VERY FAST UNIT SELECTION USING VITERBI SEARCH WITH ZERO-CONCATENATION-COST CHAINS

Jiří Kala and Jindřich Matoušek

Dept. of Cybernetics, Faculty of Applied Sciences, University of West Bohemia, Czech Rep.

ABSTRACT

This paper introduces a very fast heuristic search algorithm for unit-selection speech synthesis. The algorithm modifies commonly used Viterbi search framework by introducing zero-concatenation-cost (ZCC) chains of unit candidates that immediately neighbored in a source speech corpus. ZCC chains are preferred as they represent perfect speech segment concatenations (so there is no need to compute concatenation costs inside the chains) unless a so-called target specification is violated. The number of ZCC chains is reduced based on statistics calculated upon the synthesis of a large number of utterances. ZCC chains are then combined with single unit candidates to fill possible gaps in the sequence of candidates. The proposed method reduces the computational load of a unit selection system up to hundreds of times. According to listening tests, the quality of synthetic speech was not deteriorated.

Index Terms— speech synthesis, unit selection, Viterbi algorithm, non-uniform units, zero cost concatenation

1. INTRODUCTION

Unit-selection text-to-speech (TTS) systems are known for their ability to produce nearly natural-sounding synthetic speech. However, to achieve such high quality output, these systems utilize very large speech corpora that aim to cover as many phonetic and prosodic contexts as possible. As a result, the corpora contain a very large number of candidates of each speech unit (typically diphones, context-dependent phones or other phone-like units). In practical applications, the corpora often include up to tens of hours of speech. Taking into account the basic principle of unit selection—for each unit to dynamically select the best unit candidate from the many which are available—and the high number of candidates of each speech unit, computational load related to the selection of optimal unit candidates for a given utterance could be enormous. High computational demand could be crucial, especially in server solutions in which many parallel requests must be synthesized in real time or in less powerful devices like smartphones or tablets. Running a unit-selection TTS system on such devices could be very challenging, especially when the quality of the synthetic speech output should be preserved.

To select the optimal sequence of unit candidates, the Viterbi algorithm is often used [1] (see Sec. 2 for a short description). Various approaches were proposed to speed up the selection process. As it was shown that the computation of a so-called *concatenation cost* (see Sec. 2) is the most computationally demanding part of unit selection [2], the optimizations mostly focused on this particular issue. Some of them cache computations by synthesizing a large portion of text offline, and, in run-time, the cached values are used instead of being computed (see e.g. [2, 3]). Another approach is to reduce the number of unit candidates by removing some of them offline [4–7], or during run-time after pre-selecting them using e.g. clustering [8], statistics about unit candidates occurrences [9], or based on $F0$ discontinuities [10]. Various modifications of the baseline Viterbi algorithm were also proposed e.g. in [11] and further elaborated on in [12].

Another group of methods prefer to select unit candidates immediately neighboring in source speech corpus utterances. The motivation is clear—such candidates concatenate perfectly, and, unless they violate so-called *target specification* (see Section 2), they can be preferred during unit selection, resulting in a selection of longer (*non-uniform*) units [13–16].

In this paper, a new method for speeding up the unit selection process is presented. The method takes advantage of the non-uniform unit scheme mentioned above (here, the sequence of unit candidates immediately neighboring in a source speech corpus will be denoted as *zero-concatenation-cost chain*), but instead of improving the quality of synthetic speech, focus is primarily given to the reduction of computational demand. Unlike [16] unit chains are not tied to any linguistic structures like words, syllables or phrases.

2. VITERBI SEARCH – BASELINE ALGORITHM

In the context of unit selection, the Viterbi search (hereinafter VITBASE) utilizes two cost functions—the *target cost* $C^t(t_k, u_k)$, which describes how well or poorly a unit candidate u_k meets target specification t_k (i.e. phonetic and prosodic contexts of neighboring units, various positional aspects in the utterance, etc.), and the *concatenation cost* $C^c(u_k, u_{k+1})$, which expresses how well or poorly two potentially joinable unit candidates u_k and u_{k+1} join together (mainly with respect to spectral continuity). The resulting cost function $C(t_1^K, u_1^K)$ then combines both cost functions and expresses the total (cumulative) cost of a sequence of K candidates

$$C(t_1^K, u_1^K) = \sum_{k=1}^{K} C^t(t_k, u_k) + \sum_{k=1}^{K-1} C^c(u_k, u_{k+1}). \quad (1)$$

The goal of the search is to find an optimal sequence of unit candidates u^{*K}_1 minimizing the total cost function $C(t_1^K, u_1^K)$ [1, 17]. To do that, the Viterbi algorithm goes through a network of all units candidates u_1^K assigning each i-th candidate of k-th unit, $u_k(i)$, $(k = 1, \ldots, K)$ a best preceding candidate $u_k^*(i)$ of $(k-1)$-th unit

$$u_k^*(i) = \underset{j=1,\ldots,N(k-1)}{\operatorname{argmin}} \{C^*(u_{k-1}(j)) + C^c(u_{k-1}(j), u_k(i))\} \quad (2)$$

where $N(k)$ is a number of candidates of k-th unit and C^* is a best

Support for this work was provided by the Technology Agency TA CR, project No. TA01030476, and by the University of West Bohemia, project No. SGS-2013-032.

cumulative cost defined as

$$C^*(u_k(i)) = \begin{cases} C^t(u_k(i)) & k = 1 \\ C^t(u_k(i)) + C^*(u_k^*(i)) + C^c(u_k^*(i), u_k(i)) & k > 1 \end{cases} \quad (3)$$

The resulting sequence of selected unit candidates u^{*k}_1 can then be found by backtracking the network of all unit candidates starting at the very last unit u_K.

The sequence found by this baseline algorithm is globally optimal with respect to the total (cumulative) cost. On the other hand, when no modifications of the baseline algorithm are made, it is very computationally demanding to find u^{*k}_1. For instance, in a system with more than 650k units candidates (approx. 18 hours of speech) thousands of candidates per unit can occur. Synthesizing an utterance with the length of 30-50 units will then result in up to tens of millions of C^c computations.

In our previous work [12], the computational load of VITBASE was reduced by introducing a flexible *pruning scheme* in which the number of unit candidates was reduced based on different tunable criteria (denoted as VITPRUNED hereinafter). The selection process was then approx. nine times faster (for a "conservative" pruning setting), but the finding of globally optimized unit candidates sequence has not been guaranteed any more.

3. ZERO-COST-CONCATENATION VITERBI ALGORITHM

3.1. Zero-cost-concatenation chains

By synthesizing randomly selected 10k utterances we observed that approx. 95% of used unit candidates were part of a *zero-cost-concatenation (ZCC) chain* (see Fig. 1). We define a ZCC chain as a sequence of at least two speech segments (unit candidates) that immediately neighbored in a source speech corpus.

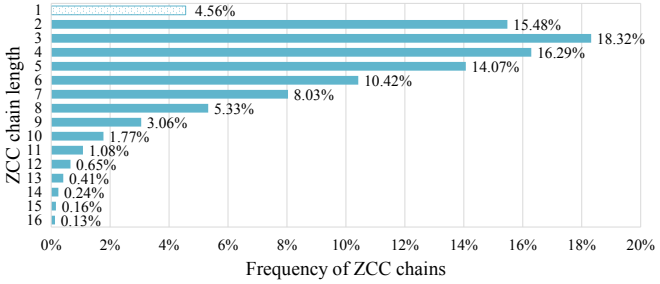

Fig. 1. ZCC chains of different lengths and their frequency in synthesized utterances. The white bar represents the frequency of single speech segment occurrences.

Considering that an important attribute of ZCC chains–C^c is zero for all unit candidates within a ZCC chain, there is no need to compute C^c at all. Thus, the cumulative cost of a ZCC chain can be computed, simplifying the Eq. 4, as a sum of target costs C^t

$$_{\text{zcc}}C(t_m^M, u_m^M) = \sum_{k=m}^{M} C^t(t_k, u_k) \quad (4)$$

where m is a starting position and M an ending position of candidates within a whole synthesized utterance. ZCC Viterbi algorithm (ZCCVIT) described further in the text then aims to minimize the total cumulative cost C by preferring to select ZCC chains. As a result, less C^c computations are required, and subsequently the process of unit selection is sped up.

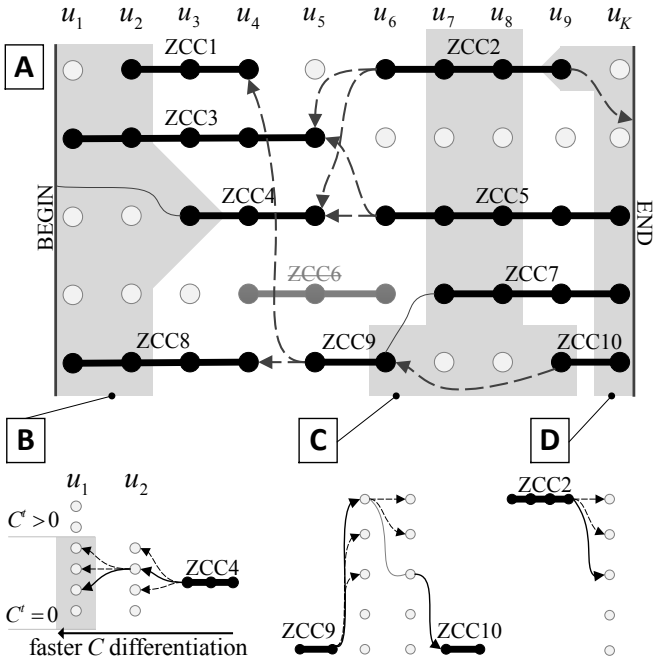

Fig. 2. Scheme of ZCCVIT algorithm.

3.2. Algorithm description

The ZCCVIT algorithm is basically similar to VITBASE. The difference is that the nodes of the network are not single candidates but all are ZCC chains which can be set up from the candidates. ZCC chains are sorted according to their starting positions in an utterance, and then for each ZCC chain, similarly as in VITBASE, a best preceding ZCC chain minimizing the cumulative cost C^* is searched for. Since the strategy is to select ZCC chains as much as possible, the best preceding ZCC chain is selected only from the closest ZCC chains. Again, an optimal sequence of candidates is then found by backtracking the whole network. The scheme of the ZCCVIT algorithm is shown in Fig. 2.

The procedure of searching for the optimal path through the network of K units with each unit u_k having $N(k)$ candidates (i.e. the selection of optimal sequence of candidates $u_k(i)$, $k = 1 \ldots K$ and $i = 1 \ldots N(k)$, for a given utterance) using the ZCCVIT algorithm could be described as follows:

1: For each candidate $u_k(i)$ compute the target cost $C^t(u_k(i))$.
2: Search for all ZCC chains in the network and manage them as described in Sec. 3.3. Denote the set of resulting ZCC chains as $zccSet$.
3: Sort ZCC chains in $zccSet$ according to their starting positions in the network.
4: Set $bestCumCost = \infty$ {defines minimum cumulative cost computed so far through the whole network}.
5: Set $bestPath = NONE$ {defines the best path found so far}.
6: **for** $zccWork$ **in** $zccSet$ **do**
7: **if** no ZCC chain precedes $zccWork$ **then**
8: **if** distance of $zccWork$ from the beginning of the network is $LIMIT^{\text{beg}}$ at the most **then**
9: Search for the path from the beginning of the network to the start of $zccWork$. Use BeFS search in a reverse order as described in Sec. 3.4 and shown in Fig. 2.B. Store the resulting path and its cumulative cost, and associate

these with $zccWork$.
10: **else**
11: Remove $zccWork$ from $zccSet$ and do not process this chain any more (see ZCC6 in Fig. 2.A).
12: **end if**
13: **else**
14: Find the closest ZCC chain(s) that precede $zccWork$ (their distance from $zccWork$ is the same and minimal). Search for paths between these preceding ZCC chains and $zccWork$ using the BeFS algorithm described in Sec. 3.4 (see Fig. 2.C).
15: The best preceding ZCC chain is the one with the minimum cumulative cost. Store the path to this ZCC chain and its cumulative cost, and associate these with $zccWork$.
16: **end if**
17: **if** the distance of $zccWork$ from the end of the network is $LIMIT^{\text{end}}$ at the maximum **then**
18: Search for the path from the end of $zccWork$ to the end of the network. Use the BeFS search as shown in Fig. 2.D.
19: Backtrack the network from $zccWork$ to the beginning of the network and together with the path found in Step 18 create the full path $fullPath$ through the whole network. Set $fullCumCost$ to be the cumulative cost of the created $fullPath$.
20: **if** $fullCumCost < bestCumCost$ **then**
21: $bestCumCost = fullCumCost$
22: $bestPath = fullPath$
23: **end if**
24: **end if**
25: **end for**
26: The optimal path through the network of all candidates (i.e. the optimal sequence of candidates for the given utterance) is stored in $bestPath$.

3.3. ZCC chain management

In order to make the ZCC chain framework more flexible, and to prevent the algorithm from selecting ZCC chains at the expense of any higher values of target costs C^t (especially at the ends of chains), a set of ZCC chains were supplemented with all the sub-chains of all ZCC chains. In this way, the number of points to concatenate ZCC chains increased, but, on the other hand, the number of ZCC chains significantly increased too, slowing down the selection process. Thus, a way of reducing the number of ZCC chains by identifying such chains that have no chance to be a part of the optimal sequence of candidates was searched for. Having analyzed 10k synthetic utterances (of a male voice), we found that 99.98% of ZCC chains that were a part of optimal sequences had a maximum target cost ${}_{\max}C^t \leqq 0.33$ and an average target cost ${}_{\text{avg}}C^t \leqq 0.30$ (see Fig. 3). Based on these findings, the pruning of ZCC chains according to ${}_{\max}C^t$ and ${}_{\text{avg}}C^t$ was incorporated into the ZCCVIT algorithm[1].

3.4. Search for candidates between ZCC chains

After ZCC chains were found within a synthesized utterance, any possible gaps between the ZCC chains had to be identified. A gap is formed by a sub-network of all single unit candidates from the original network in an area where ZCC chains are not connected directly. The first and last nodes of the sub-network are the last candidates of the previous ZCC chain and the first candidate of the next

[1]The exact values of ${}_{\max}C^t$ and ${}_{\text{avg}}C^t$ could be different for different voices. For our female voice we found ${}_{\max}C^t = 0.35$ and ${}_{\text{avg}}C^t = 0.33$.

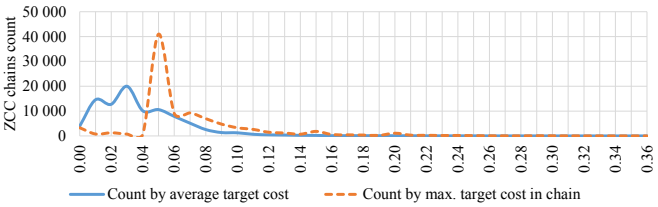

Fig. 3. Counts of ZCC chains (that were part of optimal sequences of candidates) according to average/maximum target costs C^t of the candidates in the chains.

ZCC chain, respectively. To find the optimal sequence of candidates within these gaps (and to connect ZCC chains), the VITBASE algorithm would not be a good choice because for any multiple gaps in a single utterance, the same number of C^c (or even more) had to be computed (see Fig. 4.A). Instead, the *bounded-depth best-first search* (BeFS) algorithm was applied. BeFS is very effective when there is only one candidate at the beginning of a sub-network (this is the case at the end of each ZCC chain as shown in Fig. 2.C and 2.D) because any promising candidate sequences can be determined quickly. The problem of searching for an optimal sequence starting at the beginning of the network (in which $C^t = 0$ for many candidates) and ending with the first candidate of the first ZCC chain (illustrated in Fig. 4.B) was resolved by reversing the sub-network as shown in Fig. 2.B. Experiments showed that this reversal lead to approx. $148\times$ less C^c computations.

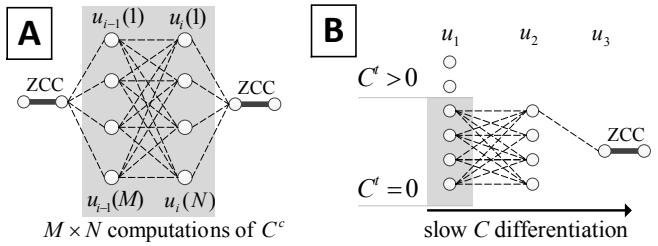

Fig. 4. Illustration of a problem of using VITBASE to find single candidates in gaps between ZCC chains (A) and of a problem of finding a path with a higher number of starting candidates (B).

Another speed-up was achieved by caching C^c during BeFS in a similar way as described in [2] or [3].

4. EVALUATION

4.1. Quality measures

To evaluate the proposed ZCCVIT algorithm, two Czech voices from the ARTIC unit-selection TTS system [18], male and female, were used. Unit inventories of both voices are of similar size (≈18 hours of news-style speech), as the same text prompts were utilized to record the speech corpus of each voice [19, 20]. To compare the ZCCVIT algorithm both to the baseline VITBASE and to the VITPRUNED algorithms, 20 randomly selected news-style utterances (not included in the source speech corpus; their average length being 39 units / 7 words) were synthesized using the three different versions of the Viterbi algorithm and compared to each other with respect to the following measures.

Speed-up rate $S = \frac{1}{n}\sum_{i=1}^{n} \frac{{}_{\text{base}}N^c(i)}{N^c(i)}$ denotes how much the algorithm in question is faster than VITBASE in terms of the num-

Table 1. Comparison of different versions of the Viterbi algorithm for the male voice

Algorithm	S	CD	Q
VITBASE	1.00	0.2939	0.00
VITPRUNED$^{\mathrm{cns}}$	8.35	0.2939	0.00
ZCCVIT$^{\mathrm{opt}}$	556.54	0.2851	14.27
VITPRUNED$^{\mathrm{opt}}$	563.74	0.3889	62.27
ZCCVIT$^{\mathrm{fst}}$	4917.20	0.2722	60.88
VITPRUNED$^{\mathrm{fst}}$	4863.00	0.4035	191.68

ber of computations of concatenation costs C^c ($n = 20$, i is the index of a testing utterance, $N^c(i)$ is the number of C^c computations in the algorithm in question, and ${}_{\mathrm{base}}N^c(i)$ is the number of C^c computations in the reference VITBASE algorithm). We consider such a measure to be sufficient because the concatenation cost computations cover 89.14% of all synthesis-time computations.

Quality deterioration $Q = \frac{1}{n}\sum_{i=1}^{n}\frac{C(i)-{}_{\mathrm{base}}C(i)}{{}_{\mathrm{base}}C(i)} \cdot 1000$ [‰] denotes an increase of the cumulative cost $C(i)$ of the algorithm in question compared to the globally minimal cumulative cost ${}_{\mathrm{base}}C(i)$ of the VITBASE algorithm. The higher is the value of Q, the lower (mathematical) quality is observed (supposing that the cumulative cost is a good estimate of the quality of synthesized speech).

Concatenation density $CD = \frac{1}{n}\sum_{i=1}^{n}\frac{N^d(i)}{N^a(i)}$ indicates how much is a synthesized utterance i "fragmented" by comparing the number of concatenations of the original non-neighboring unit candidates $N^d(i)$ to the number of all concatenations $N^a(i)$ [7]. The higher is the value of CD, the more fragmented the utterance is; thus, the higher is the probability that some artifacts occur at concatenation points deteriorating the quality of speech output. The value of $CD = 0$ means that the synthesized utterance was a part of the source speech corpus.

Listening tests are a means of subjective evaluation of synthetic speech. Pairwise preference listening tests were carried out to compare the overall quality of speech synthesized by the different versions of the Viterbi algorithm. 12 listeners (both TTS experts and inexperienced listeners) took part in the tests. Each pair of synthetic utterances A and B were compared on a 3-point scale (A is better than B, A sounds same as B, A is worse than B).

4.2. Results and discussion

As described in Sec. 3.2, several parameters are used to drive the ZCCVIT algorithm. The optimal values of the parameters were found using a grid search with respect to quality deterioration measure Q. The best value of $Q = 14.27$‰ was achieved with parameters $\mathrm{LIMIT}^{\mathrm{beg}} = 3$, $\mathrm{LIMIT}^{\mathrm{end}} = 3$, and the corresponding speed-up rate was $S = 556.55$. Comparing optimal sequences of unit candidates selected by ZCCVIT and VITBASE algorithms, we found that 7 of the 20 testing utterances consisted of the same candidates and the rest of the utterances differed only in a few candidates. The results are summarized in Table 1 and 2. VITPRUNED$^{\mathrm{cns}}$ stands for a "conservative" pruning scheme described in [12] (in our testing set, this algorithm yielded the same optimal sequence of candidates as VITBASE), ZCCVIT$^{\mathrm{opt}}$ denotes the proposed ZCC Viterbi algorithm with the optimal values of parameters, VITPRUNED$^{\mathrm{opt}}$ represents the algorithm from [12] with a pruning scheme that leads to a similar speed-up rate as ZCCVIT$^{\mathrm{opt}}$, ZCCVIT$^{\mathrm{fst}}$ is a very fast version of ZCCVIT with Q similar to VITPRUNED$^{\mathrm{opt}}$, and VITPRUNED$^{\mathrm{fst}}$ is a version of VITPRUNED with a similar speed-up rate as ZCCVIT$^{\mathrm{fst}}$.

Table 2. Comparison of different versions of the Viterbi algorithm for the female voice

Algorithm	S	CD	Q
VITBASE	1.00	0.2936	0.00
VITPRUNED$^{\mathrm{cns}}$	7.12	0.2936	0.00
ZCCVIT$^{\mathrm{opt}}$	438.30	0.2856	18.08
VITPRUNED$^{\mathrm{opt}}$	449.05	0.3702	71.14
ZCCVIT$^{\mathrm{fst}}$	4165.86	0.2667	72.81
VITPRUNED$^{\mathrm{fst}}$	4135.52	0.4555	459.21

Table 3. Listening test evaluation of different versions of the Viterbi algorithm for the male voice. A = B stands for "utterance A sounds same as B", A > B means "A sounds better than B", and A < B means "A sounds worse than B". Preferences are in percents.

Comparison	A = B	A > B	A < B
(A) ZCCVIT$^{\mathrm{opt}}$ vs. **(B)** VITBASE	87.50	5.42	7.08
(A) ZCCVIT$^{\mathrm{opt}}$ vs. **(B)** VITPRUNED$^{\mathrm{opt}}$	61.90	22.86	15.24
(A) ZCCVIT$^{\mathrm{fst}}$ vs. **(B)** VITPRUNED$^{\mathrm{fst}}$	52.07	31.03	16.90

A listening test based evaluation of the male voice is shown in Table 3. Evaluation of the female voice was limited to the comparison of ZCCVIT$^{\mathrm{opt}}$ vs. VITPRUNED$^{\mathrm{opt}}$—24.76% of listeners preferred ZCCVIT$^{\mathrm{opt}}$, 12.62% preferred VITPRUNED$^{\mathrm{opt}}$, and according to 62.62% both versions sounded the same.

As can be seen in Table 1–3, the dramatic reduction of the computation load of ZCCVIT$^{\mathrm{opt}}$ did not imply a noticeable decrease in synthetic speech quality when compared to VITBASE. Furthermore, ZCCVIT appears to be more stable in quality than VITPRUNED when increasing the speed-up factor S (both in terms of Q and according to the listening-test based evaluation). This may be caused by the fact that, to speed up VITPRUNED, a significant amount of candidates have to be pruned off, and only a few candidates per each speech unit remain available for selection. On the other hand, ZCCVIT still searches through the full network of unit candidates, or ZCC chains, respectively.

5. CONCLUSION

We presented a modified Viterbi search in which zero-concatenation-cost (ZCC) chains are utilized to speed up the process of unit selection speech synthesis. To synthesize speech with a proper target specification, the target cost is used to compromise between the length of ZCC chains (longer ZCC chains tend to violate the target specification) and the target specification.

The proposed algorithm reduces the computational load of a unit selection system up to hundreds of times (speed-up rate was 556.55 for a male voice and 438.30 for a female voice). Although the main objective was to speed up the unit-selection process, listening tests did not reveal any noticeable drop in synthetic speech quality when compared to synthetic speech produced by the baseline Viterbi search algorithm VITBASE. Unlike the Viterbi search with the pruning scheme (VITPRUNED proposed in [12]), the algorithm proposed in this paper is also more stable when very high speed-up rates (thousands of times faster than VITBASE) are required.

Listening tests revealed that synthetic speech of all versions (ZCCVIT, VITPRUNED, even VITBASE) in a comparable manner contained glitches, mainly caused by discontinuities in F0 and duration patterns. Our future work will be directed to an elimination of the glitches by avoiding to concatenate ZCC chains or single candidates with different F0 and temporal tendencies.

6. REFERENCES

[1] A. J. Hunt and A. W. Black, "Unit selection in concatenative speech synhesis system using a large speech database," in *Proc. ICASSP*, Atlanta, USA, 1996, pp. 373–376.

[2] M. Beutnagel, M. Mohri, and M. Riley, "Rapid unit selection from a large speech corpus for concatenative speech synthesis," in *Proc. EUROSPEECH*, Budapest, Hungary, 1999, pp. 607–610.

[3] J. Čepko, R. Talafová, and J. Vrabec, "Indexing join costs for faster unit selection synthesis," in *Proc. Internat. Conf. Systems, Signals Image Processing (IWSSIP)*, Bratislava, Slovak Republic, 2008, pp. 503–506.

[4] W. Hamza and R. Donovan, "Data-driven segment preselection in the IBM trainable speech synthesis system," in *Proc. INTERSPEECH*, Denver, USA, 2002, pp. 2609–2612.

[5] N. Nishizawa and H. Kawai, "Unit database pruning based on the cost degradation criterion for concatenative speech synthesis," in *Proc. ICASSP*, Las Vegas, USA, 2008, pp. 3969–3972.

[6] P. Tsiakoulis, A. Chalamandaris, S. Karabetsos, and S. Raptis, "A statistical method for database reduction for embedded unit selection speech synthesis," in *Proc. ICASSP*, Las Vegas, USA, 2008, pp. 4601–4604.

[7] Z. Hanzlíček, J. Matoušek, and D. Tihelka, "Experiments on reducing footprint of unit selection TTS system," in *Text, Speech and Dialogue*, vol. 8082 of *Lecture Notes in Computer Science*, pp. 249–256. Berlin, Heidelberg, 2013.

[8] Z. H. Ling, Y. Hu, Z. W. Shuang, and R. H. Wang, "Decision tree based unit pre-selection in mandarin chinese synthesis," in *Proc. ISCSLP*, Taipei, Taiwan, 2002.

[9] A. Conkie, M. Beutnagel, A. K. Syrdal, and P. Brown, "Preselection of candidate units in a unit selection-based text-to-speech synthesis system," in *Proc. ICSLP*, Beijing, China, 2000, vol. 3, pp. 314–317.

[10] A. Conkie and A. K. Syrdal, "Using F0 to constrain the unit selection Viterbi network," in *Proc. ICASSP*, Prague, Czech Republic, 2011, pp. 5376–5379.

[11] S. Sakai, T. Kawahara, and S. Nakamura, "Admissible stopping in Viterbi beam search for unit selection in concatenative speech synthesis," in *Proc. ICASSP*, Las Vegas, USA, 2008, p. 46134616.

[12] D. Tihelka, J. Kala, and J. Matoušek, "Enhancements of Viterbi search for fast unit selection synthesis," in *Proc. INTERSPEECH*, Makuhari, Japan, 2010, pp. 174–177.

[13] M. Lee, D. Lopresti, and J. Olive, "A text-to-speech platform for variable length optimal unit searching using perception based cost functions," *International Journal of Speech Technology*, vol. 6, no. 4, pp. 347–356, 2003.

[14] A. P. Breen and P. Jackson, "Non-uniform unit selection and the similarity metric within BT's laureate tts system," in *Proc. ESCA/COCOSDA Workshop on Speech Synthesis*, 1998, pp. 201–206.

[15] 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. ICASSP*, 2001, vol. 2, pp. 785–788 vol.2.

[16] J. Xu, D. Huang, Y. Wang, Y. Dong, L. Cai, and H. Wang, "Hierarchical non-uniform unit selection based on prosodic structure," in *Proc. INTERSPEECH*, Antwerp, Belgium, 2007, pp. 2861–2864, ISCA.

[17] A. W. Black, "Perfect synthesis for all of the people all of the time," in *Proc. IEEE Workshop on Speech Synthesis*, Santa Monica, USA, 2002.

[18] J. Matoušek, D. Tihelka, and J. Romportl, "Current state of Czech text-to-speech system ARTIC," in *Text, Speech and Dialogue*, vol. 4188 of *Lecture Notes in Computer Science*, pp. 439–446. Springer, Berlin, Heidelberg, 2006.

[19] J. Matoušek and J. Romportl, "On building phonetically and prosodically rich speech corpus for text-to-speech synthesis," in *Proc. 2nd IASTED Internat. Conf. on Computational Intelligence*, San Francisco, USA, 2006, pp. 442–447.

[20] J. Matoušek and J. Romportl, "Recording and annotation of speech corpus for Czech unit selection speech synthesis," in *Text, Speech and Dialogue*, vol. 4629 of *Lecture Notes in Computer Science*, pp. 326–333. Springer, Berlin, Heidelberg, 2007.