Received July 18, 2019, accepted July 30, 2019, date of publication August 2, 2019, date of current version August 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2932750

A New Unit Selection Optimisation Algorithm for Corpus-Based TTS Systems Using the RBF-Based Data Compression Technique

MATEJ ROJC AND IZIDOR MLAKAR
Faculty of Electrical Engineering and Computer Science, University of Maribor, 2000 Maribor, Slovenia
Corresponding author: Izidor Mlakar (izidor.mlakar@um.si)

This work was supported in part by the European Regional Development Fund and the Ministry of Education, Science and Sport of Slovenia — project SAIAL through research core funding under Grant ESRR/MIZŠ-SAIAL, and in part by the Slovenian Research Agency through research core funding under Grant P2-0069.

ABSTRACT A major drawback of corpus-based speech synthesis systems is the use of large acoustic inventories, and currently one of the main challenges is the optimal representation of concatenation costs associated with units in the acoustic inventory. These concatenation costs are used to evaluate spectral mismatches between the acoustic units to be concatenated. The combinatorics of costs grows exponentially with the size of the acoustic inventories and can result in hundreds of millions or even billions of concatenation costs to be processed. Therefore, in this paper, we represent a novel unit selection optimization algorithm, which minimizes the size of concatenation costs through the vector quantization-based compression technique and tuple structures. Furthermore, the proposed optimization algorithm is designed to be used as an objective measure to optimize the performance of the unit selection cost function regarding the quality of the speech output, and to evaluate the effect of the vector quantization-based compression technique on its performance. The results obtained show that even when data compression is above 50%, the effect on the quality of the synthesized speech is negligible.

INDEX TERMS Unit selection optimization, vector quantization, data compression, RGD algorithm, RBF-based neural networks, unit selection cost function, concatenation costs.

I. INTRODUCTION

Nowadays, there are two mainstream approaches in text-to-speech synthesis: the corpus-based (i.e. concatenative speech synthesis), and statistical parametric speech synthesis (SPSS). The SPSS systems, based on artificial neural networks (ANNs), have become popular in the text-to-speech (TTS) research field, especially since ANN-based acoustic models offer an efficient representation of complex dependencies between linguistic and acoustic features, while the recurrent neural networks (RNNs) provide an elegant way to model speech-like sequential data that embodies short- and long-term correlations [1], [2]. However, concatenative speech synthesis, with unit selection algorithms in particular, generates more natural-sounding speech [3]. The drawback of the approach is its dependence on a large acoustic inventory. When audio material is of high quality and has good coverage of the possible synthesis units' contexts, the synthesized speech will be of high quality and almost natural-sounding. Moreover, rich acoustic realizations with a good coverage of non-uniform units significantly minimize the possible spectral mismatches (i.e. quality of speech). On the positive side, with concatenative speech synthesis, realistic-sounding machine-generated speech can be created without any, or with only minimal, additional signal processing. This is achieved by analyzing the suitability of all possible combinations of acoustic units based on the spectral mismatch defined as the concatenation costs and calculated in each possible phoneme context.

The idea behind the unit selection algorithm in the corpus-based TTS system PLATTOS is to select the most spectrally suitable sequences of acoustic units from the available acoustic inventory [4]. For high-quality speech, a good coverage with diphones and triphones in many acoustic realizations must be ensured, e.g. Figure 1. In this case, lighter circles denote several clusters of triphones, while darker circles

The associate editor coordinating the review of this manuscript and approving it for publication was Victor Sanchez.

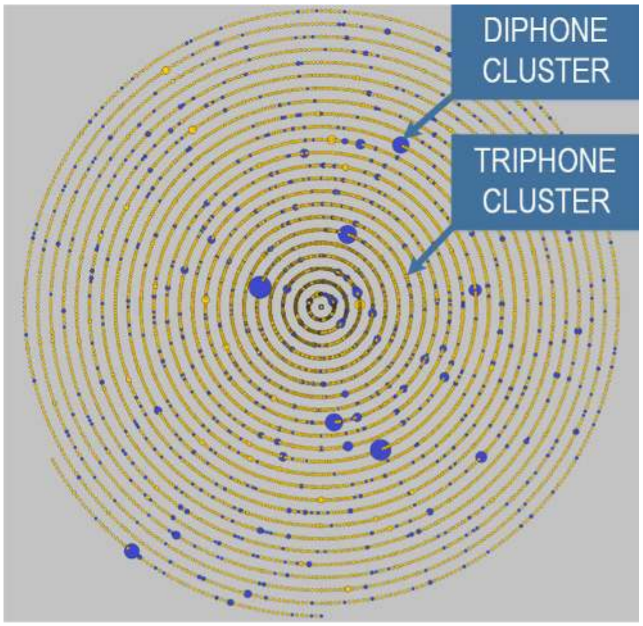


FIGURE 1. The acoustic inventory with diphones and triphones in the PLATTOS acoustic inventory.

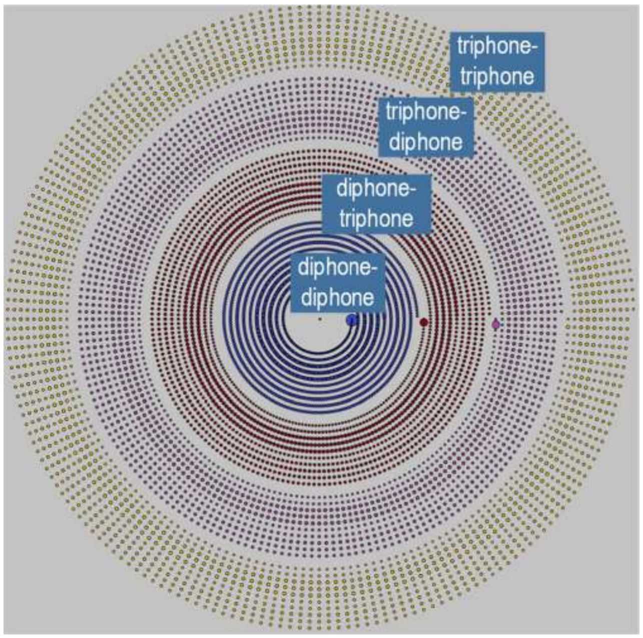


FIGURE 2. Representation of the concatenation costs space between each possible sequence of acoustic units in the PLATTOS acoustic inventory.

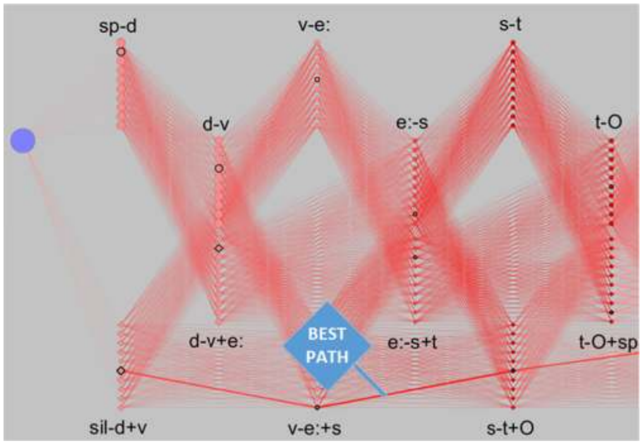


FIGURE 3. Unit selection algorithm - searching for the best path between several synthesis units (diphones or triphones) with several acoustic realizations.

denote several clusters of diphones. The dimension of each circle is proportional to the number of the same acoustic units within each cluster. As outlined in Figure 1, longer units are preferred, since longer speech units yield more natural-sounding synthesized speech [5]. In order to achieve the best speech quality, it is very important that spectral mismatches at the concatenation points between acoustic units in a sentence are minimal. These spectral mismatches are judged automatically by using the aforementioned concatenation costs. The suitability between any two acoustic units in a specific linguistic and prosodic context is evaluated by using a unit selection cost function in the unit selection algorithm. The process utilizes local cost, target cost and concatenation cost in order to select the best pairs of acoustic units with minimal spectral mismatches between units (i.e. minimal cost). In the PLATTOS TTS system, concatenation costs are calculated off-line for all possible candidates and in all possible linguistic contexts, based on acoustic features, such as energy and the spectral distances around concatenation points. The 'cost' space calculated for an inventory represented in Figure 1 is outlined in Figure 2. The four groups in Figure 2 separate concatenation costs for diphone-diphone, diphone-triphone, triphone-diphone and triphone-triphone combinations (direction inwards). Within each group, concatenation costs are clustered into 2,000 clusters. The dimension of circles is used to represent the number of costs within each unit's cluster. Therefore, acoustic inventories in corpus-based TTS systems uses acoustic inventories with many acoustic units with several acoustic realizations. Only in this way are we able to minimize the required signal processing at the concatenation points.

Such high-quality inventories generate huge costs space that the algorithms must process. When the unit selection algorithm utilizes off-line calculated concatenation costs, it can run near real time, while its performance depends primarily on the performance of the unit selection cost function. Its goal is to find the best path in a huge graph with many acoustic units for each sentence. Thus, it operates in a huge concatenation costs space, and has to process all the possible pairs of acoustic units, as can be seen in Figure 3. As already mentioned, the quality of the synthesized speech significantly relies on how close to human judgements the resulting sequences of acoustic units are at the end.

As a result, there has to be a compromise between the targeted quality of the synthesized speech and the size and quality of the acoustic inventory. To sum up, the general issues discussed in the paper are, therefore, how to evaluate the effect on synthesized speech quality due to compression of data resources and, more importantly, how to minimize and mitigate this effect on the unit selection algorithm through relaxed gradient descent (RGD)-based optimization and a novel objective measure. The proposed unit selection optimization algorithm could be a very good solution for these issues in corpus-based TTS systems. It optimizes the unit selection algorithm and unit selection cost function based on objective measures, and in this way also gives us an answer about the effect of the compression of data resources regarding concatenation costs on the unit selection process. In line with advances in artificial neural networks (ANNs), we propose to tackle the data compression issue through reduction of the concatenation costs space (Figure 2), while keeping the size of the acoustic inventory (Figure 1). For this task we deploy a VQ-based lossy data compression method by implementing radial basis function (RBF) with k-means clustering for the reduction of data, and a tuple structure for time-and-space-efficient access to these data. The proposed unit selection optimization algorithm optimizes concatenation costs space, the unit selection algorithm and the unit selection cost function based on an objective measure. This measure is also used to evaluate the effect of the data compression on the effect on synthesized speech quality. We argue, that since we maintain all possible combinations in all possible contexts and reduce only the concatenation costs space by means of clustering rather than elimination, the proposed unit selection optimization will have a minimal impact on the quality of the synthesized speech. Moreover, since the data resources required to use a given acoustic inventory are compressed to a smaller footprint, larger acoustic inventories may be incorporated, thus enabling us to further improve the quality of the synthesized speech.

The paper is structured as follows. Section 2 provides an overview of related work on speech synthesis and the optimization of large data resources in particular. Section 3 outlines the fuzzy logic formalism used to implement the unit selection process and the proposed optimization concepts and algorithm. Section 4 shows the results of the optimization algorithm. Section 5 provides the discussion, and the conclusions follow.

II. RELATED WORK

With the advances in deep learning (DL), corpus-based TTS systems have been challenged by various parametric speech synthesis (SPSS) approaches, incorporating hidden Markov models (HMM), or classification and regression tree clustering, deep neural networks and deep recurrent neural networks (DRNNs). More recently, techniques such as: WaveNet [6], Deep Voice [7] and Tacotron [8], [9] have been introduced. Overall the SPSS promises robust and fully flexible speech synthesis systems with a voice building process that may be largely automated [10]. However, the computational efficiency, robustness, and quality of synthesized speech compared with the unit selection-based techniques are a subject of debate [11], [12]. A corpus-based TTS system utilizing a unit selection algorithm is a speech synthesis technique, which can synthesize speech close to natural quality [3].

However, the main drawback of the corpus-based TTS system is the huge speech database that is required for best performance. With huge acoustic inventories a search for the best sequence of acoustic units is computationally very expensive, since the combinatorics of possible acoustic units to be concatenated results in hundreds of millions and even billions of possible combinations. Therefore, this represents a big performance issue regarding time (when calculated on-line), or regarding memory consumption (when calculated off-line). As a result, nowadays corpus-based TTS systems tend to limit the size of acoustic inventories and perform on-line calculations. For instance, pruning [3], [13], [14] and compression-based [15], [16] approaches are used to minimize the footprint of a speech synthesis system by reduction in corpus size. Most of the compression-based approaches are motivated by linear prediction-based speech coding methods [17]. Sparse representation has also been proposed to implement linear prediction analysis. However, the minimization of acoustic inventories may have a significant impact on the speech quality and lead to its degradation [18]. In general, speech coding methods are proposed to compress unseen speech data efficiently. The speech synthesis system, however, utilizes speech data, which is available beforehand. Thus, the systems will perform best when the database is large and when the audio quality is good [3], [19]. The size of the acoustic inventory is closely related with spectral mismatches in the synthesized speech (i.e. spectral compatibility between two consecutive acoustic units); the corpus-based TTS system should thus utilize an optimal set of joint and target sub-costs and corresponding weights. With the reduction of the acoustic inventory, the possibility of spectral mismatches between units only increases. Various approaches to perceptual evaluation, involving human listeners, have also been proposed [20]–[22]. However, creating an optimal corpus via ''listening'' is very time-consuming, and very expensive to conduct, especially when optimization processes have to be repeated several times. In order to optimize acoustic inventories, and the process of creating them, objective measure-based approaches have also been proposed. These approaches implement a human-like perception of spectral mismatches, by quantifying the compatibility of speech units for concatenation [23]–[25]. The advantage is that the optimization can be repeated far more easily and as many times as necessary, and the results between runs retain consistency and are statistically comparable [23].

In addition to the optimization of acoustic inventories and data sets the optimization of unit selection cost function, within a unit selection algorithm, represents an alternative method in improving the performance of the concatenative speech synthesis systems.

The unit selection algorithm's task is represented as a shortest path problem (SPP) in a huge graph of acoustic units, where an algorithm must find the best path. Value iteration networks, which mimic the application of value iteration, were shown to be capable of computing near optimal paths [26], [27]. In the particular case of a unit selection algorithm, the transition network is a graph defined by weights. The lookup for the best possible transition must be evaluated for every di- and triphone used in the input sequence. Thus, if we want to achieve responsiveness in interactive time, the whole graph must reside within the memory. In order to minimize the size of this graph, researchers adopted various approaches: for instance, a weight space search (WSS) algorithm generates a model through a search of a finite set of possible weights by utilizing analysis-by-synthesis exploration [21]. The Viterbi algorithm can also be implemented in order to find the best matching sequence of acoustic units. Since the Viterbi algorithm does not scale well with the number of acoustic units in an acoustic inventory, a multi-layer Viterbi algorithm was proposed [28], [29]. The computational requirement of both methods grows exponentially with the number of weights being tuned [30]. Researchers have also investigated the use of other methods, such as genetic algorithms (GAs) [21], [31] and the A$*$ algorithm [32] and multi-linear regression (MLR) [33]. These methods specifically target the optimal search algorithm over large databases. GAs are stochastic search algorithms and require a large number of manual evaluations for each acoustic unit in the acoustic inventory. The MLR approaches perform optimization by predicting the objective distance by weighting the sub-costs linearly. Both methodologies, however, fail to adjust the weights for acoustic units in the acoustic inventory, which are not accounted for in the optimization; a large number of user evaluations is therefore required. The main issue with GA and Viterbi-based approaches is that none of them scales well with the number of acoustic units in the acoustic inventory. Finally, from the perspective of the SPP problem, the Dijkstra algorithm [34], optimized with a relaxed gradient descent technique, has also been applied [4].

Concatenation costs must evaluate spectral mismatch for millions of combinations and, therefore, increase data resources exponentially with the increase in the targeted quality of the speech. For instance, in [4] we have utilized 300 sentences in the acoustic inventory. In turn, roughly 147 million instances of concatenation costs were generated, which amount to a 1.2 GB footprint. In order to reduce the footprint of the TTS system further and to be aware of the effect on the speech quality, we propose an optimization algorithm of the unit selection cost function when utilizing compressed concatenation costs. We apply vector quantization (VQ)-based compression [35] and tuple structures. In particular, we exploit radial basis function (RBF) with k-means, as clustering that has been intensively studied and widely applied to various applications processing large datasets [36]. Artificial neural networks have been proven to be successful in quantization approaches. The more popular of these methods are the Kohonen's networks i.e. learning vector quantization (LVQ), k-nearest-neighbors (kNN), self-organizing maps (SOM); the multilayer perceptrons (MLP), and radial basis function networks (RBFNs). The major drawback of the MLP approach is that the determination of local minima is required [37]. Moreover, since it is based on back propagation (BP), the neural network's structure (i.e. adaptive capability, hidden layers, number of neurons) should be optimal to a given data set. Both processes are complex and time-consuming. LVQs [38], [39] and RBFNs [36], [40], on the other hand, represent a smart alternative. Namely, both methods utilize a self-organizing network approach, which uses the training vectors to recursively ''tune'' the placement of competitive hidden units that represent categories of the inputs. In the RBFN, in particular, each neuron stores a ''prototype'', which is just one of the examples from the training set. When one wants to classify a new input, each neuron computes the Euclidean distance between the input and its prototype. In this sense the RBF-based approaches make neurons more locally sensitive; i.e. the model will generate more clusters where the ''value space'' is denser, while in those parts where the values are sparse and scattered significantly less clusters will appear, Thus, the rationale behind selecting the RBFN is its local sensitivity. We argue that since RBFN will optimize the databases in terms of density, the lossy compression will have minimal impact on the overall quality of the synthesized speech. At the same time, the method will significantly reduce the unit selection algorithm's computational footprint and speed up its operation. Moreover, the RBFN-based methods tend to be more data-set resilient and outperform traditional (i.e. LVQ-based) approaches in both speed and accuracy and can even approach the capability of BP-based methods [41].

However, as with any lossy data compression, the VQ-based compression of concatenation costs introduces less precise information about acoustic mismatches in the unit selection algorithm. Thus, in addition to a locally sensitive compression method, it is very important to have an objective measure in order to evaluate this effect. With such a measure, one is also able to find the best compromise between data compression and speech quality. Instead of using subjective and perceptive testing on acoustic inventories, as generally used for this purpose, in this paper a novel objective measure is proposed that can easily evaluate the effects the data compression has on the speech quality. Using the proposed method, the data compression may be easily and quickly set to best targeted cut-off performance. The proposed method achieves comparable speech quality even when the footprint of the concatenation costs is reduced by about 50%. We show that the effect of the compression on speech quality is still negligible.

III. THE UNIT SELECTION OPTIMISATION ALGORITHM IN A CORPUS-BASED TTS SYSTEM

A. UNIT SELECTION ALGORITHM

The text-to-speech-synthesis system PLATTOS is a corpus-based TTS system with innovative omni-comprehensive architecture [4]. The unit selection algorithm is based on unit selection cost function, which evaluates mismatches between two acoustics units. In the PLATTOS TTS system

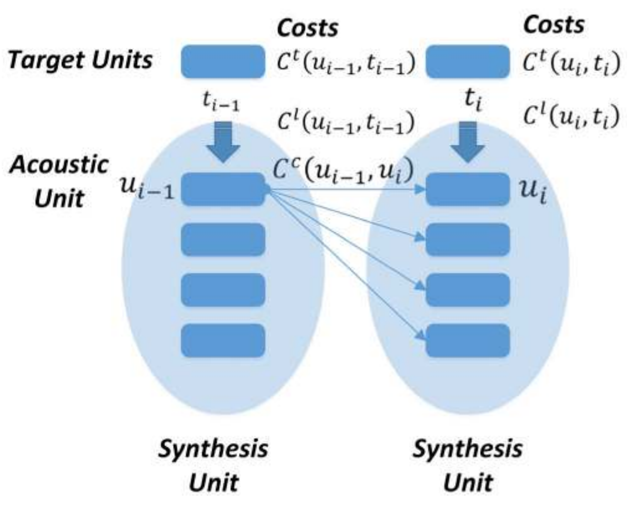


FIGURE 4. **Partial costs that are used in unit selection cost function are responsible for the estimation of how appropriate it is for a given acoustic unit in a sequence to be concatenated, where C^t is target cost, C^l local cost, and C^c concatenation cost.**

the following costs are used for the evaluation of the mismatch between two acoustic units, as shown in Figure 4:

- *local cost* $cost_{lc} = C^l(u_i, t_i)$: a distance between unit candidate u_i and the target unit specification t_i (as defined by the prosody modules), by considering their linguistic context.
- *target cost* $cost_{tc} = C^t(u_i, t_i)$: a distance between unit candidate u_i and the target unit specification t_i, by considering acoustic prosody information regarding unit-duration and pitch that can be calculated, after the target prosody is specified.
- *concatenation cost* $cost_{cc} = C^c(u_{i-1}, u_i)$: describes acoustic mismatches at the concatenation points between successive unit candidates u_{i-1} and u_i, by considering energy and spectrum.

The fuzzy-based unit selection cost function, which defines the cost of selecting unit u_i in the context of u_{i-1} is defined as follows:

$$C(u_i) = \left((S_{cc}(cost_{cc}))^{w_{cc}} \cdot \left(\prod_{j=1}^{3} \left(S_{tc_j}\left(cost_{tc_j}\right)\right)^{w_j} \right)^{w_{tc}} \cdot (S_{lc}(cost_{lc}))^{w_{lc}} \right)^{w_{unittype}} \tag{1}$$

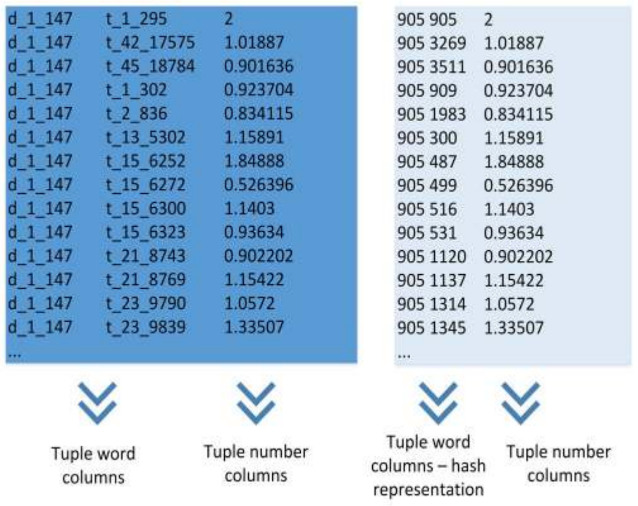
where $C(u_i)$ represents the overall cost of selecting unit u_i. The S_{cc} represents the fuzzy-based partial cost function used for evaluating the cost of selecting u_i from the spectral (i.e. acoustic) perspective (i.e. $cost_{cc}$). The S_{tc} represents the fuzzy-based partial cost function used for evaluating the cost of selecting u_i from the prosodic perspective. In the case of prosodic context, the target cost $cost_{tc}$ is determined by calculating distances by considering pitch, duration, and energy. The S_{lc} represents the fuzzy-based partial cost function used for evaluating the cost of selecting u_i from linguistic perspective. The local cost $cost_{lc}$ is determined by calculating distances regarding the linguistic context between the unit candidates. Finally, since contributions from each fuzzy-based partial cost functions are not equally important, weights w_{cc}, w_{tc}, w_j, w_{lc}, $w_{unittype}$ are used in (1) in order to emphasise the relative importance of each of the criteria, including the type of the unit (i.e. diphone vs. triphone) in a given context.

In this way, the unit selection cost function IN (1) is fuzzy-based. Its performance obviously depends on the optimization of parameters that define its complex multidimensional shape, when used in the SPP algorithm. The SPP algorithm itself is based on weighted finite-state machine (WFSM) formalism, and implements the Dijkstra algorithm [42]. Such a unit selection algorithm represents the most computationally demanding process in the concatenative speech synthesis [43], since acoustic inventory is based on a large speech database that is rich in various acoustic contexts (i.e. duration, pitch, energy and linguistic contexts), and contains non-uniform acoustic units (i.e. diphones and triphones) in multiple contexts. The unit selection algorithm has to perform a SPP algorithm for each sentence in order to select the best sequence of acoustic units with minimal discontinuities (i.e. spectral mismatches). Further, the SPP algorithm is performed on-line, thus the unit selection algorithm has to evaluate a large number of unit candidates in order to differentiate between many acoustic units' contexts captured in the acoustic inventory.

As can be seen in (1), the fuzzy-based unit selection cost function consists of several partial cost functions through which the unit selection algorithm can effectively differentiate between acoustic units in various acoustic contexts. Since partial cost functions are fuzzy-based, their values are always in the interval [0.0, 1.0]. The suitability of acoustic unit u_i is thus determined through the partial cost functions' shapes, where the best-fitted targets have value 1.0, while other neighboring targets have values assigned as defined by the partial cost functions' shapes. In this way, in (1) the shape of individual partial cost functions emphasizes significant mismatches, while the relative importance of a particular criterion is expressed through the shape of each partial cost function. Additionally, the multiplication of partial cost functions ensures that even small mismatches are noticeable in the overall cost. The overall costs for the first and final acoustic units consist of local and target cost, while for other acoustic units they are comprised of local, target and concatenation cost.

B. CONCATENATION COSTS

Calculation of concatenation costs $cost_{cc}$ is very time-consuming in corpus-based TTS systems, and can even not be feasible to perform on-line, when interactive speed or near real-time performance is needed. When calculated on-line, the concatenation costs have to be calculated between all the phonetically matched acoustic units needed in the sentence that can be found in the acoustic inventory in several contexts. Moreover, in order to evaluate spectral mismatches at the concatenation points, the speech samples of all these acoustic units have to be loaded. The off-line calculation of $cost_{cc}$



d_1_147	t_1_295	2
d_1_147	t_42_17575	1.01887
d_1_147	t_45_18784	0.901636
d_1_147	t_1_302	0.923704
d_1_147	t_2_836	0.834115
d_1_147	t_13_5302	1.15891
d_1_147	t_15_6252	1.84888
d_1_147	t_15_6272	0.526396
d_1_147	t_15_6300	1.1403
d_1_147	t_15_6323	0.93634
d_1_147	t_21_8743	0.902202
d_1_147	t_21_8769	1.15422
d_1_147	t_23_9790	1.0572
d_1_147	t_23_9839	1.33507
...		

905	905	2
905	3269	1.01887
905	3511	0.901636
905	909	0.923704
905	1983	0.834115
905	300	1.15891
905	487	1.84888
905	499	0.526396
905	516	1.1403
905	531	0.93634
905	1120	0.902202
905	1137	1.15422
905	1314	1.0572
905	1345	1.33507
...		

FIGURE 5. Tuple structure used for compact representation of concatenation cost, when using non-lossy data compression).

costs guarantees better run-time and real-time performance of the unit selection algorithm.

However, in the off-line case, the acoustic unit sequences needed for sentences are not known in advance. Thus, the unit selection algorithm must have available concatenation costs between any phonetically matched acoustic units in the acoustic inventory. This results in a huge and sparse concatenation cost matrix consisting of floating-point numbers (Figure 5). Additionally, an efficient representation and fast lookup is a necessity if the unit selection process is to retain its real-time capability. The memory footprint of concatenation costs can be defined as:

$$M_f = N \cdot N \cdot sizeof\,(float), \tag{2}$$

where N represents the number of acoustic units in the acoustic inventory, and $sizeof\,(float)$ is equal to 4 bytes. In the PLATTOS TTS system for time-and-space efficient data representation, tuple structures are used as proposed in [4].

A tuple structure $T^{i,j}$ is a finite function $(W_1 \times \ldots \times W_i) \rightarrow Z^j$, where $(W_1 \times \ldots \times W_i)$ are sets of strings, and Z are floating numbers. This finite function maps possible pairs of acoustic units into floats. In the TTS system, the tuple's word columns represent diphone IDs and triphone IDs, while the number column represents concatenation costs, as can be seen in Figure 5. Further, in the tuple structure, perfect hash finite automata are used, where an automaton is used for each finite set of acoustic units' IDs. Thus, a word column W is represented by a minimal deterministic acyclic finite automaton N that accepts each acoustic unit's ID in a word column W. Further, each transition in automaton N is assigned an integer number j, while the sum of integers along the accepting path in N is i. In this way, each perfect hash automaton represents an acoustic unit's ID in each word column with hash-keys. If the overlap between the acoustic units' IDs from multiple word columns is high enough, a single perfect hash automaton can even be used. In Figure 5, unique pairs of acoustic units are represented by 2 perfect hash finite automata, since for the corresponding tuple structure T the transformation $T\,(W_1 \ldots W_i) = \left(z_1 \ldots z_j\right)$ converts sequences of acoustic units' IDs into hash-keys by using perfect hashing. Moreover, for the minimization of the footprint, it is important that each individual hash-key is represented with as few bytes as are required by the largest integer number used in each word column. Further, the number column in tuple T (i.e. the third column in Figure 5) keeps the concatenation costs $cost_{cc}$. These costs are floating-point numbers and for their compact representation, each number is decomposed into a normalized mantissa m and an exponent t which are both stored with the minimal number of bytes. Nevertheless, since different computer platforms represent real numbers in a different way and increasing the precision can increases the memory footprint for storing each number, the tuple structure still requires significant memory resources to be allocated when acoustic inventories are huge. Therefore, in order to solve this issue, a VQ algorithm is proposed to compress even more the number column with concatenation costs, and in this way further reduce the memory footprint. The VQ algorithm used for this purpose is outlined in the next section.

C. RADIAL BASIS FUNCTION K-MEANS FOR CLUSTERING

VQ represents an efficient technique for data compression. In codebook clustering the VQ is generally implemented with k-means clustering, or Linde-Buzo-Gray (LBG) algorithm. However, in the case of large vector dimensions and large codebooks, the use of these methods suffers from complexity [44]. The constrained VQ methods, such as partitioned VQ, can be used to reduce storage and computation complexity. Nevertheless, these methods can severely increase the coding error [45]. With the advances in artificial neural networks (ANNs), the traditional VQ methods can also be significantly optimized for large codebooks. Since we are dealing with a huge number of concatenation costs, in this paper we propose to adopt the radial basis function network (RBFN) approach. This approach, as outlined in Figure 6, uses a three-layered feed forward ANN with strong approximation capability. The input vector is an n-dimensional vector which is being classified. The hidden layer is designed in a way that allows for direct mapping (without additional weights) of input vector into neurons in the layer. One neuron in the input layer corresponds to each predictor variable. With respect to categorical variables, n-1 neurons are used, where n is the number of categories. Thus, the nonlinear function of hidden neurons is symmetrical to the input space, and the output of each hidden neuron depends only on the radial distance between the input vector and the centre of the hidden neuron.

Hidden layers have a variable number of neurons. Each neuron in a hidden layer stores a ''prototype'' vector which is just one of the vectors from the training set. Each neuron compares the input vector to its prototype, and outputs a value between 0 and 1 which is a measure of similarity. $h(x)$ is the

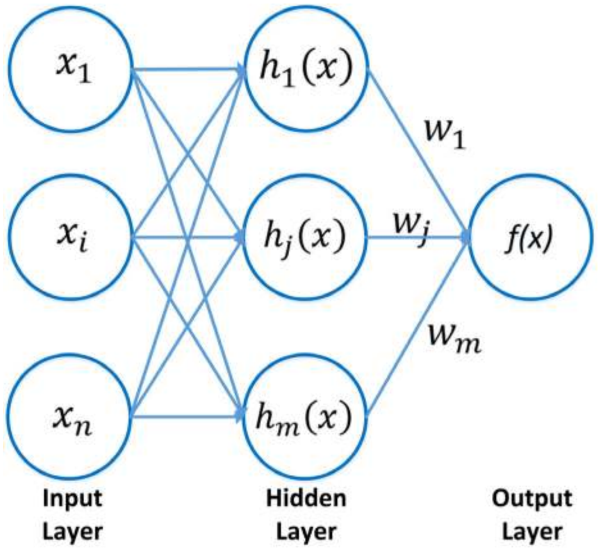


FIGURE 6. RBFN used in proposed vector quantization algorithm.

Gaussian activation function defined as:

$$h_i(x) = \varphi(x_P - c_i) = \exp\left[-\frac{1}{2\sigma^2}\|x_P - c_i\|^2\right], \quad (3)$$

where x_P is the p^{th} sample ($x_P = (x_1^P, x_2^P, \ldots, x_n^P)^T$); c_i is the centre or average of the Gaussian function (i.e. the input space); and $r = \frac{1}{2\sigma^2}$ is the standard deviation Gaussian function (i.e. the width).

The output layer is a linear function implemented as a weighted sum of the outputs of the hidden layer. It is defined as:

$$f(x) = \sum_{j=1}^{m} \mathrm{w}_j h_j(x) = \sum_{j=1}^{m} w_j \exp\left[-\frac{1}{2\sigma^2}\|x_P - c_j\|^2\right], \quad (4)$$

where w_j denotes the weights between the hidden layer and the output layer.

Overall, there are three types of parameter to be solved; the weight w between the hidden nodes and the output nodes, the centre c of each neuron of the hidden layer and the unit width r. The learning process is based on adjusting the parameters of the network to reproduce a set of input-output patterns. In order to determine the centers, any clustering algorithm can be used. A set of clusters, each with r-dimensional centers, is determined by the number of input variables, or nodes, of the input layer. The cluster centers then become the centers of the RBF units. The number of clusters is a design parameter and determines the number of nodes in the hidden layer.

We have selected the *k-means* algorithm to find the clusters. This is a clustering algorithm based on distance: its goal is to discover the natural grouping of data points in a given set [46]. It is also a method commonly used to automatically partition large data sets. Further, *k-means* is one of the most widely used algorithms for clustering because of simplicity, efficiency, and empirical success. In the training step, a standard *k-means* algorithm has to find the cluster centroids, or so-called code words, while in the coding stage, the encoder searches for the nearest centroid for a new vector. The encoder additionally transmits its index to the decoder in order that the decoder can retrieve the corresponding code word in the codebook. The distortion is defined as:

$$d = \left\|X - \hat{X}\right\|^2, \quad (5)$$

where $\hat{\boldsymbol{X}} = \boldsymbol{Q}(\boldsymbol{X})$ represents the quantized vector.

In this way, the *k-means* clustering technique is based on a distance matrix, using Euclidean distance as a criterion. It starts with m initial cluster centers. For all data, Euclidean distance from each cluster centre is then calculated, after which the data points are assigned to the closest cluster centre. This method is repeated until the squared error between the empirical mean of a cluster and the points in the cluster is minimized. The goal is to divide the entire dataset by individual m clusters, where the number of elements in each cluster is n_i, while the centre of a cluster is c_i. The algorithm proceeds until sufficient progress cannot be made, i.e. the minimal Euclidian distance between observations and centroids had been reached. This yields a codebook mapping centroid to codes and vice versa.

By applying the outlined RBF network over the concatenation costs space, we are able to significantly reduce the memory footprint of the tuple structure in the unit selection algorithm. Due to the lossy data compression, however, VQ-induced errors are bound to appear. In order to evaluate them we could, as already mentioned, perform a series of perceptive tests. However, such a task would be highly subjective, as well as cost- and labor-intensive, especially when optimizing the size of compressed concatenation costs space to allow for the best possible performance of the unit selection cost function at the lowest computational cost (including memory footprint). Therefore, we propose a novel optimization process based on an objective measure. As outlined in the following section, it allows us to reliably and fully automatically estimate the impact of the VQ algorithm on the quality of speech.

D. A RELAXED GRADIENT DESCENT-BASED OPTIMISATION ALGORITHM

As a result of data compression of concatenation costs, it is expected that errors will be introduced. These errors will generate spectral discontinuity and will have a negative effect on the performance of the unit selection cost function. Namely, the errors can distort the information about spectral mismatch between adjacent pairs of acoustic units. In order to search for the best performance of the unit selection algorithm operating in maximally compressed space, it is therefore important to measure and objectively evaluate these errors. By comparing unit selection's performance prior and after the compression, via the proposed method, it is possible to a) estimate the effect of the data compression through an objective measure, and b) search for the best compromise between unit

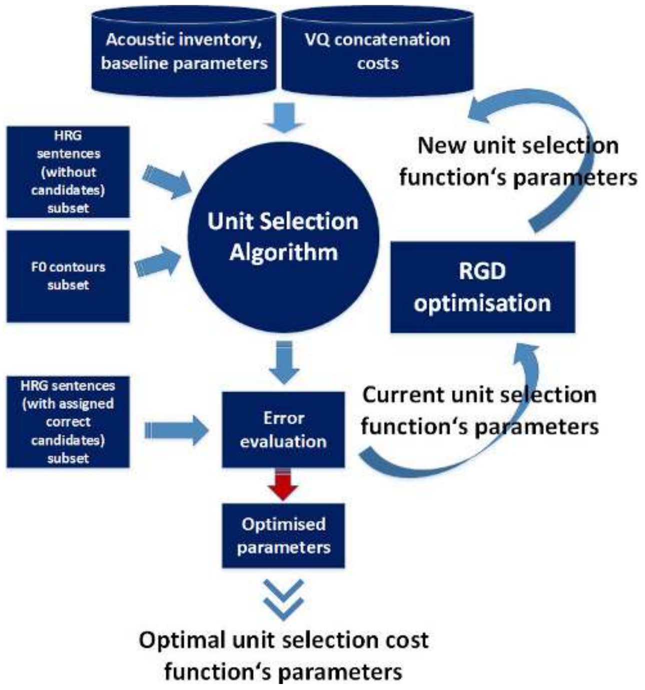

Acoustic inventory, baseline parameters
VQ concatenation costs
HRG sentences (without candidates) subset
F0 contours subset
Unit Selection Algorithm
New unit selection function's parameters
RGD optimisation
HRG sentences (with assigned correct candidates) subset
Error evaluation
Current unit selection function's parameters
Optimised parameters
Optimal unit selection cost function's parameters

FIGURE 7. An RGD-based algorithm for optimizing unit selection cost function.

selection performance (i.e. data compression) and overall quality achieved on some selected acoustic inventory.

For the measuring and the evaluation process, a fully automatic optimization algorithm, based on the RGD technique, is proposed. The algorithm is outlined in Figure 7.

As outlined in Figure 7, the input into the unit selection algorithm is a set of prosodically annotated utterances, all concatenation costs (off-line calculated, compressed, and uncompressed), and acoustic inventory consisting of diphone and triphone acoustic units. Each utterance is represented in the form of a heterogeneous relation graph (HRG) that contains comprehensive linguistic and prosodical information, original pitch (i.e. F0 contour), and original acoustic units from the acoustic inventory. In this way, target unit specifications for all acoustic units in the sentence are provided in detail. The unit selection algorithm then searches for a sequence of acoustic units from the database of utterances by using the unit selection cost function in (1) and iterates the RGD-based optimization process until the best set of parameters is generated. The goal of the optimization is to find as many original acoustic units at the same positions in the input sentence as possible. Since prosody is available for every sentence, the optimized algorithm should select exactly the same sequences of acoustic units, when the optimized parameters shape the cost function accordingly. However, the issue of applying the RGD algorithm in TTS systems for the optimization of (1) is, in general, that the prediction of prosody (at symbolic and acoustic level) is utilized, before searching for the best sequence of units of an individual sentence. As a result, degradation in the speech quality, generated by acoustic mismatches, can arise from many sources: from the prosody module, from the acoustic inventory used (a lack of suitable units with the necessary acoustic realizations for synthesis units), from the shape of the unit selection cost function, or from the unit selection algorithm itself (best path search). Thus, it is hard to reach solid conclusions regarding the origins of the acoustic mismatches.

In the proposed solution (Figure 7) this issue is solved, via the '*Error evaluation*' component and the RGD algorithm optimizes unit selection cost function performance through an objective measure. After each iteration, the evaluation of the unit selection cost function's performance is automatically performed, by calculating the error regarding how many correct acoustic units were selected at each position and within each observed sentence. The proposed objective measure is defined as:

$$Error = \frac{N}{M} = \frac{\left(\sum_{j=1}^{S}\sum_{i=1}^{length(Sentence(j))}\left(SelectedUC_i != OriginalUC_i\right)\right)}{\left(\sum_{j=1}^{S}\sum_{i=1}^{length(Sentence(j))} SelectedUC_i\right)}, \tag{6}$$

where N represents the number of acoustic units selected incorrectly, while M represents the number of all acoustics units in the given subset of sentences S used in the unit selection optimization process.

The *Error* calculated then represents an input for the RGD optimization algorithm. It is used for tuning and updating the parameters of the unit selection cost function. The proposed unit selection optimization algorithm stops when the *Error* drops below a pre-defined threshold or a pre-defined number of iterations is reached. Thus, the value of the *Error* also indicates when the parameters have been optimized to model the shape of the unit selection cost function to match the optimal curve.

For the implementation of the RGD-based optimization process in Figure 7 we propose a relaxed gradient descent (RGD) algorithm as defined in [47]. The gradient descent algorithms are popular approaches when searching for optima in both statistical, as well as fuzzy-based, selection algorithms [48]. The RGD algorithm considers the following search for minimal value problem:

$$\min f(x), \tag{7}$$

where f(x) represents the unit selection cost function in (1), which is continuously differentiable. The necessary and sufficient condition for a point x^* (a vector of weights and fuzzy-membership functions' parameters) to be optimal for (7) is then:

$$\nabla f\left(x^*\right) = 0, \tag{8}$$

In general, this search problem is solved through iterative steps, which generate the following sequence of points: $x_0, x_1, \ldots \in dom f$, for which $f(x_k) \rightarrow f^*$ as k$\rightarrow \infty$.

Thus, the algorithm for solving (7) generates a minimizing sequence of parameters x_k, k = 0,1, ... as:

$$x_{k+1}=x_k+\theta_k \cdot t_k \cdot d_k, \tag{9}$$

where $t_k > 0$ is a step size, while the d_k is the search direction. The RGD optimization in Figure 7 for tuning the unit selection cost function in (1) performs, therefore, the following steps:

a) Compute the search direction for *k'th* parameter in (1): $d_k = -\nabla f\ (x_k)$
b) Define the step length for *k'th* parameter in (1) t_k via backtracking line search procedure
c) Update all parameters in (1): select random value $\theta_k \in (0, 1)$ and update the parameters by using: $x_{k+1} = x_k + \theta_k \cdot t_k \cdot d_k$
d) Test criterion for stopping the iterations. If the test is fulfilled, stop; otherwise consider $k = k + 1$ and continue.

In this way the RGD optimization algorithm consists of the following processing steps: during the unit selection process, the unit cost functions' partial derivatives are calculated ($\vartheta \mathbf{f}_{\mathrm{tc}}[]$, $\vartheta \mathbf{f}_{\mathrm{lc}}[]$, $\vartheta \mathbf{f}_{\mathrm{cc}}[]$, $\vartheta \mathbf{f}_{\mathrm{overall}}[]$). These derivations are then used to define directions in multi-dimensional space towards which the parameters should be changed in order to minimize the *Error* calculated in (6). After parameters are updated, the unit selection cost function is used again in unit selection algorithm, while the *Error* when using a new set of parameters is evaluated. The process repeats through several iterations until reaching the desired minimal *Error*, or until the number of iterations is exceeded.

Further, the RGD-based algorithm in Figure 7 is also proposed to represent a mechanism for estimating the effect of lossy data compression on the unit selection process, when VQ-based compressed concatenation costs are used instead of non-lossy compressed concatenation costs as in the baseline step. The shape of the unit selection cost function and the perceived *Error* generated at the end of the optimization process then represent objective measures. The two measures are used to objectively evaluate the effect of VQ-based compression on the unit selection algorithm, and on the quality of the synthesized speech. In this way, we can automatically obtain an estimation of the unit selection cost function performance, for a given data compression technique. Thus, a balance between compression, quality and performance can be reached more efficiently. Additionally, the RGD-based algorithm (Figure 7) also optimizes the unit selection cost function itself and partially mitigates the effect of lossy data compression within the TTS system. Finally, the proposed solution can be used for any speech database, or any corpus-based TTS system.

IV. RESULTS

In order to evaluate the novel unit selection optimization algorithm for corpus-based TTS systems using the RBF-based data compression technique outlined in Figure 7, several experiments were performed. We used the PLATTOS speech database with a Slovenian female voice, with up to 1,200 database sentences. In Table 1, a memory footprint for concatenation costs ***cost$_{cc}$*** for different sizes of acoustic inventories is outlined. From the table it is evident how rapidly memory footprint rises, from 1.48MB to 22.2GB, when non-lossy compression is used for concatenation costs (full precision), and when concatenation costs are represented as tuple structures. A rapid increase of entries in the tuple structure with the number of sentences and acoustic units in the acoustic inventory is also quite evident. Although a tuple structure itself represents a time-and-space-efficient solution for representing concatenation costs, and optimal lookup for corpus-based TTS system, the sizes of tuples in Table 1 show that memory footprint still represents a significant issue.

TABLE 1. The tuple representation of non-compressed concatenation costs.

Number of sentences	N acoustic units	Number of entries	Tuple size
10	2830	189011	1.482MB
50	13676	4367548	37.52MB
100	26834	17242086	148MB
200	53889	68485616	586MB
300	78881	146649692	1256MB
600	158128	592775948	4.94GB
1200	316983	2389533080	22.2 GB

TABLE 2. The tuple representation of RBFK-based compressed concatenation costs.

Number of sentences	N acoustic units	Number of entries	VQ tuple size
10	2830	189011	0.744MB
50	13676	4367548	16.68MB
100	26834	17242086	65.8MB
200	53889	68485616	261.2MB
300	78881	146649692	432MB
600	158128	592775948	2.26GB
1200	316983	2389533080	11.1 GB

In order to reduce the size of tuple structures used for representing the concatenation costs, we proposed using RBF-based k-means clustering. In the first experiment we have defined 2,000 clusters to be used to represent the values of concatenation costs in the interval [0.000-2.000]. The clustering process was completed after 20 epochs. The lossy data compression targets the number column in the tuple structure (i.e. the number column in Figure 5). The results of the new tuple structures for the same acoustic inventories are then outlined in Table 2. After the RBFK algorithm, the tuple structures' memory footprints for the same acoustic inventories, as used prior to compression in Table 1, have been reduced by around 50%. That is, from 0.74MB to 11.1GB.

The distribution of the VQ codebooks for 4 groups of acoustic units is outlined in Figure 8 for an acoustic inventory with 300 sentences. The groups are used for the following acoustic unit pairs: diphone-diphone (D-D), diphone-triphone (D-T), triphone-diphone (T-D) and triphone-triphone (T-T). In all cases, the concatenation costs are distributed

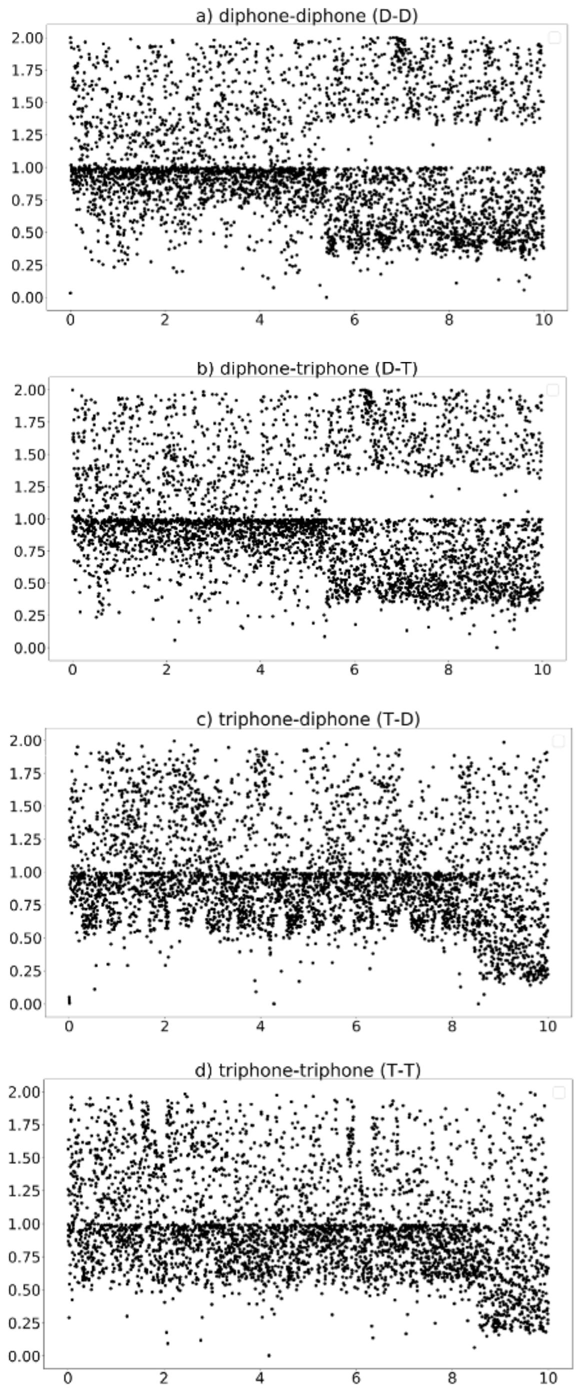


FIGURE 8. The distribution of VQ clusters for 4 groups of acoustic units (a) D-D, (b) D-T, (c) T-D, (d) T-T.

across a complete interval [0.000-2.000]. Further, clusters are denser, especially in those areas where the original concatenation costs are very close in the non-compressed concatenation costs' space. Thus, we may expect that at the lossy data compression side, the RBFK algorithm shows good performance in minimizing errors in the precision of concatenation costs.

The achieved data compression for concatenation costs is significant for the corpus-based TTS systems. However, it is very important to consider the effect of less precise information about acoustic mismatches introduced into the unit selection algorithm. Since speech quality is always above everything else, the proposed RBFK-based compression can even be pointless, if it results in significant degradation of speech quality. Moreover, another issue is how to estimate this effect on any acoustic inventory, or any corpus-based TTS system, easily and flexibly. Therefore, in the next experiment we used the proposed RGD-based optimization algorithm (in Figure 7) in order to evaluate the impact of the RBFK-based compression on the quality of synthesized speech (e.g. the *Error* parameter).

Firstly, we statistically estimated the RBFK-based compression errors by considering all concatenation costs in the acoustic inventory for 300 database sentences. Each value was calculated as a difference between the original value and the assigned cluster's value. The results are outlined in Table 3.

Table 3 also compares the compression error when the number of clusters is doubled (i.e. for 2000 and 4000 clusters). The number of selected clusters for lossy data compression is a key player in searching for the best "tuple size/precision" ratio that can additionally be analyzed easily and flexibly through the proposed algorithm in Figure 7. In Table 3, we can conclude that (not surprisingly) the sum of VQ-based errors is significantly smaller for a double number of clusters, while absolute differences regarding minimum or maximum values are not noticeable. In order to evaluate precisely how much could be gained with more clusters within a corpus-based TTS system, further experiments have to be performed.

Secondly, as outlined in Figure 9 and Table 3, the errors generated by the RBFK-based data compression are in the case of 2,000 clusters roughly 4%, and in the case of 4,000 clusters, 3%. In both cases the memory footprint of

TABLE 3. The lossy data compression errors, when using 2000/4000 clusters for the task on 300 sentences.

N clusters	N concatenation costs	MIN/MAX	Mean	Sum
2000	592775948	0.0/0.041629431615	0.0007145633	423575.9751
4000	592775948	0.0/0.030292526844	0.0002199206	130363.664

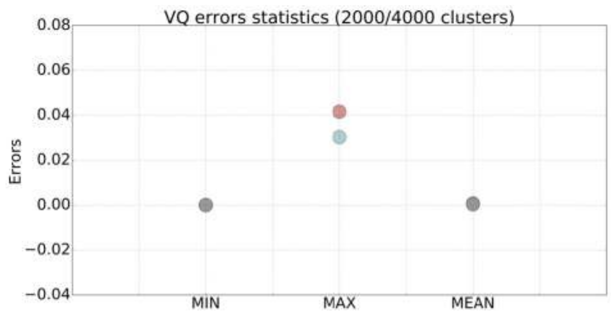


FIGURE 9. Comparing the lossy data compression errors, when using 2000 (red), or 4000 clusters (blue).

the corresponding tuple structures is reduced up to 50%. The error rate on its own, however, does not clarify how the errors will degrade the quality of speech. It only shows that in original sentence another acoustic unit was selected. As a result, the RGD-based unit selection optimization algorithm, with objective measure, is also carried out in the evaluation phase. It is performed in the following steps:

1. the unit selection algorithm on the subset of 10 database sentences is performed by utilizing the unit selection cost function in (1), and by using an acoustic inventory for 300 sentences,
2. for each sentence in the subset partial derivations over all parameters utilized in the unit selection cost function are calculated, in order to compute a proper search direction in the multi-dimensional search space as follows:

$$d_k = -\nabla f\ (x_k)$$

3. step t_k via *backtracking line search* procedure for each parameter k is then defined,
4. parameters are updated, where $\theta_k \in (0, 1)$ is randomly defined, as follows:

$$x_{k+1} = x_k + \theta_k \cdot t_k \cdot d_k$$

5. a test is performed for stopping the optimization process, where *Error* is calculated between the number of correctly selected acoustic units (diphones/ triphones) N, and the number of all selected acoustic units M in the subset of database sentences.

The algorithm was first performed with an acoustic inventory for 300 sentences, when utilizing non-lossy data compression on concatenation costs (see Figure 10 (a)). Next, the algorithm was performed on the same acoustic inventory, but utilizing lossy data compression with 2,000 clusters on concatenation costs (see Figure 10 (b)). And thirdly, the algorithm was performed on the same acoustic inventory, but utilizing lossy data compression with 4,000 clusters on concatenation costs (see Figure 10 (c)). Further, in all three cases, the unit selection optimization algorithm started with the unit selection cost function using a set of the same baseline pre-defined parameters (i.e. with the same shape). The objective measure, after performing an RGD-based unit selection optimization algorithm in each case, can tell us about the performance in selecting the most perceptually suitable acoustic units in a given subset of sentences. The results are outlined in Figure 10. The x-axis represents the number of iterations, while the y-axis represents the objective measure, i.e. *Error* in (6) that represents the ratio of correctly selected acoustic in the subset of sentences being evaluated.

Figure 10 shows that the unit selection algorithm's performance is gradually improving at each iteration in all cases. Thus, the *Error* for the subset of sentences in (a)-(c) decreases and, after 320 iterations, stabilizes at 11.901% for non-lossy compressed set, while at 12.921% for both lossy compressed sets. From these experiments we can conclude that the proposed lossy data compression of concatenation

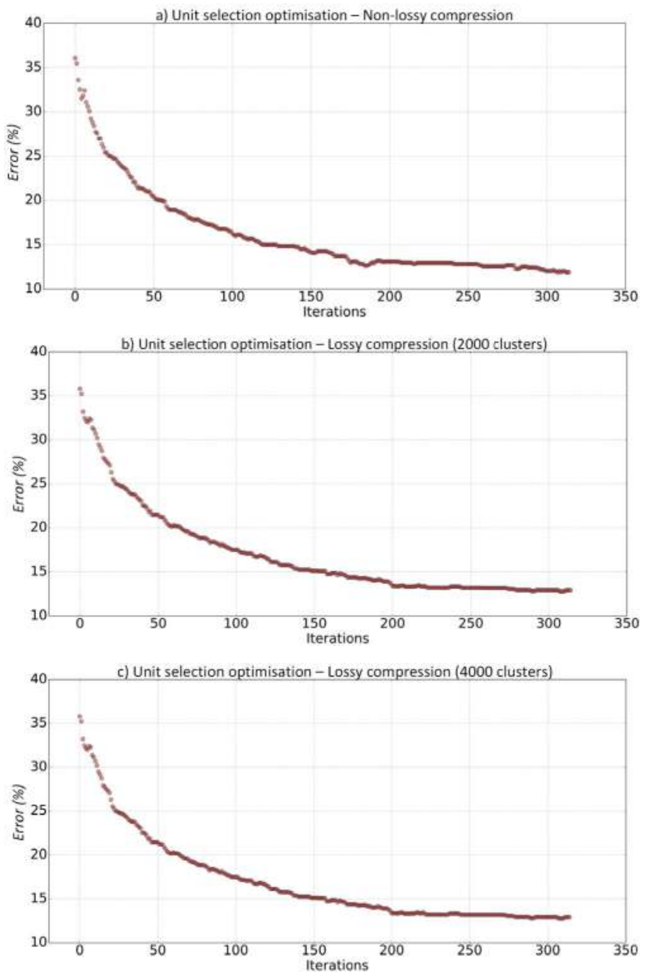


FIGURE 10. ***Error*** **used as objective measure in selecting acoustic units by unit selection algorithm during optimization, (a) – utilization of non-lossy data compression, (b) utilization of lossy data compression (2,000 clusters), (c) utilization of lossy data compression (4,000 clusters).**

costs is not without degradation of the speech quality, since final results are a bit worse than in experiment (a). However, since the *Error* is higher for only around 1%, we can say that the degradation is only negligible, if noticeable at all. Therefore, when using non-lossy data compression, or the proposed lossy data compression of concatenation costs $\boldsymbol{cost_{cc}}$, we can conclude that the proposed RBFK-based lossy data compression has negligible impact on the unit selection cost function performance. Secondly, we also observed that there is no noticeable effect on the *Error* when we use double the clusters for VQ-based lossy data compression. Therefore, 2,000 clusters can be used in the tuple structure for the same results. Further, although the statistical analysis above showed that double the clusters reduces VQ errors, the unit selection algorithm after performing RGD-based optimization proves the same performance. From this we can conclude that the proposed RGD-based optimization also minimizes the effect of VQ errors incorporated into the TTS system, when using lossy data compression of concatenation costs. This is also a beneficial conclusion, since we can use fewer clusters for the same unit selection algorithm performance.

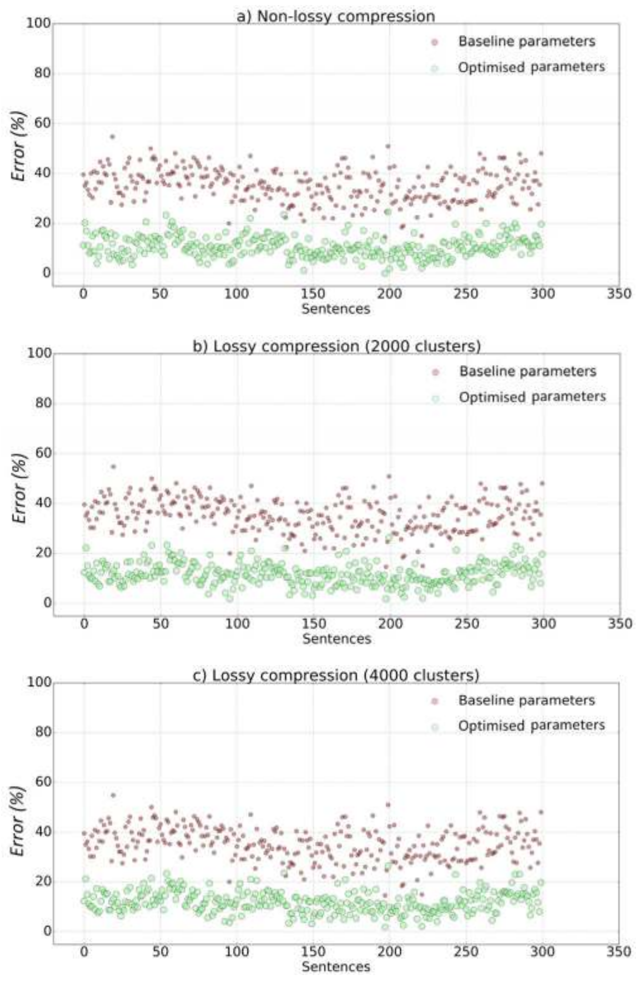


FIGURE 11. ***Error,* when using baseline non-/optimized unit selection cost function without lossy compression (a), non-/optimized unit selection cost function with RBFK compression with 2,000 clusters (b), non-/optimized unit selection cost function with RBFK compression with 4,000 clusters (c).**

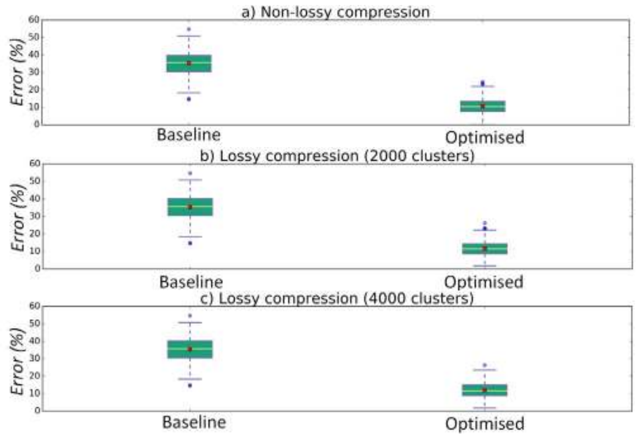


FIGURE 12. ***Error,* when using non-/optimized unit selection cost function without lossy data compression (a), RBFK compression with 2,000 clusters (b), and RBFK compression with compression with 4,000 clusters (c).**

TABLE 4. **The tuple representation of RBFK-based compressed concatenation costs.**

	Non-lossy compression	Lossy compression VQ-2000 clusters	Lossy compression VQ-4000 clusters
***Error*, non-optimised (Median value)**	35.5991	35.7143	35.7143
***Error*, optimised (Median value)**	10.4187	11.7	11.5385

In the next experiment, we have conducted a performance analysis of the unit selection algorithm on a larger set, with 300 sentences that were not included in the RGD-based optimization process. The goal in this case has been, firstly, to see how the optimized unit selection cost function is performed on a set of sentences not included in the subset for optimization, and to estimate the unit selection performance on the same task for all three cases in previous experiments in Figure 10. The results are outlined in detail in Figure 11 (a)-(c), Figure 12 (a)-(c), and Table 4. By analyzing the results, we can conclude that in the proposed unit selection optimization algorithm the VQ errors also have negligible impact on the unit selection cost function performance on unseen sentences. We can conclude, firstly, that the performance of the RBFK algorithm is very good, since comparable unit selection cost function performance results are obtained for all cases (a)-(c). Further, as outlined in Table 4 and Figure 11, for all cases (a)-(c) the *Error* rate is reduced from 36% to roughly 11% after the proposed optimization. When comparing these results, therefore, whether we are using non-lossy or VQ-based lossy data compression (2,000 or 4,000 clusters) on concatenation costs $\boldsymbol{cost_{cc}}$, we can conclude that in the unit selection algorithm VQ errors have negligible impact on the unit selection cost function performance. Therefore, advanced RBFK clustering using 2,000 clusters guarantees very good representations for the concatenation costs, and on the other hand significant reduction in the size of the tuple structure.

Moreover, as outlined in Figure 12 (a)-(c), the *Error* in selecting acoustic units per sentence in the set of 300 sentences without lossy compression ranges from [20-60%] for non-optimized unit selection cost function, while after RGD-based optimization this *Error* drops into the interval [0-20%]. Further, the boxplot in Figure 12 (a) shows that the median value for *Error* is 35.59% for the baseline unit selection cost function, and 10.41% after RGD-based optimization. The boxplots in Figure 12 (b) and (c) then show that the median value for *Error* is 35.7143% in the non-optimized case, and around 11.7% after RGD-based optimization, when the RBFK-based compression has been used. Again, deviations in *Error* between non-lossy compressed, and lossy compressed spaces, is minimal, roughly 1%.

V. CONCLUSION

The main drawback of the corpus-based TTS systems is that its speech quality and performance greatly depend on the acoustic inventory. That is to say, when the inventory is of high-quality, and covers lots of possible contexts, the synthesized speech will also be of high quality and close to natural-sounding. But, on the other hand, large acoustic inventories have huge memory footprints. For instance, in the case of the TTS system PLATTOS, storage for concatenation costs for 1,200 sentences requires 22.2 GB. Further, any significant reduction in the diversity of acoustic realizations will increase the presence of spectral discontinuities in the synthesized speech, which results in less natural-sounding speech. Thus, in this paper we have proposed a new algorithm designed to optimize unit selection cost function in corpus-based TTS systems. We have also provided an objective measure to estimate automatically the degradation in speech quality due to lossy compression and, even more importantly, to minimize and mitigate the effects of the optimization. In particular, rather than targeting the acoustic inventory, we proposed to further compress the concatenation costs' space. We specifically aimed to compress the floating-point space used to represent the costs. For this purpose, we deployed a VQ-based compression implemented via an RBF network, with a k-means algorithm used for clustering. The comprehensive experiments show that tuple structures' footprints, for the same acoustic inventories as used prior to compression, were reduced by up to 50%, and the *Error* rates introduced by the compression were insignificant, roughly a 1% increase. The distribution of clusters generated by the RBFK algorithm implies that the actual impact of the errors, however, is not noticeable. Thus, the clusters obtained are more compact in those areas where the concatenation costs are very close in the non-compressed concatenation costs' space.

Further, we also investigated the effect of the proposed RGD-based optimization algorithm when using lossy data compression on concatenation costs. It showed that the *Error* for the selected subset of sentences decreases, and after 320 iterations stabilizes at 11.901% for the non-lossy compressed set, and at 12.921% for the lossy compressed sets. Thus, the impact of the RBFK-based compression is successfully mitigated to 1%. From this we can conclude that the proposed RGD-based optimization minimizes the degrading effect of VQ errors introduced by the lossy compression of concatenation cost space. Additionally, when comparing results between concatenation cost space with 2,000 and 4,000 clusters, we observed that the difference in performance of the unit selection cost function is only minimal. This is highly beneficial, since we can use fewer clusters for the same unit selection algorithm performance. The outlined trends were also confirmed on large sets with 300 sentences not involved in the proposed optimization. After the optimization, the proposed lossy compression introduces a minimal, roughly 1%, increase in *Error*.

To sum up, the size of tuples used for concatenation costs directly affects the efficiency and the performance of the unit selection algorithm in the corpus-based TTS system PLATTOS; thus, increasing the speech quality with more acoustic units and with more acoustic realizations can be very costly for unit selection algorithm performance and available memory resources. The algorithm proposed in this paper significantly reduces the memory footprint with a proven minimal degrade in the quality of the synthesized speech. In contrast to similar optimization approaches (e.g. those involving clustering), the decrease in accuracy is negligible. However, the size of the part in the tuple structure used for storing hash numbers still represents a significant computational problem, especially if the algorithm is to be used on embedded solutions. The tuple structure for 1,200 sentences after using RBFK lossy compression still consumes 11.2 GB. Thus, in the future we plan to investigate using DNNs architectures and multi-layered long short-term memory recurrent neural networks, in order to optimize even further the representation of the concatenation costs.

REFERENCES

[1] S. Desai, A. W. Black, B. Yegnanarayana, and K. Prahallad, "Spectral mapping using artificial neural networks for voice conversion," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 18, no. 5, pp. 954–964, Jul. 2010.

[2] X. Miao, X. Zhang, M. Sun, C. Zheng, and T. Cao, "A BLSTM and wavenet based voice conversion method with waveform collapse suppression by post-processing," *IEEE Access*, vol. 7, pp. 54321–54329, 2019.

[3] T. Capes, P. Coles, A. Conkie, L. Golipour, A. Hadjitarkhani, Q. Hu, N. Huddleston, M. Hunt, J. Li, M. Neeracher, and K. Prahallad, "Siri on-device deep learning-guided unit selection text-to-speech system," in *Proc. INTERSPEECH*, Aug. 2017, pp. 4011–4015.

[4] M. Rojc, I. Mlakar, and Z. Kačič, "The TTS-driven affective embodied conversational agent EVA, based on a novel conversational-behavior generation algorithm," *Eng. Appl. Artif. Intell.*, vol. 57, pp. 80–104, Jan. 2017.

[5] M. Chu, H. Peng, H.-Y. Yang, and E. Chang, "Selecting non-uniform units from a very large corpus for concatenative speech synthesizer," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 2, May 2001, pp. 785–788.

[6] Y. Zhao, S. Takaki, H.-T. Luong, J. Yamagishi, D. Saito, and N. Minematsu, "Wasserstein GAN and waveform loss-based acoustic model training for multi-speaker text-to-speech synthesis systems using a wavenet vocoder," *IEEE Access*, vol. 6, pp. 60478–60488, 2018.

[7] A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2962–2970.

[8] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards end-to-end speech synthesis," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, 2017, pp. 4006–4010.

[9] Y. Yasuda, X. Wang, S. Takaki, and J. Yamagishi, "Investigation of enhanced Tacotron text-to-speech synthesis systems with self-attention for pitch accent language," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 6905–6909.

[10] X. Zhu, Y. Zhang, S. Yang, L. Xue, and L. Xie, "Pre-alignment guided attention for improving training efficiency and model stability in end-to-end speech synthesis," *IEEE Access*, vol. 7, pp. 65955–65964, 2019.

[11] R. Barra-Chicote, J. Yamagishi, S. King, J. M. Montero, and J. Macias-Guarasa, "Analysis of statistical parametric and unit selection speech synthesis systems applied to emotional speech," *Speech Commun.*, vol. 52, no. 5, pp. 394–404, May 2010.

[12] A. Rosenberg and B. Ramabhadran, "Bias and statistical significance in evaluating speech synthesis with mean opinion scores," in *Proc. INTERSPEECH*, 2017, pp. 3976–3980,.

[13] M. Gruber, J. Matousek, D. Tihelka, and Z. Hanzlicek, "Reducing footprint of unit selection TTS system by removing linguistic segments with rarely selected units," in *Proc. 12th Int. Conf. Signal Process. (ICSP)*, Oct. 2014, pp. 494–499.

[14] H. Lu, W. Zhang, X. Shao, Q. Zhou, W. Lei, H. Zhou, and A. Breen, "Pruning redundant synthesis units based on static and delta unit appearance frequency," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, Sep. 2015, pp. 269–273.

[15] J. Nurminen, H. Silén, and M. Gabbouj, "Speaker-specific retraining for enhanced compression of unit selection text-to-speech databases," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, 2013, pp. 388–391.

[16] P. Sharma, V. Abrol, and A. K. Sao, "Reducing footprint of unit selection based text-to-speech system using compressed sensing and sparse representation," *Comput. Speech Lang.*, vol. 52, pp. 191–208, Nov. 2018.

[17] M. K. Jayesh and C. S. Ramalingam, "A one-dimensional search method with stable 1-norm solution for linear prediction," *J. Acoust. Soc. Amer.*, vol. 142, no. 2, pp. 170–176, Aug. 2017.

[18] D. Giacobello, M. G. Christensen, T. L. Jensen, M. N. Murthi, S. H. Jensen, and M. Moonen, "Stable 1-norm error minimization based linear predictors for speech modeling," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 22, no. 5, pp. 912–922, May 2014.

[19] F. Hinterleitner, B. Weiss, and S. Möller, "Influence of corpus size and content on the perceptual quality of a unit selection MaryTTS voice," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Dec. 2016, pp. 680–685.

[20] H. Peng, Y. Zhao, and M. Chu, "Perpetually optimizing the cost function for unit selection in a TTS system with one single run of MOS evaluation," in *Proc. 7th Int. Conf. Spoken Lang. Process.*, 2002, pp. 2613–2616.

[21] F. Alías, L. Formiga, and X. Llorá, "Efficient and reliable perceptual weight tuning for unit-selection text-to-speech synthesis based on active interactive genetic algorithms: A proof-of-concept," *Speech Commun.*, vol. 53, no. 5, pp. 786–800, May/Jun. 2011.

[22] N. P. Narendra and K. S. Rao, "Optimal weight tuning method for unit selection cost functions in syllable based text-to-speech synthesis," *Appl. Soft Comput.*, vol. 13, no. 2, pp. 773–781, Feb. 2013.

[23] M. Pobar, S. Martincic-Ipsic, and I. Ipsic, "Optimization of cost function weights for unit selection speech synthesis using speech recognition," *Neural Netw. World*, vol. 22, no. 5, pp. 429–441, Sep. 2012.

[24] S. K. Gill and P. Singh, "Discontinuity removal in concatenative synthesised speech," *Int. J. Eng. Technol. Sci. Res.*, vol. 4, no. 4, pp. 415–419, 2017.

[25] D. Lolive, P. Alain, N. Barbot, J. Chevelu, G. Lecorvé, C. Simon, and M. Tahon, "The IRISA text-to-speech system for the blizzard challenge 2017," in *Proc. Blizzard Challenge*, Stockholm, Sweden, Aug. 2017, pp. 1–7.

[26] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 2154–2162.

[27] L. Lee, E. Parisotto, D. S. Chaplot, and R. Salakhutdinov, "LSTM iteration networks: An exploration of differentiable path finding," in *Proc. ICLR*, 2018, pp. 1–9.

[28] M. L. Padmesh and P. S. Kumar, "Implementation of Viterbi coder for text to speech synthesis," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, Dec. 2015, pp. 1–5.

[29] V. Wan, Y. Agiomyrgiannakis, H. Silen, and J. Vit, "Google's next-generation real-time unit-selection synthesizer usingsequence-to-sequence LSTM-based autoencoders," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, Aug. 2017, pp. 1143–1147.

[30] Z. Jin, A. Finkelstein, S. DiVerdi, J. Lu, and G. J. Mysore, "Cute: A concatenative method for voice conversion using exemplar-based unit selection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 5660–5664.

[31] Y. C. Lim, T. S. Tan, S. H. S. Salleh, and D. K. Ling, "Application of genetic algorithm in unit selection for malay speech synthesis system," *Expert Syst. Appl.*, vol. 39, no. 5, pp. 5376–5383, Apr. 2012.

[32] D. Guennec and D. Lolive, "Unit selection cost function exploration using an a based text-to-speech system," in *Proc. Int. Conf. Text, Speech, Dialogue (TSD)*, 2014, pp. 432–440.

[33] S. Taylor, T. Kim, Y. Yue, M. Mahler, J. Krahe, A. G. Rodriguez, J. Hodgins, and I. Matthews, "A deep learning approach for generalized speech animation," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017, Art. no. 93.

[34] P. Venkataram, S. Ghosal, and B. P. V. Kumar, "Neural network based optimal routing algorithm for communication networks," *Neural Netw.*, vol. 15, no. 10, pp. 1289–1298, Dec. 2002.

[35] M. T. Martín-Valdivia, L. A. Ureña-López, and M. García-Vega, "The learning vector quantization algorithm applied to automatic text classification tasks," *Neural Netw.*, vol. 20, no. 6, pp. 748–756, Aug. 2007.

[36] M. Capó, A. Pérez, and J. A. Lozano, "An efficient approximation to the K-means clustering for massive data," *Knowl. Based Syst.*, vol. 117, pp. 56–69, Feb. 2017.

[37] X. Driancourt, L. Bottou, and P. Gallinari, "Learning vector quantization, multi layer perceptron and dynamic programming: Comparison and cooperation," in *Proc. Seattle Int. Joint Conf. Neural Netw.*, vol. 2, Jul. 1991, pp. 815–819.

[38] T. Kohonen, "Learning vector quantization," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 1995, pp. 537–540.

[39] T. Villmann, A. Bohnsack, and M. Kaden, "Can learning vector quantization be an alternative to SVM and deep learning?-Recent trends and advanced variants of learning vector quantization for classification learning," *J. Artif. Intell. Soft Comput. Res.*, vol. 7, no. 1, pp. 65–81, Jan. 2017.

[40] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Netw.*, vol. 14, nos. 4–5, pp. 439–458, 2001.

[41] Z. Wang, Y. He, and M. Jiang, "A comparison among three neural networks for text classification," in *Proc. 8th Int. Conf. Signal Process.*, vol. 3, Nov. 2006, pp. 1–4.

[42] M. Mohri, P. J. Moreno, and E. Weinstein, "Efficient and robust music identification with weighted finite-state transducers," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 1, pp. 197–207, Jan. 2010.

[43] M. Rojc and Z. Kačič, "Gradient-descent based unit-selection optimization algorithm used for corpus-based text-to-speech synthesis," *Appl. Artif. Intell.*, vol. 25, no. 7, pp. 635–668, Aug. 2011.

[44] W. Jiang, P. Liu, and F. Wen, "An improved vector quantization method using deep neural network," *AEU-Int. J. Electron. Commun.*, vol. 72, pp. 178–183, 2017.

[45] A. Vasuki and P. T. Vanathi, "A review of vector quantization techniques," *IEEE Potentials*, vol. 25, no. 4, pp. 39–47, Jul./Aug. 2006.

[46] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[47] N. Andrei, "A new gradient descent method for unconstrained optimization," ICI Tech. Rep., Apr. 2004. [Online]. Available: https://camo.ici.ro/neculai/newstep.pdf

[48] C.-Y. Lin, L.-C. Wang, and K.-H. Tsai, "Hybrid real-time matrix factorization for implicit feedback recommendation systems," *IEEE Access*, vol. 6, pp. 21369–21380, 2018.

MATEJ ROJC is currently an Associate Professor with the Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests include text-to-speech synthesis, automatic speech recognition, speech-to-speech translation, non-verbal speech processing, prosodic information modelling, finite-state machines, machine learning algorithms, coverbal synchrony, and advanced multimodal interfaces. He is a Co-editor of the book *Coverbal Synchrony in Human-Machine Interaction,* and coauthor of the book *An Expressive Conversational-Behavior Generation Model for Advanced Interaction within Multimodal User Interfaces.*

IZIDOR MLAKAR is currently a Researcher with the Faculty of Electrical Engineering and Computer Science, University of Maribor. He is involving in and leading a research project in conversational models for assisted environments, funded by the EU and the Ministry of Education, Science, and Sport of Slovenia, entitled SAIAL. His research interests include embodied conversational agents, natural language processing and generation, multimodal conversational intelligence, kinesics and sentics, machine learning, advanced multimodal interfaces, the Internet of Things, and smart homes and ambient assisted living.