 6,300 TIMIT sentences to form the NE speech. We then extracted their voice activity from their phonetic transcription (given on a sample by sample basis) to determine the durations of the speech and non-speech parts. Since the TIMIT sentences have little non-speech sections, we randomly zero-padded the beginning and the end of the concatenated speech file as well as between the two TIMIT sentences so that the speech activity had a uniform duration distribution between 30% to 45% and the non-speech probability between 55% to 70%, in line with the studies on conversational speech presented in [31].

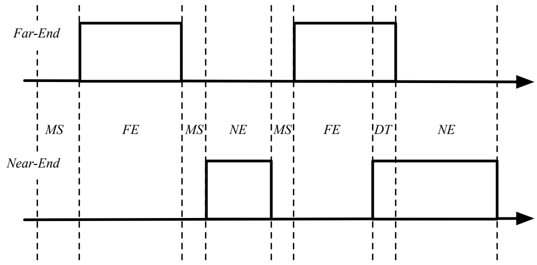

Fig. 2. Example of a conversational speech sequence.

The FE speech pattern was generated using a 2-state Markov chain which is a collapsed version of the 4-state Markov chain used in [18], given that the NE pattern is already given. In particular, from the FE side, MS coincides with NE, creating a PAU state, and DT coincides with FE itself, creating a TS state. We tuned the transition probabilities in the transition matrix of the Markov chain to match the above mentioned statistics of the NE speech using a Markov chain Monte Carlo sampling algorithm [33]. The FE speech database was generated by concatenating and removing pauses from the ITU-T P-Series [34]. Once the on-off speech pattern of the FE was created, we randomly chose the starting and ending point in the FE channel, and then we overlapped it with the NE. Given that certain transitions are not allowed in the conversational model [32], we ran several instances of the Markov chain until the DT probability ranges from 7% to 17%, the MS probability from 20% to 30%, and no DT-MS and NE-FE transitions occurred. An example of the pattern of conversational events is shown in Figure 2.

A noise database comprising of babble noise (e.g., airport, cafeteria, exhibition, and restaurant), white and pink noise, impulsive noise (e.g., hammering), airplane cabin noise, car noise from a variety of car models, and street noise was used. The room impulse responses (RIRs) were calculated in office environments using the Audio Precision APx525 log-swept chirp signal through the *Beats Pill*™ portable speaker and truncated to the desired length. A set of 10 RIRs was then chosen with an average reverberation time RT$_{60}$ = 0.28 s. The 3,150 NE and FE segments were then normalized to -26 dBov to avoid clipping by following the ITU-T Recommendation P.835 [35], and convolved with their respective RIR with normalized unitary energy. The NE signal was mixed with the FE signal at speech-to-echo ratio (SER) uniformly distributed between -30 and 5 dB. The scaling was done by calculating the energy of the signals according to [36]. The noise was then mixed at an SNR uniformly distributed between -5 to 10 dB, according to the noise and the mixed speech signal energies [37]. The choices of RIRs, SER, and SNR were considered empirically appropriate given the possible usage scenarios for a portable teleconferencing device.

4.2. Objective Functions

For the full-duplex communication scenario, we used the standardized POLQA [5] to measure the improvement in MOS. Since POLQA is a full-referenced measurement system, our objective function is the difference in MOS compared to a clean reference, i.e., $\text{Q}(\hat{s}[n, \mathbf{p}]) = \Delta\text{MOS}(\hat{s}[n], y[n])$ [18].

For the ASR front-end scenario, the capability of the recognizer were examined by measuring its accuracy in recognizing phones, the building blocks of words and utterances [38], through PAR. We used the HTK toolkit [39] to train an acoustic model composed of 61 phones [22]. A set of 13 Mel-frequency cepstral coefficients (MFCCs) with their first and second derivatives, for a total of 39 coefficients, were generated and used as features for our experimental analysis. We used a 5-state HMM with an 8-mixture Gaussian mixture model (GMM) for each phone, a fairly standard setup [38]. We normalized the mean of the MFCCs as suggested in [40] for the proper application of the direct masking. We trained our HMMs with clean speech only to focus only on the SE capabilities.

4.3. Optimization Process

For the optimization problem in (2), the total complexity was fixed to $C_{\text{max}} = 50$ MCPS. The genetic algorithm had a population of $M = 100$ possible candidates and $K = 10$ generations, which we observed to be a good trade-off between the accuracy of the solution and the duration of the optimization process. Given the relatively small size of the population, we chose a deterministic tournament selection [41] by calculating the fitness function $\mathrm{Q}\left(\cdot\right)$ for all the elements of the population. A seed was given to generate the initial population by biasing this towards a known hand-tuned solution that achieved reasonable values in the algorithmic design phase, $\mathbf{p}_{\text{INIT}}$. This was done with the same operator used in the crossover operation (3), where each randomly generate solution is weighted with $\mathbf{p}_{\text{INIT}}$ and $\boldsymbol{\beta} \sim \mathcal{U}(\mathbf{0.3}, \mathbf{0.7})$. The best $M = 20$ or less sets of parameters in each generation that fulfill the constraint were migrated to the next generation, of the remaining sets half went through crossover and half through mutation. The optimization process took about 90 hours on a 16-core Intel Xeon machine with parallelized scripts. Note that while the tuning database is fixed, calculating $\mathrm{Q}\left(\hat{s}[n, \mathbf{p}]\right)$ requires running all 3,150 signals for each population element $\mathbf{p}$ at each iteration. The analysis-modification-synthesis as well as the different algorithmic components operated on a 16 ms frame size (256 samples at 16 kHz) with 50% overlap.

4.4. Results

The scatterplots of the fitness values $\mathrm{Q}\left(\cdot\right)$ for each element of the initial population and final population of the evolutionary optimization process are shown in Figure 3. The solution optimized for PAR, $\hat{\mathbf{p}}_{\text{PAR}}$, and the solution optimized for MOS, $\hat{\mathbf{p}}_{\text{MOS}}$, on the training database not only achieve much higher PAR and ΔMOS but also achieve a net 20% reduction in computational complexity. The unconstrained solutions are also calculated, $\hat{\mathbf{p}}_{\text{PAR}_\text{u}}$ and $\hat{\mathbf{p}}_{\text{MOS}_\text{u}}$, respectively. The final sets of parameters are chosen according to (5) and evaluated on the testing database. The results are shown in Table 2.

While similar mean fitness values for the the last population $\mathbf{\Pi}^{(K)}$ and its immediate preceding ones, i.e., $\mathbf{\Pi}^{(K-1)}$ proved overall convergence, it was observed the existence of quasi-optimal solutions within the final population that can have significantly different element-wise values, $p_{m_l}^{(K)}$. This *non-uniqueness* problem, often encountered in nonlinear programming, [26] is, arguably, not a weakness in our case. In fact, having a set of possible candidates with different characteristics increases our chances of determining a set of parameters that offers better properties for our purposes, while still moving in the neighborhood of the optimal value. In this regard, we have observed that the seed for the generation of the initial population given by $\hat{\mathbf{p}}_{\text{INIT}}$, did not affect the values of the final fitness function or the overall behavior of the final population. In fact, $\hat{\mathbf{p}}_{\text{INIT}}$ biased the initial population but did not restrict the actual search region determined by the bounds $\mathbf{L}$ and $\mathbf{U}$. The major impact of pointing the search towards a reasonable path results in speeding up the genetic algorithm convergence by reducing both number of iterations and size of the population.

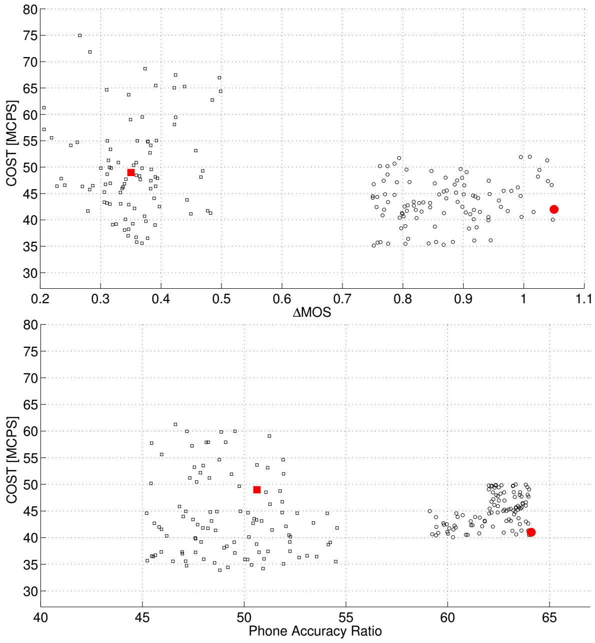

Fig. 3. Initial population (squares) and final population (circles) in the constrained optimization over ΔMOS and PAR on the training database. The initial solution $\mathbf{p}_{\text{INIT}}$ is the red square, while the optimal final solution that respects the constraint is the red circle.

Table 2. Results of the GA optimization algorithm.

	PAR [%]	ΔMOS	$\mathrm{C}\left(\mathbf{p}\right)$ [MCPS]
$\mathbf{p}_{\text{INIT}}$	51.04	0.32	49.14
$\hat{\mathbf{p}}_{\text{PAR}}$	62.94	0.65	41.17
$\hat{\mathbf{p}}_{\text{PAR}_\text{u}}$	63.15	0.68	53.56
$\hat{\mathbf{p}}_{\text{MOS}}$	60.07	0.87	42.56
$\hat{\mathbf{p}}_{\text{MOS}_\text{u}}$	60.22	0.92	55.23

In informal listening, the difference in the output processed with $\hat{\mathbf{p}}_{\text{PAR}}$ and $\hat{\mathbf{p}}_{\text{MOS}}$, follow known differences in SE when targeting recognition and intelligibility versus perceived quality of speech. A clear example is the binary mask being enabled by the optimization process only in the $\hat{\mathbf{p}}_{\text{PAR}}$, while the $\hat{\mathbf{p}}_{\text{MOS}}$ solution exploited the perceptual masking properties of speech in noisy conditions.

5. CONCLUSIONS

In this work, we presented an optimization framework for tuning parameters and selecting algorithms of a speech enhancement system under the constraint of limited computational complexity imposed by a given target platform. The results showed a net improvement over an initial solution hand-tuned by an expert both in terms of mean opinion score (MOS) and phone accuracy rate (PAR). On the test set, the PAR increased by 11.90% and ΔMOS increased by 0.55, while keeping the complexity below the imposed target of 50 MCPS. In fact, the optimized system resulted in a 20% lower complexity solution than the initial hand tuned system. The proposed system can be very helpful in the prototyping phase as well as in the conceptual stage of algorithm design.

6. REFERENCES

[1] J. Benesty, M. Sondhi, and Y. Huang, *Springer Handbook of Speech Processing*, Springer, 2008.

[2] I. Tashev and M. Slaney, "Data Driven Suppression Rule for Speech Enhancement," *Proc. Information Theory and Applications Workshop*, pp. 1–6, 2013.

[3] *Methods for Subjective Determination of Transmission Quality*, ITU-T Rec. P.800, 1996.

[4] S. Möller, W.-Y. Chan, N. Côté, T. H. Falk, A. Raake, and M. Wältermann, "Speech Quality Estimation: Models and Trends," *IEEE Signal Processing Magazine*, vol. 28, no. 6, pp. 18–28, 2011.

[5] *Perceptual Objective Listening Quality Assessment*, ITU-T Rec. P.863, 2010.

[6] *Perceptual Evaluation of Speech Quality (PESQ): An Objective Method for End-to-End Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs*, ITU-T Rec. P.862, 2001.

[7] B.H. Juang, "Speech Recognition in Adverse Environments," *Computer, Speech & Language*, vol. 5, no. 3, pp. 275–294, 1991.

[8] O. Kalinli, M. L. Seltzer, J. Droppo, and A. Acero, "Noise Adaptive Training for Robust Automatic Speech Recognition," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 18, no. 8, pp. 1889–1901, 2010.

[9] O. D. Deshmukh, "Embedded Automatic Speech Recognition and Text-To-Speech Synthesis," in *Speech in Mobile and Pervasive Environments*, N. Rajput and A. A. Nanavati Eds., John Wiley & Sons, 2012.

[10] J. Li, L. Deng, Y. Gong, and R. Haeb-Umbach, "An Overview of Noise-Robust Automatic Speech Recognition," *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 745–777, 2014.

[11] B. Favre et al., "Automatic Human Utility Evaluation of ASR Systems: Does WER Really Predict Performance?" in *Proc. Interspeech*, 2013.

[12] Philipos C Loizou, *Speech Enhancement: Theory and Practice*, CRC Press, 2007.

[13] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, 2007.

[14] S. W. Smith, "Digital Signal Processors," in *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing, 1997.

[15] M. Püschel et al., "Spiral: Code Generation for DSP Transforms," *Proc. of the IEEE*, vol. 93, no. 2, pp. 232–275, 2005.

[16] B. G. Lipták, *Instrument Engineers' Handbook, Volume Two: Process Control and Optimization*, CRC press, 4th Ed., 2005.

[17] D. Giacobello, J. Atkins, J. Wung, and R. Prabhu, "Results on Automated Tuning of a Voice Quality Enhancement System Using Objective Quality Measures," in *Proc. 135th Audio Engineering Society Convention*, 2013.

[18] D. Giacobello, J. Wung, R. Pichevar, and J. Atkins, "Tuning Methodology for Speech Enhancement Algorithms Using a Simulated Conversational Database and Perceptual Objective Measures," in *Proc. 4th Joint Workshop on Hands-free Speech Communication and Microphone Arrays*, 2014.

[19] R. Pichevar, J. Wung, D. Giacobello, and J. Atkins, "Design and Optimization of a Speech Recognition Front-end for Distant-Talking Control of a Music Playback Device," 2014. Available: `arxiv.org/abs/1405.1379`.

[20] I. Tashev, A. Lovitt, and A. Acero, "Unified Framework for Single Channel Speech Enhancement," in *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 883–888, 2009.

[21] S. Watanabe and J. Le Roux, "Black Box Optimization For Automatic Speech Recognition," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 3280–3284, 2014.

[22] J. S. Garofolo et al.,"TIMIT Acoustic-Phonetic Continuous Speech Corpus," Linguistic Data Consortium, 1993.

[23] J. Wung, D. Giacobello, and J. Atkins, "Robust Acoustic Echo Cancellation in the Short-Time Fourier Transform Domain Using Adaptive Crossband Filters," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 1314–1318, 2014.

[24] D. Giacobello and J. Atkins, "A Sparse Nonuniformly Partitioned Multidelay Filter for Acoustic Echo Cancellation," in *Proc. 14th IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2013.

[25] Y. Ephraim and D. Malah, "Speech Enhancement Using a Minimum-Mean Square Error Short-Time Spectral Amplitude Estimator," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 32, no. 6, pp. 1109–1121, 1984.

[26] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[27] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[28] K. Deb, "An Efficient Constraint Handling Method for Genetic Algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 311–338, 2000.

[29] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, John Wiley & Sons, 2012.

[30] D. Markovic, F. Antonacci, A. Sarti, and S. Tubaro, "Estimation of Room Dimensions from a Single Impulse Response," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 1–4, 2013.

[31] P. T. Brady, "A Statistical Analysis of On-Off Patterns in 16 Conversations," The Bell System Technical Journal, vol. 47, pp. 73–91, 1968.

[32] *Artificial Conversational Speech*, ITU-T Rec. P.59, 1993

[33] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An Introduction to MCMC for Machine Learning," *Machine Learning*, vol. 50, no. 1–2, pp. 5–43, 2003.

[34] *Telephone Transmission Quality, Telephone Installations, Local Line Networks*, ITU-T P. Series.

[35] *Subjective Test Methodology for Evaluating Speech Communication Systems that Include Noise Suppression Algorithms*, ITU-T Rec. P.835, 2003.

[36] *Objective Measurement of Active Speech Level*, ITU-T Rec. P.56, 1993.

[37] Y. Hu and P. C. Loizou, "Subjective Comparison and Evaluation of Speech Enhancement Algorithms," *Speech Communication*, vol. 49, no. 7, pp. 588–601, 2007.

[38] L. R. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[39] S. Young et al., *The HTK Book*, Cambridge University Engineering Department, 2006.

[40] W. Hartmann, A. Narayanan, E. Fosler-Lussier, and D. L. Wang, "A Direct Masking Approach to Robust ASR," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 1993–2005, 2013.

[41] M. V. Butz, K. Sastry, and D. E. Goldberg, "Tournament Selection: Stable Fitness Pressure in XCS," *Lecture Notes in Computer Science*, vol. 2724, pp. 1857–1869, 2003.