FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Optimalizace rychlosti výběru řečových jednotek v konkatenační syntéze řeči

Ing. Jiří Kala

disertační práce
k získání akademického titulu doktor
v oboru *Kybernetika*

Školitel: Doc. Ing. Jindřich Matoušek, Ph.D.
Katedra kybernetiky

Plzeň, 2014

FACULTY
OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

Speed Optimization of Unit Selection Algorithm in Concatenative Speech Synthesis

Ing. Jiří Kala

dissertation
submitted in conformity with the requirements
for the degree of Doctor of Philosophy
in the field of *Cybernetics*

Supervisor: Doc. Ing. Jindřich Matoušek, Ph.D.
Department of Cybernetics

Pilsen, 2014

Prohlášení

Předkládám tímto k posouzení a obhajobě disertační práci zpracovanou na závěr doktorského studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto práci vypracoval samostatně s použitím odborné literatury a dostupných pramenů, jejichž seznam je součástí práce.

V Plzni dne Jiří Kala

Poděkování

Chtěl bych v první řadě poděkovat svému školiteli Doc. Ing. Jindřichu Matouškovi, Ph.D. za cenné rady, konzultace a trpělivost při vedení předložené práce. Také bych rád poděkoval Ing. Danielu Tihelkovi, Ph.D. za jeho připomínky a názory na řešení řady problémů. Dále děkuji Ing. Martinu Grůberovi, Ph.D. za pomoc s nástrojem na vytváření poslechových testů a všem ostatním kolegům na Katedře kybernetiky, kteří se testů zúčastnili.

Speciálně bych rád poděkoval Prof. Ing. Josefu Psutkovi, CSc., jehož přístup k výuce a studentům mě v začátcích inženýrského studia přiměl k přehodnocení původního studijního zaměření a jeho změně na obor Kybernetika a řídící technika.

Velké poděkování patří i mé manželce Vandě za její trpělivost a významnou podporu v průběhu celého studia.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek a symbolů

ARTIC	Artificial Talker in Czech
BEFS	Best–First Search (hledání cesty grafem rozvojem nejlepší cesty)
BW	Beam Width
C^{c}	Concatenation Cost (cena řetězení)
CD	Concatenation Density (hustota spojení)
C^{t}	Target Cost (cena cíle)
F_0	frekvence základního hlasivkového tónu
TTS	Text–to–Speech
TW	Target Cost Width
TWR	Relative Target Cost Width
VITORIG	Original Viterbi Algorithm (základní Viterbiův algoritmus)
VITBASE	Baseline Viterbi Algorihm (vylepšený Viterbiův algoritmus používaný v systému ARTIC)
VITOPT	Optimized Viterbi Algorithm (optimalizovaný Viterbiův algoritmus)
VITBEAM	Beam search Viterbi Algorithm (optimalizovaný Viterbiův algoritmus doplněný o prořezávání typu BEAM)
VITPRUNE	Pruned Viterbi Algorithm (optimalizovaný Viterbiův algoritmus s prořezáváním typu BEAM a prořezáváním dle ceny cíle)
ZCC chain	Zero–Concatenation–Cost Chain (řetězec realizací jednotek s nulovou cenou řetězení)
ZCCBEFS	ZCC Best–First–Search (hledání cesty grafem rozvojem nejlepší cesty s využitím ZCC řetězců)
ZCCFRAME	ZCC Frame–Search (dvouúrovňové prohledávání grafu v němž se nejprve složí kostry cest pomocí ZCC řetězců)
ZCCVIT	ZCC Viterbi–Search (hledání cesty grafem s využitím ZCC řetězců založené na principu Viterbiova algoritmu)

Nově zaváděné pojmy jsou v textu vždy vyznačeny *kurzívou*.

Kapitola 1

Úvod

Syntéza řeči z textu se v dnešní době stala již poměrně běžnou součástí systémů, které řeší komunikaci mezi strojem a člověkem. Jejím úkolem je konverze psaného textu do podoby lidské řeči a nalézá své využití v různých oblastech, zejména pak v informačních systémech a počítačích obecně. Systémy, které jsou schopné komunikovat s člověkem pomocí řeči, tj. pro něj přirozenou cestou, mohou pomáhat postiženým lidem s ovládáním počítačů nebo jim číst knihy. Dále je lze využívat v mobilních zařízeních a telefonech ke čtení informací a zpráv v situacích, kdy je ztížené jejich ovládání, například při řízení vozidla. Syntéza řeči je i součástí dialogových systémů, které mají za úkol zautomatizovat činnosti, které by jinak musel řešit lidský operátor, jako jsou poskytování informací z různých databází jízdních řádů nebo jakou je problematika rezervačních systémů (hotely, letenky). V kombinaci se systémy na rozpoznání řeči a digitálním slovníkem se začínají postupně objevovat i takové aplikace, které kupříkladu umožní v reálném čase překládat mluvené slovo do cizího jazyka a nahradit tím funkci tlumočníka.

Přestože paměťová kapacita a výpočetní výkon elektronických zařízení roste každým rokem, není ani dnes zcela jednoduché vytvořit systém převádějící text na řeč, který by dokázal generovat plynulou přirozenou řeč a současně by významným způsobem nezatěžoval přístroje velmi vysokými nároky. U přenosných zařízení, například mobilních telefonů, je kromě stále ještě problematických vysokých nároků na kapacitu úložného prostoru i nezbytná vyšší výpočetní síla. Rychlost těchto zařízení bývá sice již dostačující, ale větší zátěž znamená nutně i větší spotřebu energie, která může být pro reálné použití limitující. S rozvojem internetu se v poslední době začíná

prosazovat i u mobilních telefonů řešení, kdy je řeč zpracovávána na serveru a k uživateli je přenášen pouze výsledek. Serverový přístup však naráží na náročnost datového připojení, které opět nemusí být uživateli dostupné, a to ani finančně (např. poplatky za roaming v zahraničí), ani technologicky (dostupnost, dostatečná rychlost). Je také v dnešní době nepravděpodobné, že by si uživatel se zrakovým hendikepem nechal číst celou knihu syntetizérem umístěným jinde než na lokálním přístroji, ať již mobilním nebo i klasickým stolním počítačem.

Pokud je dialogový systém provozován na serverové infrastruktuře, nastávají opět problémy s dostatkem výkonu, byť odlišného charakteru než u mobilních zařízení. Výpočetní síla serveru při využití možností vícejádrových procesorů sice postačuje pro syntézu řeči pro jednotky uživatelů, ale ani při využití nejrychlejších procesorů nemusí být výkon dostačující, pokud má celý systém v jednom okamžiku obsloužit velké množství požadavků.

Předkládaná disertační práce se zabývá obecnou optimalizací TTS systémů na bázi konkatenační syntézy, přičemž prováděné experimenty jsou testovány na systému ARTIC[1)], který je vyvíjen na Katedře kybernetiky Západočeské univerzity v Plzni (Matoušek *et al.*, 2006).

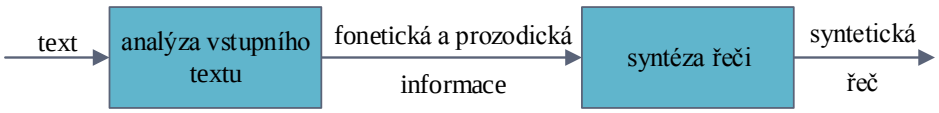
[1)] ARTIC je zkratkou z angl. *Artificial talker in Czech.*

Kapitola 2

Syntéza řeči

Syntézou řeči obecně nazýváme proces, který generuje umělou lidskou řeč. Umělou v tom smyslu, že není vytvářena samotným člověkem, ale technickým zařízením, které se nazývá *syntetizér řeči*. Takovému zařízení je na vstupu typicky předávána fonetická a prozodická informace. Výstupem syntetizéru je pak syntetická řeč, která by v ideálním případě měla být srozumitelná a přirozená tak, že ji posluchač nerozezná od lidské promluvy. Způsoby generování umělé řeči se postupem času vyvíjely od primitivních mechanických nástrojů, přes různé analogové elektronické zařízení, až po dnešní digitální syntetizéry řešené prostřednictvím číslicových počítačů. Přestože je kvalita syntetické řeči v dnešní době nesrovnatelně lepší než dříve, stále se pouze přibližuje popsanému ideálnímu stavu.

Komplexnější úlohou, než prostá syntéza řeči, je obecný převod psaného textu na odpovídající promluvu, který se nazývá *syntéza řeči z textu*. Systémy převádějící psaný text na syntetickou řeč se označují zkratkou TTS (z angl. *Text-To-Speech*). Jejich základními prvky jsou dva na sebe navazující subsystémy, kde první z nich zajišťuje analýzu původního vstupního textu a jeho výstupem je fonetická a prozodická informace. Takto získané informace jsou následně předány druhému subsystému – syntetizéru, který již zajišťuje samotné vygenerování syntetické řeči.



Obrázek 2.1: Schéma obecného TTS systému.

Těžiště této práce spočívá v optimalizaci procesů v rámci druhého subsystému a kompletní schéma na obrázku 2.1 je zde uvedeno pouze pro úplnost. Detailnější informace o obecných TTS systémech a analýze vstupního textu lze získat např. v Psutka *et al.*, 2006 nebo Benesty *et al.*, 2008.

2.1 Přístupy k řešení syntézy řeči

Historicky první pokusy v oblasti syntézy řeči byly realizovány pomocí mechanických nástrojů využívajících různé měchy, trubice a pryžové části, kterými bylo možné napodobit některé hlásky, případně i slova. Nejstarší zmínka o pokusech s vytvářením umělé řeči se datuje k roku 1779, ve kterém sestrojil profesor Christian Kratzenstein zařízení schopné generovat dlouhé samohlásky. Později v roce 1791 zkonstruoval Wolfgang von Kempelen *mluvící stroj*, který byl údajně schopen vytvářet slova i části vět. Mechanické syntetizéry se pokoušely v podstatě modelovat celý hlasový trakt, ale produkovaná řeč nebyla úplně srozumitelná. Navíc bylo ovládání těchto přístrojů velmi komplikované a do určité míry i fyzicky náročné. Mezi další průkopnické pokusy s vytvářením syntetické řeči patří i některé elektricky napájené přístroje. Jedním z příkladů může být klávesami ovládaný *VOCODER*, který byl představen společností *Bell Laboratories* v roce 1943[1)]. Přestože se stále ještě nedá hovořit o přirozeně znějící produkci, jednalo se pravděpodobně o první srozumitelný syntetizér. Detailnější informace o historii syntézy řeči můžeme najít např. v publikaci Psutka *et al.*, 2006, podkapitola 10.2.

Základní rozdělení syntetizérů řeči dle přístupu

Obecně se začínají syntetizéry moderního typu objevovat až s rozmachem číslicových počítačů, které svými možnostmi a výpočetním výkonem umožnily postupně vytvářet systémy schopné produkovat relativně srozumitelnou a přirozenou řeč.

Přístupy k řešení syntézy řeči lze rozdělit podle použitého principu, nebo také modelu, na tradiční tři typy – *formantovou*, *artikulační* a *konkatenační syntézu*. V posledních letech se prosazuje i čtvrtý nový přístup nazývaný

[1)] První verzi syntetizéru VOCODER sestrojil Homer Dudley již v roce 1928. Systém byl později v roce 1939 patentován a veřejně představen na světové výstavě v New Yorku.

statistická parametrická syntéza (Matoušek, 2008, oddíl 2.3.3 nebo Dutoit, 2008).

Formantová syntéza vychází z teorie zdroje a filtru, která zjednodušeně modeluje tvorbu hlasu u člověka pomocí dvou nezávislých složek. První složkou je zdroj buzení, jenž představuje hlasivky a generuje periodický signál pro znělé hlásky a bílý šum pro hlásky neznělé. Druhou složkou je lineární akustický filtr, který reprezentuje hlasový trakt, resp. modeluje jeho formantové rezonance (detaily např. v Allen *et al.*, 1987).

Výhodou formantové syntézy je malý počet potřebných parametrů, schopnost vytvářet vysoce srozumitelnou řeč, a protože se jedná o syntézu na bázi pravidel, je velmi úsporná z hlediska velikosti, protože nepracuje s databází řečových segmentů[2)]. Umožňuje také snadné řízení prozodických charakteristik a změnu rychlosti řeči bez ztráty kvality nebo výskytu nežádoucích zvuků. I přes nízký počet parametrů je velkou nevýhodou velmi náročný proces jejich nastavení. Další podstatnou nevýhodou je velmi nízká přirozenost generované promluvy, která zní uměle, někdy až „roboticky“.

Dříve byla formantová syntéza velmi rozšířená, ale právě vzhledem k nízké úrovni přirozenosti vytvářené řeči není dnes tento přístup již tak populární.

Artikulační syntéza využívá fyzikální model hlasového traktu. Na rozdíl od formantové syntézy, jež používá pouze zjednodušený model, je artikulační syntéza postavena na komplexním modelu všech částí lidského těla, které se podílejí na produkci řeči (detaily např. v Rubin *et al.*, 1981).

Přístup artikulační syntézy je sice nejobecnější a měl by být schopen generovat syntetickou řeč nejvyšší kvality, ale vzhledem ke své vysoké složitosti a náročnosti není v podstatě jako celek dořešen a není využíván v žádném produktivním nebo komerčně využívaném systému[3)].

[2)] Viz oddíl 3.2

[3)] První pokusy vytvořit obecný syntetizér založený na artikulační syntéze se datují do poloviny sedmdesátých let dvacátého století, kdy se jeho vývojem zabývala společnost *Haskins Laboratories*. Jedním z významnějších kroků v této oblasti bylo později vytvoření jiného systému společností *Trillium Sound Research* v roce 1995 (Hill *et al.*, 1995) pro počítače společnosti *NeXT*. Později byl zdrojový kód tohoto systému uvolněn pod názvem *gnuspeech* a pod licencí GNU GPL.

Konkatenační syntéza využívá dopředu připravenou databázi segmentů reálné řeči, které jsou podle požadavků na vstupu syntetizéru zřetězeny (spojeny – angl. *concatenate*) do výsledné promluvy. Výhodou této metody je relativně snadný návrh i příprava nezbytné databáze řečových segmentů. Díky tomu, že se zde využívá úseků reálné řeči, je generovaná syntetická řeč velmi přirozená a také srozumitelná. Z těchto důvodů se metoda těší velkému zájmu a patří mezi nejčastěji využívané v reálných nebo komerčně využívaných systémech. Protože je problematika konkatenační syntézy předmětem této práce, je jejímu detailnímu popisu věnována celá následující kapitola č. 3.

Statistická parametrická syntéza je postavená na modelování jednotlivých řečových jednotek pomocí statistických modelů. Až na výjimky se používají výhradně skryté Markovovy modely a generování řeči neprobíhá přímo řetězením reálných řečových segmentů jako u konkatenační syntézy, ale výsledný průběh řečového signálu je generován právě z připravených Markovových modelů (angl. *Hidden Markov Model* – proto se tato technika nazývá také *HMM syntéza*). Podrobnosti k tomuto způsobu syntézy řeči viz např. Tokuda *et al.*, 1995, Yoshimura *et al.*, 1999, Dutoit, 2008 nebo Zen *et al.*, 2009.

2.2 Aktuální trendy a způsoby využití syntetizérů

V dnešní době patří mezi hlavní proudy zájmu konkatenační a HMM syntéza. Oba přístupy se řadí mezi tzv. *korpusově orientované*, neboť využívají velmi rozsáhlé řečové korpusy, byť každý z přístupů jej využívá jiným způsobem. Řečový korpus může obsahovat až desítky hodin nahraného řečového signálu doplněného o spektrální a prozodické vlastnosti řečových jednotek.

V přenosných zařízeních se kvůli úspoře paměťové kapacity a nižšího výpočetního výkonu velmi často implementuje klasická konkatenační syntéza, kdy je v databázi řečových segmentů pouze jeden zástupce pro každou jednotku. Pokud je syntéza provozována na serverech nebo i na pracovních stanicích s dostatečnou rezervou systémových prostředků, pak se volí spíše konkatenační syntéza s výběrem jednotek (angl. *unit selection*), která produkuje kvalitnější a přirozenější syntetickou řeč.

V posledních letech se přesouvá většina výzkumného zájmu směrem k poměrně slibné HMM syntéze a začínají se objevovat i reálné a zdařilé implementace tohoto přístupu. Nespornou výhodou tohoto přístupu je výrazná úspora požadované paměťové kapacity, jelikož není nutné uchovávat velkou databázi reálných řečových segmentů, ale pouze parametry jejich modelů. Dále pak lze při užití tohoto přístupu velmi flexibilně a efektivně řešit změny hlasu nebo vytvoření hlasu nového.

Konkatenační syntéza na jednu stranu ustupuje do pozadí zájmu výzkumu zejména proto, že se jedná již o léty prověřený přístup, na kterém již nezbývá mnoho zajímavých výzkumných problémů. Na straně druhé, se však jedná o metodu, kterou lze poměrně efektivně nasadit ve velkém množství reálných aplikací a která je zároveň schopná generovat velmi kvalitní a přirozeně znějící syntetickou řeč. V době nástupu mobilních zařízení je tak stále aktuální i oblast zabývající se optimalizací velikosti nezbytné databáze řečových vzorků nebo urychlení algoritmů výběru řečových jednotek.

Kapitola 3

Konkatenační syntéza

Jak již bylo naznačeno v předchozí kapitole, je základním principem konkatenační syntézy vytváření souvislé lidské řeči spojením kratších úseků reálného řečového signálu. Ty se nazývají *řečovými segmenty* a jsou uložené v *databázi řečových segmentů*, kde každý segment reprezentuje jednu z řečových jednotek, a proto se pro ně také užívá pojmu *realizace* nebo i *zástupce* řečové jednotky. Systémy konkatenační syntézy pak z připravené databáze řečových segmentů vybírají podle potřeby vhodné realizace a řetězí je do výsledné promluvy.

3.1 Volba typu řečových jednotek

V úvodu vývoje syntetizéru je potřeba zvážit cílový způsob jeho využití a na jeho základě zvolit vhodné parametry, které budou pro daný účel optimální. Jedním z klíčových parametrů je typ použitých *řečových jednotek*. Pojmem řečová jednotka se označuje abstraktní typový úsek řeči, kterým je například slovo, slabika nebo hláska, ale také se může jednat o delší úseky řeči, jako jsou celé fráze nebo věty. Při rozhodování jaký typ jednotek bude použit, je důležité zvážit i další kritéria jako je snaha o maximální pokrytí koartikulačních jevů[1)] a minimalizace problémů při řetězení jednotek (bližší popis viz např. Psutka *et al.*, 2006, oddíl 10.4.3 nebo Benesty *et al.*, 2008, kapitola 24).

[1)] Lidská řeč je složena ze zvuků, kde každý zvuk je dán konkrétní konfigurací hlasového traktu (artikulačních orgánů). Změna konfigurace hlasového traktu není skoková, a tak je znění jednotlivých zvuků ovlivněno kontextem předcházejícího a následujícího zvuku. Dochází tedy k tzv. koartikulačním jevům – blíže viz např. Palková, 1994 nebo Šiška, 2001.

Pokud má navrhovaný systém konkatenační syntézy sloužit k produkci řeči, která využívá pouze omezený slovník (angl. *domain limited synthesis*), pak se využívá delších řečových jednotek, například vět, slov nebo frází. Příkladem takových systémů jsou automatické hlásiče na nádražích nebo v prostředcích hromadné dopravy, případně subsystémy v dialogových systémech používaných na zákaznických linkách (call centra). Zde se používají delší úseky promluvy, které jsou konstantní a které se doplňují o proměnlivé části, kupříkladu číslovky. Generovaná řeč je pak velmi kvalitní a přirozená, ale syntetizér není schopen tvořit řeč mimo rámec svého omezeného „slovníku".

Pokud je cílem vývoje připravit obecný systém, jenž je schopen generovat libovolnou promluvu, tj. lze jej využít pro převod libovolného textu na řeč (TTS), pak se typicky volí kratší řečové jednotky. Jejich použitím lze lépe pokrýt potřeby „neomezeného slovníku", protože nahrát všechny věty nebo i slova daného jazyka ve všech kontextech je prakticky nemožné.

Pro přehled bude nyní uveden výčet typů řečových jednotek, které se nejvíce používají u obecných syntetizérů[2)].

Difony (angl. *diphones*) jsou řečové jednotky, jejichž začátek je v polovině jednoho fonému a končí v polovině následujícího. Výhodou je zachování přechodu mezi hláskami a segmentace prochází spektrálně stabilními částmi uprostřed fonémů.

Trifony (angl. *triphones*) bývají definovány dvěma různými způsoby. První z nich definuje trifon jako jednotku, která zahrnuje jednu celou hlásku společně s polovinou předcházející i následující hlásky. V druhém případě je trifon specifikován jako *kontextově závislá hláska*, kdy se jedná pouze o samotnou hlásku, ale posazenou do kontextu levého a pravého okolí.

Polofony (angl. *half-phones*) začínají na začátku hlásky a končí v její polovině, nebo začínají uprostřed hlásky a končí na hranici s další hláskou.

[2)]Kromě uvedených typů jednotek lze zmínit i další méně využívané typy jako jsou slabiky, demislabiky nebo fonémy (hlásky). Úplný přehled včetně detailnějšího popisu viz Psutka *et al.*, 2006, str. 555 až 557.

3.2 Příprava databáze řečových segmentů

Jedním z hlavních úkolů přípravné fáze vývoje systému konkatenační syntézy je vytvoření databáze řečových segmentů, ze které se pak vybírají realizace při samotné syntéze řeči. Tento proces se provádí v několika krocích, které jsou schematicky znázorněny na obrázku 3.1 a stručně popsány v následujícím textu, protože detailní vysvětlení jde nad rámec tématu této práce (bližší detaily viz např. Psutka *et al.*, 2006, oddíl 10.4.4 nebo Matoušek, 2008, podkapitola 4.1).

Výběr vět do textového korpusu

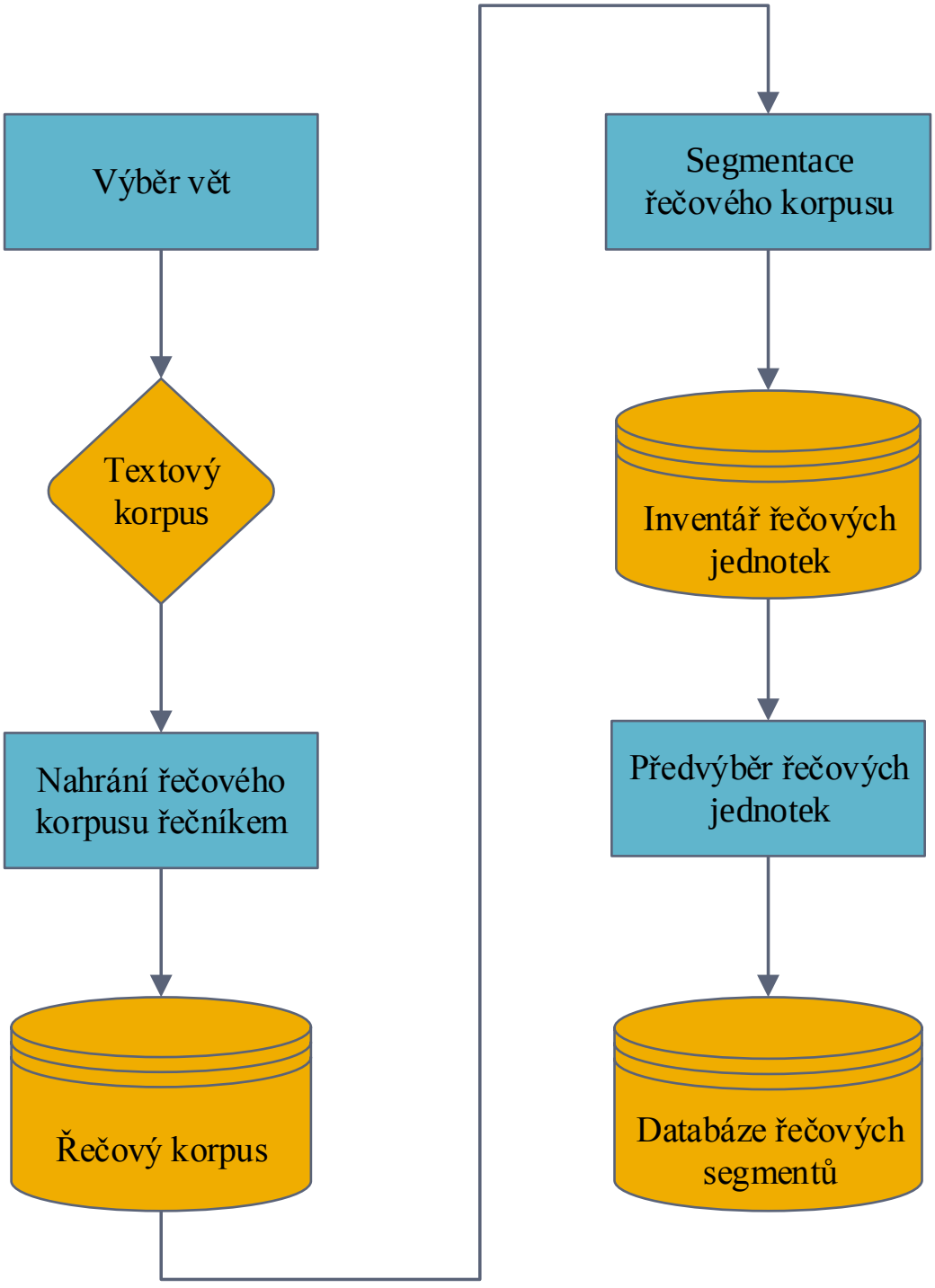
Prvním krokem při přípravě databáze řečových segmentů je výběr vět do *textového korpusu*, který je tvořen vhodně vybraným textem (typicky soubor vět) pokrývajícím potřeby syntetizéru tak, aby v něm byly zastoupeny všechny řečové jednotky minimálně jednou ve všech vyžadovaných kontextech.

Namluvení řečového korpusu

Dále se provádí namluvení vět nebo frází z textového korpusu řečníkem a záznam je uložen do *řečového korpusu* společně s odpovídajícím textem a typicky i s dalšími pomocnými informacemi jako jsou například hlasivkové pulsy[3)], které se používají zejména k nalezení vhodných bodů napojení signálu dvou řečových segmentů při samotném řetězení a minimalizují riziko vzniku fázových nespojitostí nebo nespojitostí průběhu F_0.

Kvalita nahrávky řečového korpusu má naprosto zásadní dopad na kvalitu řeči produkované syntetizérem. Navíc výsledný hlas je v podstatě stejný s hlasem řečníka. Proto je potřeba zvolit i kvalitní technické zázemí pro nahrávání (ideálně nahrávací studio) a dobře zvolit řečníka, který musí být schopen namluvit text konzistentním stylem a s pečlivou artikulací.

[3)] Hlasivkové pulsy odpovídají okamžikům uzavření hlasivek. Získávají se ze signálu měřeného přímo na hlasivkách pomocí přístroje nazývaného *laryngograf* (EGG). Bližší popis viz např. Matoušek, 2008, oddíl 4.1.5.



Obrázek 3.1: Schéma procesu přípravy databáze řečových segmentů.

Segmentace řečového korpusu a vytvoření inventáře řečových jednotek

V této části přípravy systému se řeší problém nalezení hranic jednotlivých realizací řečových jednotek v nahraném hlasovém signálu uchovaném v řečovém korpusu. Jednou z metod je manuální segmentace, která je prováděna zkušenými experty z oblasti fonetiky. Problémem tohoto přístupu je velká časová náročnost a pracnost, proto se používá spíše v případech vytváření syntetizéru s omezeným slovníkem.

U systémů založených na velmi rozsáhlých řečových korpusech se hranice realizací řečových jednotek hledají automaticky. Nejrozšířenějšími jsou techniky automatické segmentace využívající skryté Markovovy modely – HMM (Matoušek *et al.*, 2008 nebo Matoušek *et al.*, 2003) a porovnání se syntetickou řečí metodou dynamického borcení časové osy – DTW[4)] (Horák, 2002).

Nalezením hranic realizací řečových jednotek v řečovém korpusu vzniká *inventář* zahrnující všechny zástupce řečových jednotek, které lze následně využívat při syntéze řeči.

Předvýběr realizací do databáze řečových segmentů

Zejména v případě použití obsáhlého řečového korpusu v kombinaci s jeho automatickou segmentací, obsahuje inventář jednotek množství realizací pro každou řečovou jednotku, a to v různých fonetických, spektrálních a prozodických kontextech. Řečové segmenty jedné jednotky si mohou být navzájem velmi podobné, a proto se v závislosti na záměru s využitím syntetizéru může provést jejich *předvýběr*[5)], kterým se vyřadí nadbytečné realizace a do databáze řečových segmentů[6)] se pak uloží pouze vhodní zástupci příslušné skupiny obdobných zástupců dané řečové jednotky.

Předvýběrem jednotek lze také řešit eliminaci realizací, které jsou vyhodnoceny jako chybně segmentované nebo ne zcela ideálně nahrané (řečník udělá

[4)] DTW je zkratka z angl. *Dynamic Time Warping* a popis této techniky je uveden např. v Psutka *et al.*, 2006, str. 509 až 511.

[5)] V případě, že se počítá s použitím všech jednotek v řečovém korpusu, je inventář řečových jednotek obsahově shodný s databází řečových segmentů.

[6)] Technicky se velmi často ani nevytváří speciální databáze řečových segmentů, ale používá se přímo původní řečový korpus. Pokud se provádí předvýběr jednotek, pak lze nahradit databázi řečových segmentů například pomocným indexem jednotek, jež do ní byly zařazeny.

chybu). To lze provést za pomoci automaticky zpracovaných statistik, jež určí segmenty lišící se výrazně délkou, hlasitostí nebo jinými charakteristikami od průměru pro danou řečovou jednotku.

Vhodným omezením počtu realizací řečových jednotek lze dosáhnout i snížení velikosti objemu dat potřebných pro funkci syntetizéru. Jedním z přístupů je předvýběr na základě frekvence použití daných realizací jednotek při syntéze většího množství textu (viz např. Hamza & Donovan, 2002, Isogai & Mizuno, 2010 nebo Tihelka, 2007). Jiným příkladem je práce Tsiakoulis *et al.*, 2008, v níž se při syntéze textu pro získání statistik frekvence výskytu uchovává navíc i pomocné skóre, které je vypočítané na základě ceny cíle a ceny řetězení. Skóre udává míru podobnosti dvou realizací řečových jednotek a používá se při předvýběru k eliminaci sice často používaných, ale velmi blízkých (a tedy redundantních) realizací.

Obecné syntetizéry založené na principu konkatenační syntézy lze také rozlišit podle toho, zda pro jednu řečovou jednotku uchovávají v databázi jednu nebo více realizací. V případě jedné realizace vybírá proces předvýběru nejvhodnějšího zástupce pro každou jednotku.

Klasická konkatenační syntéza (angl. *fixed inventory* nebo také *single unit synthesis*) má v databázi řečových segmentů uchovaného pouze jednoho univerzálního zástupce ke každé řečové jednotce. To znamená, že databáze řečových segmentů má malý objem a že algoritmus výběru vhodných kandidátů je pro svou jednoduchost velmi rychlý. Nevýhodou je pak nižší pokrytí koartikulačních jevů nebo zvýšené riziko výskytu problémů při řetězení jednotek, protože v daném kontextu nemusí segmenty zcela ideálně navazovat. Tento přístup k syntéze tedy vyžaduje výrazné modifikace řečového signálu a výsledkem je pak horší kvalita řeči, než při uchování více zástupců řečových jednotek.

Syntéza s výběrem jednotek (angl. *unit selection* nebo také *multiple unit selection*) na rozdíl od klasické konkatenační syntézy ukládá do databáze řečových segmentů pro řečové jednotky více zástupců. Tím lze dosáhnout výrazně kvalitnější a přirozeněji znějící produkci syntetické řeči, nicméně tato výhoda je vykoupena nutností použít vhodný algoritmus, který ze všech možných zástupců vybere nejvhodnější segmenty. Tyto algoritmy jsou totiž velmi náročné na výpočetní výkon a při masivnějším nasazení

mohou působit potíže s provozem i na vysoce výkonných moderních serverech.

3.3 Proces syntézy řeči

Pokud máme připravenou databázi řečových segmentů, pak je možné přistoupit k realizaci samotného mechanismu syntézy řeči. Schematicky je proces probíhající v rámci subsystému označeného jako *syntéza řeči* znázorněn na obrázku 3.2. V následujícím textu budou uvedeny hlavní principy jednotlivých kroků procesu (podrobnosti viz např. Psutka *et al.*, 2006, oddíly 10.4.5 a 10.4.6).

Generování posloupnosti jednotek

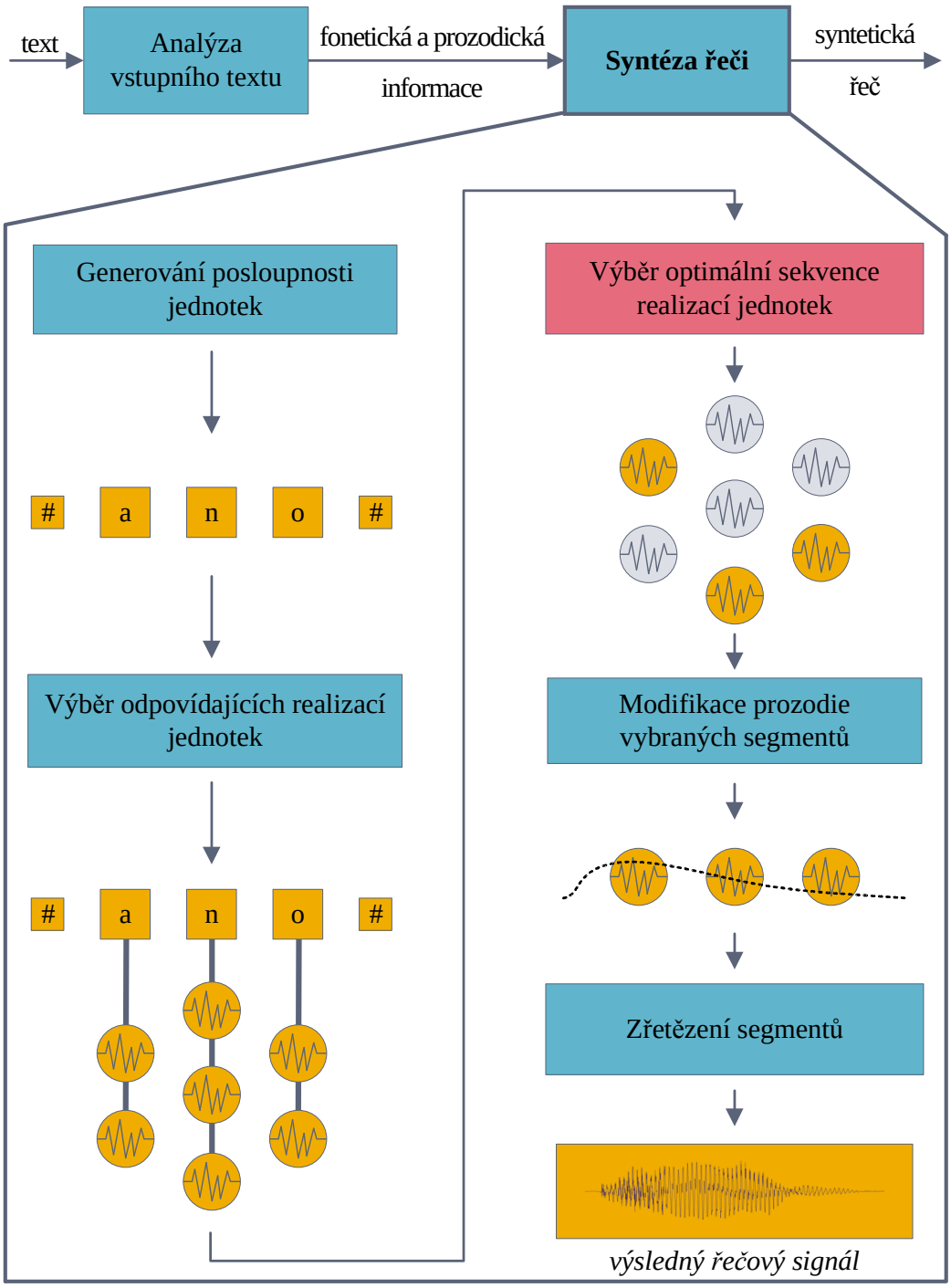
Subsystém provádějící samotnou syntézu pracuje na vstupu s fonetickým a prozodickým popisem požadované promluvy, která se má vygenerovat. Fonetická informace zahrnuje řetěz hlásek (fonémů), jež mají být vysloveny. Prozodická část vstupních informací určuje prozodické charakteristiky, tj. melodii, časování, intenzitu hlasu, případně další doplňující informace používané například pro vyjádření emocí, expresivních znaků řeči, nálady atd. (více viz Benesty *et al.*, 2008, kapitola 23). Nejčastěji se prozodie v procesu syntézy řeči modeluje *explicitně* generováním kontury F_0.

Prozodii lze modelovat i *implicitně*[7] pomocí příznaků, jež vhodným způsobem vyjadřují prozodický kontext jednotek[8] a jejichž zohledněním v procesu výběru jednotek dochází k výběru realizací, které se svými vlastnostmi blíží požadovaným prozodickým specifikacím. Prozodie syntetizované řeči je tak určena nepřímo a odpadá zde potřeba dalších (explicitních) modifikací (více viz Psutka *et al.*, 2006, podkapitoly 2.5 a 2.8 nebo Taylor, 2009, kapitola 6).

Na základě těchto informací a s ohledem na typ a délku řečových jednotek uložených v databázi řečových segmentů se vytvoří sekvence odpovídajících

[7] Místo prozodických parametrů popisujících konturu F_0 se pracuje s příznaky a symboly popisujících nepřímo prozodii řečových jednotek – proto se implicitní model nazývá také jako *symbolický popis prozodie*.

[8] Prozodický kontext řečových jednotek lze popsat například pomocí příznaků jako je pozice fonému v rámci prozodických struktur (kluze, fráze, prozodém, ...) nebo lze použít příznaky zohledňující paralingvistické jevy (nálada, styl, ...). Detaily viz Romportl, 2006 nebo Grůber & Tihelka, 2010.



Obrázek 3.2: Schéma procesu tvorby řeči u syntetizéru založeného na principu konkatenace.

řečových jednotek, která s sebou nese i detailní popis požadavků na prozodii jednotlivých jednotek a je také nazývána jako *specifikace cíle*.

Výběr odpovídajících realizací jednotek

Ke každé jednotce vybrané sekvence se následně vyberou z databáze řečových segmentů její příslušné realizace. Pokud systém využívá pouze jednoho zástupce pro každou řečovou jednotku, pak se následující krok výběru optimální sekvence z procesu syntézy vynechává.

Výběr optimální sekvence realizací jednotek

Pokud je však v databázi řečových segmentů uloženo k jednotkám více realizací, pak se stávají *kandidáty*, z nichž se pomocí různých algoritmů vybírají nejvhodnější zástupci. Daným způsobem se stanoví optimální řetězec řečových segmentů, který nejlépe vyhovuje fonetickým a prozodickým požadavkům. Kromě toho se také hodnotí, jak dobře bude možné napojit zástupce jednotek na předchozí a následující segmenty.

Optimalizace algoritmu výběru nejlepšího řetězce realizací řečových jednotek je součástí cílů této práce a tomuto tématu je podrobněji věnována kapitola č. 4.

Modifikace prozodie vybraných segmentů

Prozodické charakteristiky vybraných segmentů se většinou liší od požadovaných, a proto se v tomto kroku u každého z nich prozodie upravuje (pokud je to nutné), aby vyhověla požadovanému prozodickému kontextu. Tyto modifikace pomáhají zlepšit přirozenost výsledné syntetické řeči, ale v případě výrazných změn může dojít i ke vzniku *artefaktů*[9)] způsobujících snížení její kvality.

Nejčastěji používanou technikou pro modifikaci prozodie je TD-PSOLA (z angl. *Time-Domain Pitch Synchronous Overlap and Add*). Jako příklady dalších metod lze uvést například HNM (z angl. *Harmonic plus Noise Model*)

[9)] Pojem artefakt (v angl. jazyce se užívá výraz *glitch*) se používá k označení části syntetizované promluvy, v níž došlo v souvislosti napojením dvou ne zcela vyhovujících segmentů ke vzniku nepřirozeného prvku, např. výrazného skoku ve frekvenci hlasu, jenž nelze ani lidskými hlasivkami vytvořit.

nebo *LPC* (z angl. *Linear Predictive Coding*). Přehled těchto metod viz např. Benesty *et al.*, 2008, kapitola 24, Taylor, 2009, kapitola 14 nebo Psutka *et al.*, 2006, oddíly 10.4.6.3 a 10.4.6.4.

Zřetězení segmentů

Závěrečným krokem při syntéze řeči je zřetězení segmentů řeči do podoby výsledného řečového signálu. Existuje několik technik k řešení tohoto problému.

Nejjednodušším přístupem je přímé řetězení řečového signálu v časové oblasti, tedy zařazení řečových segmentů za sebe a vyhlazení přechodů mezi nimi. Tato technika neumožňuje provádět změny prozodie a vyhlazovat přechody mezi segmenty, a proto se hodí spíše pro syntézu s omezeným slovníkem, případně ji lze využít u velmi rozsáhlých řečových korpusů pokrývajících širokou škálu různých kontextů použití jednotek.

Vyšší kvalitu syntetické řeči lze – teoreticky – dosáhnout použitím dalších přístupů, které jsou mnohem složitější a pracují nejen v časové, ale i ve frekvenční oblasti, a umožňují mnohem lépe měnit spektrální vlastnosti realizací řečových jednotek. Jako příklad je možno uvést techniku FD-PSOLA (z angl. *Frequency-Domain Pitch Synchronous Overlap and Add*), která se však pro svou složitost v reálných aplikacích prakticky nepoužívá.

Souhrnný podrobnější popis k této i předchozí části procesu syntézy řeči lze nalézt v Psutka *et al.*, 2006, oddíl 10.4.6.

Kapitola 4

Syntéza výběrem jednotek

Jednou z nejpoužívanějších metod[1)] korpusově orientované syntézy je syntéza výběrem jednotek (angl. *unit selection*). Její použití předpokládá databázi řečových segmentů s více než jedním zástupcem pro každou řečovou jednotku. Většinou je databáze vytvořena automatickou segmentací velmi rozsáhlého řečového korpusu a často se zde pracuje až se stovkami nebo tisíci realizacemi jednotek.

Principem metody výběru jednotek je nalezení optimálního řetězce zástupců jednotek ze všech dostupných realizací v databázi řečových segmentů. Kritériem hodnocení optimality je vybrat takovou posloupnost zástupců jednotek, ve které budou minimalizovány zásahy do původního řečového signálu pro dosažení požadavků na prozodické charakteristiky a hladké zřetězení segmentů. Volbou vhodných řečových segmentů lze dosáhnout vyšší přirozenosti a celkové kvality syntetické řeči.

4.1 Algoritmus výběru řečových jednotek

Algoritmus vyhledávání nejvhodnějších realizací pracuje se dvěma hodnoticími funkcemi (Hunt & Black, 1996), které vyjadřují vhodnost použití konkrétních realizací řečových jednotek v daném kontextu.

Cena cíle vyjadřuje svou hodnotou, jak se liší realizace řečové jednotky od požadované specifikace cíle (viz část 3.3, odstavec o generování posloup-

[1)] Existují i jiné metody pro výběr vhodných řečových segmentů, které jsou například založené na binárních rozhodovacích stromech – více viz např. Psutka *et al.*, 2006, oddíly 10.4.5.1 a 10.4.5.2

nosti jednotek). Funkci budeme značit C^{t} podle anglického názvu *target cost*.

Jak bylo již naznačeno v předchozích kapitolách, je specifikace cíle generována z výstupů modulu systému syntézy řeči provádějícího analýzu vstupního textu a určuje požadované vlastnosti řečových jednotek, kterými jsou prozodie a fonetický kontext (sousední jednotky, pozice ve větě či slově, apod.). Všechny realizace řečových jednotek z řečového korpusu jsou také opatřeny těmito informacemi.

S příznaky specifikace cíle řečových jednotek a s příznaky kandidátů se pracuje jako s vektory o dimenzi N^{t}. Hodnota funkce ceny cíle je počítána jako vážený součet rozdílů mezi odpovídajícími prvky těchto vektorů.

Pokud definujeme t_k specifikaci cíle řečové jednotky k z posloupnosti požadovaných jednotek a u_k kandidáta na výběr jednotky z databáze, pak je cena cíle definována vztahem:

$$C^{\mathrm{t}}(t_k, u_k) = \sum_{j=1}^{N^{\mathrm{t}}} w_j^{\mathrm{t}} C_j^{\mathrm{t}}(t_k, u_k), \tag{4.1}$$

kde w_j^{t} určuje váhu dílčí ceny cíle, kterou lze nastavit buď manuálním způsobem, který je velmi subjektivní a často nepřesný, nebo lze použít některou z technik automatického nastavení (prohledávání váhového prostoru, regresní trénink). Detaily ke způsobu nastavení vah viz např. Hunt & Black, 1996 nebo Psutka *et al.*, 2006, str. 571 a 572.

Cena řetězení udává, jak dobře bude možné dvě realizace jednotek řetězit za sebe a značíme ji C^{c} dle anglického názvu *concatenation cost*[2)]. Čím se spektrální charakteristiky signálu na konci prvního a na začátku druhého segmentu více liší, tím je nutné provést větší změny, aby se řečový signál na hranici vyhladil, a také tím dochází ke zhoršení kvality řeči. Ideálním případem je situace, kdy jsou realizace přímými sousedy v původním řečovém korpusu, protože pak není třeba signál vůbec vyhlazovat a spojení je naprosto přirozené.

[2)] V některých publikacích je použit i jiný anglický termín *join cost*, případně odlišný český překlad *cena konkatenace*.

Cena řetězení se opět počítá jako vážený součet rozdílů vektoru příznaků, nyní však o dimenzi N^{c}, přičemž prvky vektoru jsou typicky tvořeny spektrálními charakteristikami a hodnotou F_0 řečového signálu na hranici řetězení. Problematikou konstrukce funkce ceny řetězení se zabývají např. práce Legát, 2012, Vepa *et al.*, 2002 nebo Blouin *et al.*, 2002. Obecný vztah pro výpočet hodnoty ceny řetězení má podobu:

$$C^{\mathrm{c}}(u_k, u_{k+1}) = \sum_{j=1}^{N^{\mathrm{c}}} w_j^{\mathrm{c}} C_j^{\mathrm{c}}(u_k, u_{k+1}). \tag{4.2}$$

Pomocí uvedených vztahů pro výpočet ceny cíle a ceny řetězení lze definovat vztah pro výpočet hodnoty kriteriální funkce (tj. celkové kumulativní ceny) celého řetězce řečových segmentů o délce K výrazem:

$$C(t_1^K, u_1^K) = \sum_{k=1}^{K} C^{\mathrm{t}}(t_k, u_k) + \sum_{k=1}^{K-1} C^{\mathrm{c}}(u_k, u_{k+1}). \tag{4.3}$$

Úkolem algoritmu výběru jednotek je nalézt optimální posloupnost segmentů řeči $\overline{u_1^K}$ z databáze ve smyslu minimální hodnoty kriteriální funkce 4.3, tedy nalezení řešení vztahu:

$$\overline{u_1^K} = \underset{u_1,\ldots,u_K}{\arg\min}\{C(t_1^K, u_1^K)\}. \tag{4.4}$$

Zde stojí za zmínku, že v případě, kdy řetězec vybraných segmentů odpovídá souvislému bloku (větě) z řečového korpusu, pak jsou ceny řetězení posloupnosti segmentů rovné nule.

4.1.1 Algoritmus hledání optimálního řetězce

Systémy obecné konkatenační syntézy většinou pracují s krátkými řečovými jednotkami[3)] (difony, trifony). Po úvodním zpracování vstupního textu a jeho převodu na fonetický přepis je pro každou řečovou jednotku vybrána množina odpovídajících realizací – kandidátů. Všechny vybrané kandidáty lze uspořádat

[3)] Základní řečovou jednotkou v systému ARTIC je difon.

do pomyslného orientovaného grafu, kde realizace tvoří uzly a přechody mezi nimi hrany. Z každého uzlu vedou pouze hrany do všech kandidátů řečových jednotek odpovídajících následující jednotce. Výsledný graf má orientované hrany, je acyklický a má ohodnocené jak hrany, tak i uzly (znázorněno na obrázku 4.1). Uzly grafu zastupují jednotlivé kandidáty řečových jednotek a je jim přiřazena cena cíle C^{t}. Hrany grafu reprezentují spojení dvou kandidátů s ohodnocením daném cenou řetězení C^{c}.

4.1.2 Viterbiův algoritmus

K nalezení optimálního řetězce kandidátů se nejčastěji využívá tzv. *Viterbiův algoritmus*, který vychází ze strategie prohledávání grafu do šířky. Tato technika postupuje v krocích 1 až K, kdy jeden krok odpovídá zpracování celé množiny kandidátů jedné řečové jednotky v syntetizované promluvě.

K popisu Viterbiova algoritmu bude zavedena funkce C^*, která odpovídá kumulativní ceně[4)] (angl. *cumulative cost*) nejlepší cesty, kterou se lze dostat do daného uzlu od první řečové jednotky posloupnosti. Dále označíme konkrétního kandidáta u s pořadím i v rámci množiny všech kandidátů řečové jednotky v kroku k jako $u_k(i)$ a počet kandidátů označíme $N(k)$.

Vztah pro výpočet C^* u konkrétního kandidáta pak zapíšeme takto:

$$C^*(u_k(i)) = \begin{cases} C^{\mathrm{t}}(u_k(i)), & k = 1 \\ C^{\mathrm{t}}(u_k(i)) + C^*(u_k^*(i)) + C^{\mathrm{c}}(u_k^*(i), u_k(i)), & k > 1 \end{cases}, \tag{4.5}$$

kde má $u_k^*(i)$ význam *nejlepšího předchůdce*, kterým je myšlen kandidát z předchozího kroku $k-1$, jenž minimalizuje součet kumulativní ceny cesty od počátku grafu a cenu spojení s kandidátem $u_k(i)$. Předpis pro nalezení nejlepšího předchůdce je dán výrazem:

$$u_k^*(i) = \underset{j=1,\ldots,N(k-1)}{\arg\min} \{C^*(u_{k-1}(j)) + C^{\mathrm{c}}(u_{k-1}(j), u_k(i))\}. \tag{4.6}$$

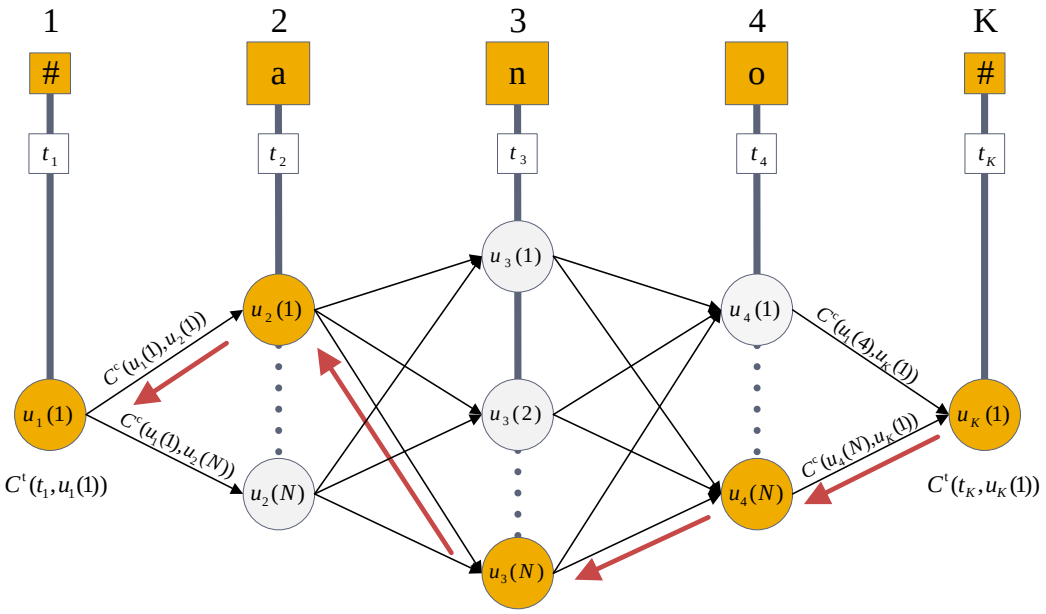
[4)] Kumulativní cena odpovídá součtu ohodnocení všech uzlů dané cesty (C^{t}) a přechodů mezi nimi (C^{c}).

Samotný algoritmus hledání optimální posloupnosti realizací řečových jednotek k řetězení lze popsat v následujících bodech:

1. Všichni kandidáti kroku $k = 1$ se ohodnotí cenou cíle C^{t}, a protože prvnímu kroku nepředchází žádný jiný krok, stanou se hodnoty C^{t} automaticky i hodnotami C^{*}.
2. Pro všechny kandidáty $u_k(i)$ s indexem $i = 1, \ldots, N(k)$ v kroku k (pro $k > 1$):
   (a) Vyhodnotí se cena cíle $C^{\mathrm{t}}(t_k, u_k(i))$.
   (b) Vypočte se cena řetězení C^{c} se všemi předchůdci z kroku $k - 1$.
   (c) Vyhledá se nejlepší předchůdce $u_k^*(i)$ z předchozího kroku podle vztahu 4.6, který minimalizuje kumulativní součet ceny k němu směřující nejlepší cesty grafem C^{*} a ceny přechodu C^{c}. Hodnota C^{*} uzlu grafu $u_k(i)$ je pak tvořena součtem jeho vlastní ceny cíle C^{t}, hodnoty C^{*} nejlepšího předchůdce a ceny C^{c} spojení kandidáta s nejlepším předchůdcem (viz vztah 4.5).
   (d) Ke kandidátovi $u_k(i)$ se uloží odkaz na nejlepšího nalezeného předchůdce $u_k^*(i)$.
3. Bod 2 se opakuje pro hodnoty $k = 2, \ldots, K$, tj. postupně se zpracují všechny podmnožiny kandidátů požadované posloupnosti jednotek.
4. V tomto okamžiku nesou všechny uzly grafu hodnotu C^{*} a uzly odpovídající krokům 2 až K i odkaz na nejlepšího předchůdce ve smyslu minimální ceny cesty grafem. Optimální cesta grafem je pak nalezena tak, že se nejprve vyhledá uzel z kroku K s nejnižší hodnotou C^{*}, a dále se zpětným prohledáním grafu přes odkazy na nejlepší předchůdce získá řetězec kandidátů, který minimalizuje cenu průchodu grafem.

4.1.2.1 Výpočetní náročnost Viterbiova algoritmu

Z hlediska náročnosti celého procesu vyhledání optimálních kandidátů je výpočet ohodnocení uzlu C^{t} relativně zanedbatelný, neboť se v něm používá



Obrázek 4.1: Znázornění prohledávaného grafu pro slovo „ano“. Jednotky označené symbolem # reprezentují pauzy před a za generovaným větným úsekem. Červené šipky představují odkazy na nejlepší předchůdce kandidátů.

pouze několik operací násobení nebo dělení, a hlavně je počet vyhodnocení roven počtu všech kandidátů. Ovšem vypočtení cen spojení kandidátů C^c je mnohem náročnější, neboť kromě provádění složitějších matematických operací (mocniny a odmocniny reálných čísel), musí být vypočteny pro každou hranu prohledávaného grafu[5)].

Výhodou základního Viterbiova algoritmu je to, že je úplný a vždy nalezne globálně minimální cestu z hlediska celkové (kumulativní) ceny. Hlavním problémem je však vysoká náročnost v řešené úloze, zejména u velkého řečového korpusu. Databáze řečových segmentů obsahuje v případě systému ARTIC celkem 670 757 segmentů a množina kandidátů pro každou řečovou jednotku může zahrnovat až tisíce reprezentantů. V druhém a dalším kroku algoritmu musí dojít pro každého kandidáta k postupnému výpočtu cen řetězení C^c se všemi kandidáty předchozího kroku. Pokud je například v jednom kroku zpracováváno 1000 uzlů a v množině kandidátů předchozího kroku jich je 500, pak je sice nutné vypočítat pouze 1000 hodnot C^t, ale počet vyhodnocení C^c

[5)] Algoritmická složitost základního Viterbiova algoritmu je $O(KN^2)$, kde K je počet jednotek syntetizované posloupnosti a N je počet kandidátů pro každou jednotku.

je $1000 \times 500 = 500\ 000$.

Proces výběru jednotek z databáze řečových segmentů tedy obecně spotřebuje největší část výpočetního času nutného k produkci syntézy řeči konkatenační metodou[6)] a z toho je většina využita k vyhodnocení cen řetězení. Příkladem může být i měření získané při prvotních experimentech na optimalizaci algoritmu výběru jednotek v systému ARTIC, kde například při syntéze věty zhruba o padesáti jednotkách dochází při použití základního Viterbiova algoritmu k vyhodnocení přibližně třiceti miliónů cen řetězení.

Ukazuje se tedy, že pokud by se podařilo výrazně omezit nutnost provádět tak obrovské množství výpočtů C^c, pak by došlo k významnému snížení výpočetní náročnosti syntézy řeči. Tím by se dosáhlo výrazného urychlení a minimalizovaly by se tak nevýhody konkatenační syntézy uvedené v úvodu této práce. Možnostem optimalizace a popisu přístupů k této problematice je věnován celý oddíl 6.1 následující kapitoly.

4.1.3 Ostatní algoritmy

K nalezení optimální (suboptimální) cesty orientovaným grafem popsaným v oddíle 4.1.1 lze použít i jiné algoritmy, ale pro řešení tohoto problému v úloze syntézy řeči výběrem jednotek se používají minimálně. Jako příklad si můžeme uvést Dijkstrův algoritmus, představený v práci Dijkstra, 1959, nebo postup popsaný v Taylor & Black, 1999, který využívá binární rozhodovací stromy[7)] a snaží se nalézt nejdelší souvislé úseky z řečového korpusu odpovídající specifikaci cíle.

[6)] Např. v práci Beutnagel *et al.*, 1999 se na základě měření na systému vyvíjeném společností AT&T uvádí, že výběr optimální posloupnosti zástupců jednotek zabíral 78 % výpočetního času celé syntézy.

[7)] Používá se zkratka CART z angl. *Classification And Regression Tree.*

Kapitola 5

Cíle práce

Přístupů k řešení syntézy řeči existuje celá řada, nicméně dnes se v reálných aplikacích nejčastěji využívá syntéza řetězením neboli konkatenační syntéza. Předkládaná disertační práce se konkrétně zabývá metodou konkatenační syntézy s výběrem jednotek. Ta, jak bylo popsáno v předchozích kapitolách, využívá rozsáhlou databázi, která může obsahovat desítky až stovky tisíc pečlivě připravených segmentů reálné lidské řeči. Princip syntetizéru je takový, že po zpracování vstupního textu a jeho převodu do fonetické podoby se vyberou z databáze nejvhodnější odpovídající realizace řečových jednotek a následně se spojí (zřetězí) do výsledné promluvy. Vzhledem k obrovskému počtu možných kombinací, jak řečové segmenty zřetězit do výsledné promluvy, je proces výběru optimální sekvence nejnáročnější část systému a spotřebuje většinu (více než 80 %[1)]) celkového výpočetního času syntézy řeči z textu.

Tato práce je zaměřena na snížení výpočetních nároků TTS systémů na bázi konkatenační syntézy. Například u systému ARTIC dosahuje rychlost generování řeči z textu pomocí původního algoritmu 0,7 násobek reálného času výsledné promluvy na průměrně vybaveném osobním počítači. V případě nasazení syntézy řeči jako součásti serverového řešení (např. u rozsáhlejšího dialogového systému) nebo při provozu na zařízení se slabším výpočetním výkonem (např. mobilní telefon, PDA) by byla uvedená rychlost překážkou.

Hlavním cílem je urychlení procesu výběru jednotek, resp. minimalizace množství nutných výpočtů cen řetězení. V rámci práce je zkoumáno a porovnáváno několik algoritmů, z nichž některé vznikly na základě existujících a publikovaných modifikací nejpoužívanějšího Viterbiova algoritmu. Dále jsou

[1)] Blíže viz statistické údaje v oddíle 7.2.

zde zastoupeny i zcela originální algoritmy, jejichž společnou základní myšlenkou je pokusit se využít větších celků, než jsou jednotlivé jednotky. Pokud totiž lze z jednotek připadajících v úvahu pro syntézu dané promluvy složit řetězec odpovídající části původní souvislé promluvy z řečového korpusu, pak lze pro něj vyvodit některé předpoklady a vlastnosti, jež by mělo být možné využít k optimalizaci výběru jednotek a snížení výpočetních nároků. Jednou z předností takových řetězců je i to, že zřetězení v nich obsažených jednotek za sebe je naprosto bezproblémové (řečové segmenty není nutné vůbec nijak upravovat) a nehrozí vznik rušivých zvuků nebo nepřirozených skoků frekvence hlasu (artefaktů) v místě spojení jednotek.

Z pozorování a statistik vytvořených na velkém množství syntetizovaných vět se navíc zcela jasně ukazuje, že při dostatečně obsáhlé databázi řečových segmentů jsou optimální sekvence jednotek převážně složeny právě z různě dlouhých řetězců z původní promluvy a samostatné jednotky jsou zastoupeny zcela minimálně. Přestože navržené metody založené na těchto řetězcích pracují s delšími jednotkami, nejde o snahu využít model řeči na úrovni slabik nebo celých slov podobně jako např. v publikacích Taylor & Black, 1999 nebo Erdem *et al.*, 2002. Na rozdíl od nich pracují zde prezentované algoritmy se všemi, tedy i jednotlivými, řečovými jednotkami, pouze je jejich strategií využít přímo sousedící segmenty z řečového korpusu ve prospěch zrychlení výběru jednotek (bez ohledu na to, zda daný řetězec tvoří nějaký fonetický celek jakou je slabika nebo slovo). Zde je vhodné i poznamenat, že výše citované publikace se ani nezabývají přímo urychlením výběru jednotek, ale jsou zaměřeny na zvýšení kvality řeči.

U syntetické promluvy generované systémem na bázi konkatenační syntézy řeči hrozí vznik artefaktů v důsledku nepřirozeně velkých skoků frekvence základního hlasivkového tónu F_0. Toto riziko lze předpokládat ve zvýšené míře právě na hranici dvou delších řečových segmentů z původního řečového korpusu. V práci Conkie & Syrdal, 2011 je popsána metoda optimalizace výběru jednotek, která omezuje možnost zřetězení dvou jednotek pouze na ty, u nichž se frekvence F_0 na konci první a začátku následující realizace jednotky neliší o více jak stanovenou hodnotu. Tato technika prořezávání (předvýběru) jednotek by kromě efektu urychlení, díky snížení počtu možných spojení, mohla přinést i odstranění nerovností frekvence F_0. Jedním z dalších cílů práce je experimentálně ověřit efekt prořezávání jednotek na základě frekvence F_0 u všech zkoumaných algoritmů, a to jak z pohledu rychlosti, tak

i možného přínosu ve zvýšení kvality výsledné řeči.

Dalším problémem snižujícím kvalitu výstupu jsou artefakty vznikající nevhodnou délkou některé z vybraných jednotek projevující se jako určité zakolísání v promluvě. V systému ARTIC není zatím pracováno s žádným modelem trvání, který by délku jednotek korigoval, ale je zde používána metoda hodnocení kandidátů jednotek, která bere v úvahu kromě jiného i pozici jednotky v původní větě v řečovém korpusu. Tím se upřednostňují na danou pozici řečové segmenty, které by měly svou délkou a dalšími vlastnostmi implicitně vyhovět. Vedlejším cílem práce je i pokusit se vhodným způsobem omezit použití jednotek nevhodné délky. Touto cestou by kromě očekávaného efektu zvýšení kvality mohlo opět dojít i ke zvýšení rychlosti díky snížení počtu kandidátů jednotek.

Jak již bylo uvedeno, je databáze řečových jednotek poměrné rozsáhlá a zabírá nemalou paměťovou kapacitu, což může být u zmiňovaných mobilních zařízení problematické i dnes. Jedním z možných přístupů jak tento problém řešit, je vhodný předvýběr, resp. prořezání jednotek a jejich úplné odstranění z používané databáze. Samozřejmě s nižším počtem jednotek se zvyšuje pravděpodobnost zhoršení kvality syntetické řeči, nicméně zvolením vhodného kritéria jaké jednotky odstranit, se toto riziko minimalizuje. Zároveň lze i důvodně předpokládat, že odstraněním jednotek z databáze se snižuje pravděpodobnost výskytu řetězců jednotek, které se vyskytují vedle sebe v řečovém korpusu. V rámci systému ARTIC je vytvořen mechanismus, jenž dokáže odstranit požadované množství jednotek z databáze řečových segmentů (detaily viz Tihelka, 2007), a cílem experimentů bude i ověření toho, jak se redukce počtu možných jednotek projeví u porovnávaných algoritmů.

Ověření jednotlivých zkoumaných algoritmů a jejich kombinací nastavení parametrů je časově velmi náročné. Z tohoto důvodu byly všechny experimenty a srovnání prováděny v systému ARTIC za použití jedné databáze řečových segmentů vytvořené z řečového korpusu nahraného mužským hlasem (databáze obsahuje 670 757 segmentů). Až následně byly nejlepší varianty algoritmů testovány i nad srovnatelně obsáhlou databází řečových segmentů pro ženský hlas (656 406 segmentů).

Shrnutí cílů

Hlavní cíle této disertační práce lze shrnout v následujících bodech:

1. Analýza stávajících možností a technik optimalizace výběru jednotek.
2. Experimentální ověření optimalizací základního Viterbiova algoritmu.
3. Analýza výskytu řetězců s nulovou cenou řetězení v promluvách vygenerovaných TTS systémem.
4. Návrh a experimentální ověření algoritmů založených na řetězcích s nulovou cenou řetězení.
5. Analýza výskytu artefaktů ve vygenerovaných promluvách.
6. Návrh a experimentální ověření technik vedoucích k odstranění výskytu artefaktů.
7. Ověření výkonnosti algoritmů pro další hlas řečníka (ženský) a pro redukované databáze řečových segmentů.
8. Srovnání a vyhodnocení všech vytvořených algoritmů a jejich variant.

Kapitola 6

Přehled optimalizací procesu výběru jednotek

Tato kapitola je věnována přehledu a rozboru publikovaných přístupů a technik, jež se přímo zabývají nebo určitým způsobem souvisí s optimalizací procesu výběru řečových jednotek.

6.1 Optimalizace procesu výběru jednotek

Nejčastěji se k vyhledání optimální posloupnosti řečových segmentů využívá Viterbiův algoritmus, který byl popsán v oddíle 4.1.2. Ve své podstatě se jedná o prohledávání do šířky ve stavovém prostoru reprezentovaném grafem, kde jednotlivé uzly představují kandidáty a hrany spojení mezi nimi.

6.1.1 Optimalizace Viterbiova algoritmu prořezáváním grafu

V práci Sakai *et al.*, 2008 bylo představeno několik modifikací základního Viterbiova algoritmu, které vedou k omezení počtu nezbytných vyhodnocení cen spojení mezi kandidáty, a tím k urychlení výběru jednotek (resp. snížení výpočetní náročnosti). Vedlejším efektem některých z úprav je to, že nemusí být vždy dosaženo globálního minima celkové ceny vybraného řetězce realizací řečových jednotek a mohlo by tak dojít ke zhoršení kvality syntetizované promluvy.

6.1.1.1 Minimální cena řetězení

Optimalizace představené v publikaci[1)] vycházejí ze znalosti minimální hodnoty ceny řetězení pro každé dvě řečové jednotky syntetizovaného řetězce – tj. minima všech cen řetězení C^{c} mezi všemi jejich kandidáty. Toto minimum pro kandidáty sousedních jednotek v krocích $k-1$ a k budeme značit jako ${}_{\min}C^{\mathrm{c}}_{k-1,k}$ definované výrazem:

$$ {}_{\min}C^{\mathrm{c}}_{k-1,k} = \min_{m,n}(C^{\mathrm{c}}(u_{k-1}(m), u_k(n))), \quad m = 1, \ldots, N(k-1), n = 1 \ldots, N(k), \tag{6.1} $$

kde $N(k-1)$ a $N(k))$ jsou počty kandidátů odpovídajících krokům $k-1$ a k.

Získat všechny hodnoty ${}_{\min}C^{\mathrm{c}}_{k-1,k}$ znamená nejprve vypočítat hodnoty cen řetězení C^{c} mezi úplně všemi kandidáty[2)] všech možných realizací v databázi řečových segmentů (resp. všech přípustných kombinací daného jazyka) a pro každou dvojici si uložit minimum. Tento proces může být velmi náročný, ale na druhou stranu je potřeba jej provést pouze jednou.

U řečových segmentů, jež spolu v řečovém korpusu přímo sousedí v původních namluvených větách, je vždy hodnota řetězení rovna nule – a tedy i minimální hodnota ${}_{\min}C^{\mathrm{c}}_{k-1,k}$ je nulová.

V případě nulové minimální hodnoty cen řetězení dvou jednotek ${}_{\min}C^{\mathrm{c}}_{k-1,k}$ vedou dále uvedené optimalizace k urychlení algoritmu, avšak čím je tato hodnota vyšší, tím je pro urychlení algoritmu cennější a přispívá k většímu urychlení.

6.1.1.2 Modifikace č. 1: Optimalizace při vyhledávání nejlepšího předchůdce kandidáta

Při vyhledávání nejlepší cesty mezi kandidáty řečových jednotek syntetizované promluvy se v každém kroku vyhledává pro každého kandidáta nejlepší

[1)] V publikaci Sakai *et al.*, 2008 se používá odlišný způsob ohodnocení uzlů a přechodů grafu (viz oddíl 4.1.1), a to za pomoci skóre (lepší je vyšší hodnota). Aby byl text této práce konzistentní, byl popis optimalizací upraven a přepsán s využitím cen, které mají opačný význam (lepší je nižší hodnota).

[2)] Pokud by byly vypočteny pouze některé kombinace, například u nejčastěji se vyskytujících dvojic řečových jednotek, pak se u neznámých kombinací použije minimální hodnota rovná nule.

předchůdce. Tento proces, který prochází v cyklu všechny kandidáty předchozího kroku a vyhodnocuje cenu řetězení, lze za určitých podmínek předčasně ukončit bez toho, že by byla ohrožena záruka nalezení globálního minima kumulativní ceny vybraného řetězce.

Předpokladem této optimalizace je to, že kandidáti řečové jednotky z předchozího kroku $k-1$ jsou seřazeni podle hodnoty jejich kumulativní ceny C^*. Hledání nejlepšího předchůdce pak probíhá postupně od kandidátů s nižšími hodnotami C^* směrem k hodnotám vyšším. Pro každého předchůdce z kroku $k-1$ je vždy vyhodnocena cena řetězení C^{c} u přechodu ke kandidátovi v kroku k, tak jako u základního Viterbiova algoritmu, ale před tímto výpočtem se vždy prověří, zda není splněná podmínka pro ukončení cyklu. Pokud totiž platí, že součet ceny C^* kandidáta v kroku $k-1$ a minimální hodnoty ${}_{\min}C^{\mathrm{c}}_{k-1,k}$ je větší než dosud nalezené minimum součtu C^* kandidáta v kroku $k-1$ a ceny řetězení C^{c}, pak vzhledem k tomu, že kandidáti v kroku $k-1$ jsou seřazeni dle C^*, nemůže se již tento ani žádný další kandidát stát nejlepším předchůdcem. Proto lze při splnění této podmínky cyklus přerušit a přejít na vyhledávání nejlepšího předchůdce pro dalšího kandidáta z množiny kroku k.

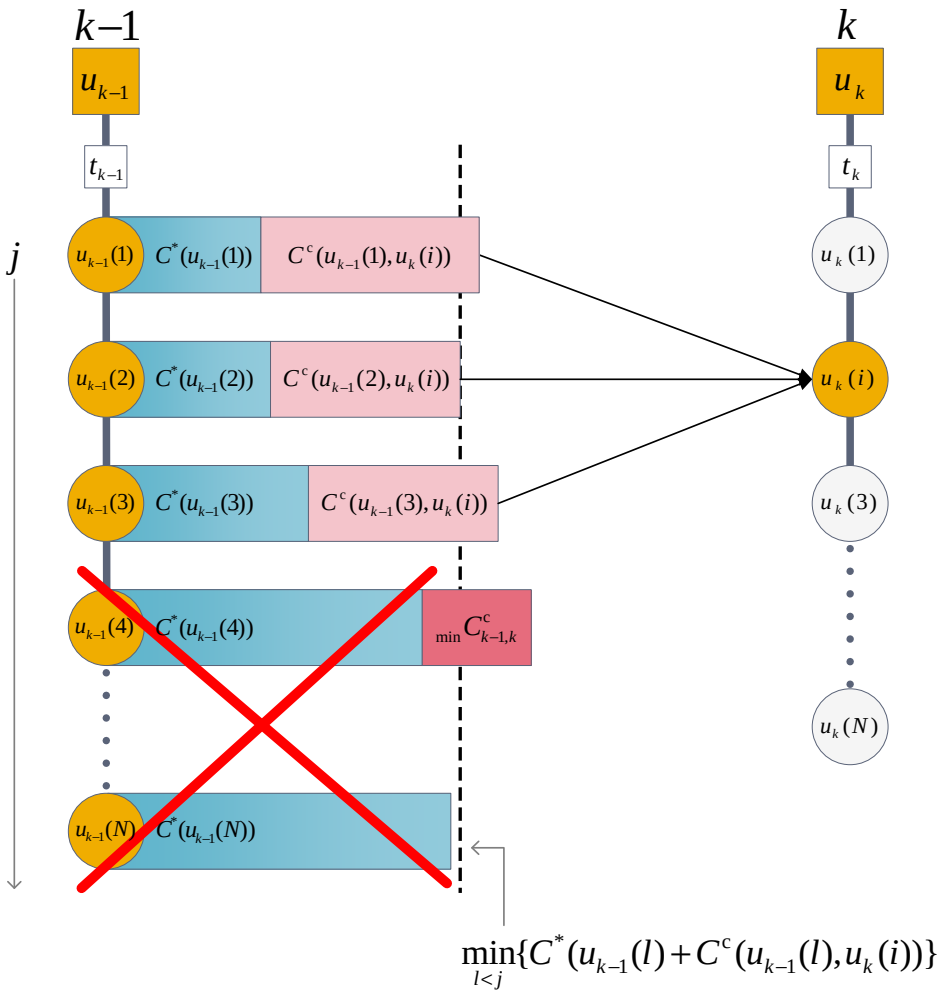
Popsaná modifikace odpovídá bodu č. 2b v popisu základního Viterbiova algoritmu uvedeného na straně 21, kde je řešen výběr nejlepšího předchůdce a lze ji bodově zapsat takto (schéma viz obrázek 6.1):

1. Všichni kandidáti $u_{k-1}(j)$ předchozího kroku $k-1$ se seřadí vzestupně podle kumulativní ceny nejlepší cesty $C^*(u_{k-1}(j))$.
2. Vyhodnotí se cena řetězení $C^{\mathrm{c}}(u_{k-1}(1), u_k(i))$ mezi prvním kandidátem $u_{k-1}(1)$ z předchozího kroku a zpracovávaným kandidátem $u_k(i)$ kroku k. Cena se sečte s kumulativní cenou nejlepší cesty $C^*(u_{k-1}(1))$ až do uzlu $u_{k-1}(1)$ a uchová se jako minimum ${}_{\min}C^*_k$.
3. Pro $j = 2, \ldots, N(k-1)$:
   (a) Pokud je splněna podmínka $C^*(u_{k-1}(j)) +_{\min} C^{\mathrm{c}}_{k-1,k} >_{\min} C^*_k$, pak se cyklus ukončí a přechází se na bod č. 4.
   (b) V opačném případě se vyhodnotí cena řetězení mezi kandidátem $u_{k-1}(j)$ z předchozího kroku a zpracovávaným kandidátem $u_k(i)$

kroku k.

(c) Pokud je součet ceny řetězení $C^{c}(u_{k-1}(1), u_k(i))$ a kumulativní ceny nejlepší cesty $C^{*}(u_{k-1}(j))$ do uzlu $u_{k-1}(j)$ menší než minimální hodnota ${}_{\min}C^{*}_{k}$, pak tento součet minimální hodnotu ${}_{\min}C^{*}_{k}$ nahradí.

4. V množině všech zpracovaných kandidátů předchozího kroku $k-1$ se vyhledá kandidát, přes kterého vede cesta s nejnižší kumulativní cenou do uzlu $u_k(i)$ – technicky se jedná o kandidáta, u něhož došlo naposledy k aktualizaci minimální hodnoty ${}_{\min}C^{*}_{k}$. Na závěr se k uzlu $u_k(i)$ uloží odkaz na nejlepšího kandidáta z kroku $k-1$.



Obrázek 6.1: Schéma prořezávání při vyhledávání nejlepšího předchůdce kandidáta (znázorněno pro kandidáta kroku k s indexem $i = 2$).

6.1.1.3 Modifikace č. 2: Prořezávání kandidátů po vyhodnocení kumulativní ceny

Úprava mění způsob hledání optimální cesty grafem z klasického úplného prohledávání do šířky (angl. *Breadth-First Search*, zkratka *BFS*) na hledání typu *BEAM* (angl. *Beam Search*), kdy se ve stavovém prostoru (grafu) zpracovává v každém kroku pouze určitý počet kandidátů s dosud nejlepším ohodnocením.

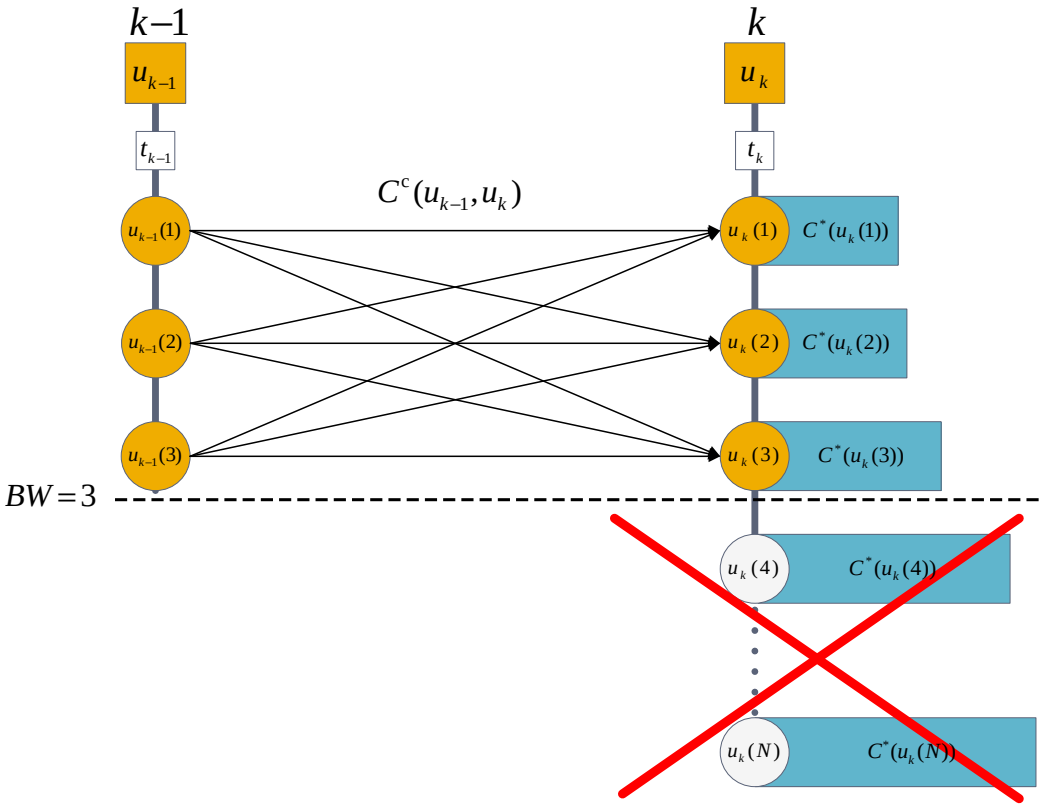
Algoritmus se zároveň stává neúplným a nezaručuje tak nalezení globálního optima, nicméně pokud se vhodně zvolí hranice pro omezení kandidátů (s ohledem na počet segmentů v databázi řečových jednotek a způsob využití syntetizéru), pak lze předpokládat zachování velmi dobré kvality generované řeči.

Základním prvkem této modifikace Viterbiova algoritmu je zavedení pevně stanovené hranice značené jako BW (z angl. *beam width*), která se uplatní na závěr zpracování všech kandidátů jednoho kroku. Po vyhodnocení kumulativní ceny nejlepší cesty $C^*(u_i(k))$ pro všechny uzly $u_k(i), i = 1, \ldots, N(k)$ kroku k se množina pro další zpracování omezí pouze na BW nejlepších uzlů a zbývající uzly s horší cenou se odstraní.

Oříznutí množiny kandidátů se tedy provádí na konci každého vnitřního cyklu bodu č. 2 v popisu základního Viterbiova algoritmu, který by byl rozšířen o položky 2e a 2f se zněním:

> 2. Pro všechny kandidáty s indexem $i = 1, \ldots, N(k)$ v kroku k (pro $k > 1$):
>
> (a) – (d) *Algoritmus je v těchto částech shodný s vnitřním algoritmem bodu č. 2 na straně 22.*
>
> (e) Kandidáti se seřadí podle kumulativní ceny nejlepší cesty $C^*(u_i(k))$ vzestupně.
>
> (f) V množině kandidátů se ponechá pouze BW prvních realizací a zbývající jsou odstraněny.

Jak je patrné i z obrázku 6.2, má úprava ten význam, že po vyhodnocení kandidátů jednoho kroku dojde k omezení počtu kandidátů, a tím se zároveň odpovídajícím způsobem sníží počet potřebných vyhodnocení cen řetězení v kroku následujícím.



Obrázek 6.2: Schéma prořezávání po vyhodnocení kumulativních cen kandidátů.

6.1.1.4 Modifikace č. 3: Optimalizace při vyhodnocování kumulativních cen kandidátů

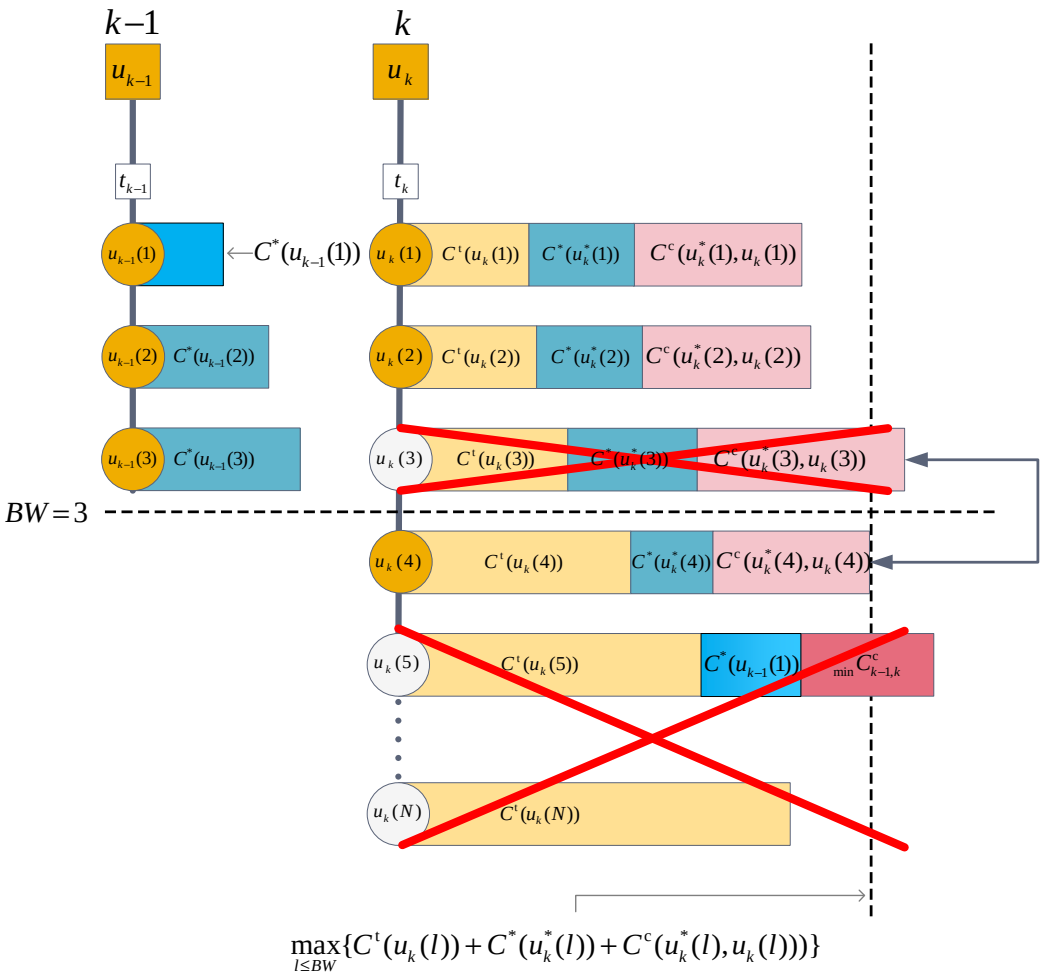
Cyklus základního Viterbiova algoritmu označený jako bod č. 3 v popisu na straně 21 zajišťuje postupné vyhodnocení kumulativní ceny C^* nejlepší cesty grafem od jeho počátku až k uzlům $u_k(i)$ v kroku k.

Poslední z optimalizací spočívá v tom, že tento cyklus lze za určitých podmínek zastavit a není nutné dále pokračovat, a to bez dalších ztrát v rámci již „ztrátového“ algoritmu, který ořezává kandidáty každého kroku užitím hranice BW.

Předpokladem je seřazení kandidátů kroku k podle ceny cíle $C^t(t_k, u_k(i))$ a znalost minimální hodnoty ceny řetězení ${}_{\min}C^c_{k-1,k}$ mezi všemi kandidáty z kroků $k-1$ a k. Cyklus probíhá standardním způsobem, dokud počet zpracovaných uzlů nedosáhne hranice BW, tj. ke každému uzlu $u_k(i)$ se

nalezne nejlepší předchůdce a vypočte se kumulativní cena $C^*(u_k(i))$.

Poté cyklus pokračuje, ale pouze do splnění podmínky, která zaručí, že již není možné najít dalšího kandidáta daného kroku, který by byl zařazen do množiny BW nejlepších. Pokud totiž součet hodnoty ceny cíle C^{t} zkoumaného kandidáta, hodnoty minimální ceny řetězení ${}_{\min}C^{\mathrm{c}}_{k-1,k}$ a nejnižší hodnoty kumulativní ceny ${}_{\min}C^*_k$ předchozího kroku $k-1$ je větší než dosud nejhorší nalezená kumulativní cena kandidátů kroku k, pak ani zkoumaný a ani žádný další kandidát kroku k již nemůže mít takovou hodnotu kumulativní ceny, která by jej posunula mezi BW nejlepších kandidátů. Schéma této optimalizace je naznačeno na obrázku 6.3.



Obrázek 6.3: Optimalizace při vyhodnocování kumulativních cen kandidátů.

Zápis algoritmu v bodech – vložení nového bodu za bod č. 2a v původním popisu základního Viterbiova algoritmu:

(b) Pokud je počet zpracovaných kandidátů (hodnota i) větší než BW, pak:

i. Seřadit kandidáty u_{k-1} v kroku $k-1$ vzestupně podle kumulativní ceny C^*, takže kumulativní cena prvního kandidáta $u_{k-1}(1)$ bude nejnižší ze všech kandidátů kroku $k-1$.

ii. Pokud platí $\{C^{\mathrm{t}}(t_k, u_k(i)) + C^*(u_{k-1}(1)) +_{\min} C^{\mathrm{c}}_{k-1,k}\} > \max_{j<BW}\{C^*(u_k(j))\}$, pak:

A. Seřadit dosud vyhodnocené kandidáty kroku k podle kumulativní ceny C^*.

B. Odstranit z množiny pro další zpracování kandidáty $u_k(i)$, pro které platí $i > BW$.

C. Ukončit vyhodnocování dalších kandidátů kroku k a přejít na bod č. 4 původního algoritmu.

6.1.1.5 Zhodnocení modifikací Viterbiova algoritmu

V předchozím textu byly popsány tři úpravy Viterbiova algoritmu, které byly původně publikovány v článku Sakai *et al.*, 2008. Autoři uvádějí, že experimenty s optimalizací výběru jednotek prováděli nad databází řečových segmentů, která byla vytvořena z řečového korpusu o rozsahu 1132 promluv v celkové délce 50 minut. Po implementaci modifikací dospěli měřením na vzorové větě ke snížení počtu vyhodnocených kandidátů až přibližně o 70 % při nastavení $BW = 50$, jak je uvedeno v tabulce 6.1.

Na závěr je nutné připomenout, že implementací druhé a třetí popsané úpravy se stává algoritmus neúplným, tj. nezaručuje nalezení globálního minima kumulativní ceny. Je pak otázkou, jak bude ovlivněna kvalita syntetizované řeči, protože autoři žádné testy v tomto směru neuvádějí. Je zcela zřejmé, že pravděpodobnost zhoršení kvality promluvy závisí na volbě hranice BW a pro velmi malé hodnoty bude algoritmus velmi rychlý, ale snížení kvality řeči může být již výrazné.

Tabulka 6.1: Výsledky měření počtu vyhodnocených kandidátů při použití modifikací Viterbiova algoritmu

BW	Celkový počet kandidátů	Počet vyhodnocených kandidátů	Úspora [%]
2 000	37 545	37 142	1,1
500	37 545	27 817	25,9
200	37 545	20 334	45,8
50	37 545	12 061	32,1

Výhodou optimalizací je také velmi snadná změna parametru BW, což by se dalo využít například u serverového řešení, kdy by se hodnota BW mohla dynamicky snížit nebo zvýšit v závislosti na vytížení celého systému. Při vysoké zátěži by se sice zvýšilo riziko zhoršení kvality generované řeči, ale nedocházelo by k prodlevám nebo výpadkům.

6.1.2 Využití mezipaměti cen řetězení

Jedna z cest jak urychlit algoritmus výběru jednotek je popsána v publikaci Beutnagel *et al.*, 1999 a její podstatou je využití mezipaměti (angl. *cache*) obsahující hodnoty cen řetězení mezi všemi možnými řečovými segmenty. S ohledem na počet vyhodnocení při výběru optimálního řetězce řečových segmentů není samotný výpočet ceny řetězení mezi dvěma realizacemi řečových jednotek zanedbatelný. Pokud by byly hodnoty již dopředu vypočítané a dostupné v nějaké databázi s rychlým přístupem, pak by bylo možné tímto způsobem snížit výpočetní náročnost algoritmu, zejména při použití rozsáhlé databáze řečových segmentů. Tato optimalizace tedy primárně neřeší omezení počtu nutných vyhodnocení cen řetězení, ale snížení výpočetní náročnosti samotných výpočtů při výběru jednotek v reálném čase.

V uvedené práci jsou experimenty prováděné na systému vytvořeném pro anglický jazyk s databází řečových segmentů obsahující 84 tisíc difonů. Počet všech možných spojení mezi segmenty tak čítá 1,76 miliardy kombinací. Ne všechny kombinace jednotek však dávají z hlediska jazyka smysl, takže reálný počet možných spojení bude nižší, ale i tak se bude jednat o velmi vysoké číslo.

Autoři se rozhodli ověřit možnost použít mezipaměť pouze s omezenou

podmnožinou cen spojení u nejčastěji používaných dvojic segmentů, protože vypočítat dopředu všechny kombinace by nebylo praktické, a to jak z hlediska doby vyhodnocení cen, tak i kvůli problémům s uchováním a efektivním vyhledáváním hodnot v mezipaměti. Provedli nejprve syntézu 10 tisíc vybraných vět z novin, při které bylo vyhodnoceno celkem 50 miliónů cen řetězení, z nichž získali zhruba 1,2 milionu unikátních spojení realizací jednotek, a ty uložili do mezipaměti. Následně si připravili testovací vzorek 8 tisíc (odlišných) vět z novin a zjišťovali pokrytí potřebných cen řetězení v mezipaměti. Zjistili tak, že jimi vytvořená mezipaměť pokrývá 99 % všech spojení na množině testovacích vět.

Na základě předchozích zjištění byl upraven algoritmus výběru jednotek tak, že využíval mezipaměť cen řetězení vytvořenou z původních 10 tisíc vět. Ta obsahovala zmíněných 1,2 milionu cen řetězení, což odpovídá pouze 0,7 % z teoretických 1,76 miliardy všech možných spojení.

6.1.2.1 Implementace mezipaměti

Úspěšnost optimalizace využitím mezipaměti s cenami řetězení také závisí velmi výrazně na efektivitě nalezení hodnoty pro příslušné dva segmenty řečových jednotek, tj. způsobu implementace mezipaměti. Autoři použili hašovací tabulku (angl. *hash table*), která je tvořena pomocí dvojic *klíč – hodnota*, kde klíč reprezentuje index záznamu v tabulce. Vyhledávání je urychleno tím, že se definuje vhodná tzv. *hašovací funkce* (angl. *hash function*), jejímž vstupem je v tomto případě identifikace obou sousedních segmentů a výstupem je odpovídající klíč. Na základě získaného klíče lze v tabulce velmi rychle vyhledat záznam s hodnotou. Ideální je taková hašovací funkce, která pro množinu dat v tabulce vrací unikátní index a zároveň nevytváří prázdná místa bez hodnot (tzv. *řídká tabulka*).

V prezentovaném článku byla nejprve použita obecná hašovací funkce, která nezaručovala na výstupu unikátní hodnotu pro všechny vstupy. Tento způsob lze také použít, ale algoritmus hledání hodnot v tabulce pak musí počítat s tím, že klíč reprezentuje množinu více hodnot, v níž se teprve vyhledá konkrétní požadovaná hodnota. To znamená, že rychlost vyhledání hodnot je nižší než při použití ideální hašovací funkce (angl. *perfect hash*).

První verze optimalizace s využitím mezipaměti cen řetězení dosáhla 3,5 násobného urychlení v porovnání s původním algoritmem výběru jednotek.

Autoři následně na základě poznatků z publikace (Tarjan & Yao, 1979) definovali novou hašovací funkci, která vracela pouze unikátní klíče pro všechny uchované hodnoty, a tím došlo k dalšímu urychlení. Výsledný algoritmus pak zvýšil rychlost výběru jednotek o další 2,5 násobek, takže celkově došlo k 8,75 násobnému urychlení vůči původnímu algoritmu bez použití mezipaměti.

6.1.2.2 Zhodnocení optimalizace pomocí mezipaměti s hodnotami cen řetězení

Pomocí experimentů na testovacích datech autoři ověřili, že použitím mezipaměti obsahující nejčastěji se vyskytující dvojice řečových segmentů lze dosáhnout významného urychlení i s poměrně malou množinou předpočítaných hodnot. Autoři uvádějí, že pomocí mezipaměti obsahující pouhých 0,7 % všech možných cen spojení dosáhli přibližně 9 násobného urychlení při výběru optimální sekvence řečových segmentů.

Zajímavostí na publikovaném přístupu je i řešení případů, kdy není cena řetězení mezi dvěma segmenty součástí mezipaměti. V takovém případě se místo skutečného výpočtu použila jako cena řetězení konstantní vysoká hodnota. Předpokladem tohoto zjednodušení je to, že pár realizací řečových jednotek s nízkou frekvencí výskytu by stejně byl pravděpodobně penalizován, zároveň ale není zcela vynechán z výběru. Toto zjednodušení sice vede k dalšímu urychlení, avšak znamená také ztrátu záruky nalezení globálně optimálního řetězce vybraných jednotek. Projev zjednodušení na kvalitě produkované syntetické řeči byl ověřen matematicky i pomocí poslechových testů. V prvním případě bylo na množině 8 tisíc testovacích vět zjištěno, že 98,2 % všech vybraných segmentů je shodných s výběrem podle základního Viterbiova algoritmu zaručujícího nalezení globálního minima. Poslechové testy pak potvrdily, že ke snížení kvality generované řeči téměř nedošlo.

Nevýhodou využití mezipaměti cen řetězení je kromě nutnosti předpočítat velké množství hodnot také nezanedbatelný nárůst paměťového prostoru potřebného k uchování dat pro syntézu. V publikaci Čepko *et al.*, 2008, kde je mezipaměť cen řetězení také použita, se uvádí nárůst až o 21,9 %. Proto je také otázkou, zda by bylo vhodné použít tuto optimalizaci v případě nasazení systému u mobilních zařízení s omezenou kapacitou paměti.

6.1.3 Omezení množiny realizací jednotek před vyhodnocením cen cíle na základě kontextu

Základní Viterbiův algoritmus v každém kroku vybere pro příslušnou řečovou jednotku všechny vhodné kandidáty a pokračuje tím, že pro ně vyhodnotí cenu cíle C^{t}. I když je v oddíle 4.1.2.1 uvedeno, že výpočet ceny cíle není z pohledu výpočetní náročnosti až tak významný, byla v článku Conkie *et al.*, 2000 publikována metoda, která si klade za cíl omezit výpočetní náročnost předvýběrem kandidátů ještě před vyhodnocením C^{t}. Tím samozřejmě dojde i k následnému efektu urychlení díky snížení počtu nutných vyhodnocení cen řetězení C^{c}.

Autoři vycházejí z předpokladu, že *kontext* sousedících jednotek hraje významnou roli při výběru vhodného kandidáta. Tato optimalizace zavádí pomocnou hodnoticí funkci ${}_{pre}C^{\mathrm{t}}(u_k)$, jež kandidátům přiřadí cenu určenou na základě kontextu okolních jednotek požadované promluvy. Z článku není zcela patrné, jak je přesně pomocná hodnoticí funkce definována, nicméně základní princip je takový, že se vypočítává na základě kontextu okolních jednotek a větší váhu při vyhodnocení mají bližší sousední jednotky, protože obecně vliv okolních jednotek na vlastnosti aktuálně zpracovávané jednotky klesá se zvětšující se vzdáleností (například díky koartikulačním jevům).

Implementace této optimalizace byla provedena tak, že nejprve bylo vybráno 10 tisíc nejčastěji použitých trifonů, tj. sekvencí u_j–u_k–u_l. Pro všechny realizace jednotek u_k byla předpočítána hodnota ${}_{pre}C^{\mathrm{t}}(u_k)$, která zohledňuje kontext dvou předchozích a následujících segmentů v původním řečovém korpusu. Následně bylo pro každý kontext (trifon) vybráno n nejlepších kandidátů a vloženo do pomocné množiny S_{jkl}. Urychlení výběru jednotek v čase syntézy řeči pak spočívá v omezení kandidátů zpracovávané jednotky u_k pouze na ty, které odpovídají stejnému kontextu okolních jednotek a nacházejí se v množině S_{jkl}. Pokud by v omezující množině nebyly pro daný kontext nalezeny žádné realizace, pak se použije úplná množina kandidátů stejně jako bez použití optimalizace.

Použitá úprava algoritmu snížila podle autorů výpočetní nároky syntézy řeči o 30 % ve srovnání se základním Viterbiovým algoritmem. Z hlediska posouzení kvality řeči, uvádějí autoři, že výstupy optimalizovaného syntetizéru byly identické s jednotkami vybranými základním Viterbiovým algoritmem, a proto ani neprováděli poslechové testy. V následující tabulce 6.2 jsou uvedeny

prezentované výsledky experimentu s touto optimalizací.

Tabulka 6.2: Výsledky experimentů s omezením množiny realizací jednotek na základě kontextu

	Základní Viterbiův algoritmus	Optimalizovaný algoritmus
Celkový počet kandidátů	212 miliónů	59 miliónů
Počet vyhodnocených cen řetězení	333 miliónů	225 miliónů
Celkový čas syntézy	1	0,7

6.1.4 Omezení množiny realizací jednotek na základě četnosti jejich výskytu

V článku Conkie *et al.*, 2000 byla představena i druhá metoda, jak omezit množinu realizací dané jednotky v reálném čase. Jedná se prakticky o stejný postup, který je zmíněn v kapitole 3.2 popisující předvýběr jednotek do databáze řečových segmentů, pouze s tím rozdílem, že nedochází k úplnému (fyzickému) odstranění méně vhodných jednotek z databáze řečových segmentů předvýběrem v přípravné fázi vývoje TTS systému.

S odkazem na experimenty v publikaci Beutnagel *et al.*, 1999, kde bylo ukázáno, že i velmi malá mezipaměť cen řetězení pokrývá až 98,5 % všech spojení mezi segmenty, implementovali autoři optimalizaci založenou na znalosti četnosti výskytu trifonů v množině testovacích dat. Nejprve provedli syntézu velkého množství vět tvořených množinou celkem 25 miliónů realizací řečových jednotek. Z této množiny následně extrahovali všechny sekvence u_j–u_k–u_l, jež se zároveň vyskytovaly i v původním korpusu. Tím bylo získáno přibližně 30 tisíc sekvencí, které byly následně uchovány v pomocné databázi společně se seznamem všech řečových segmentů použitých při syntéze testovacích vět. Seznam tak prakticky obsahoval realizace, jež byly nejčastěji použité v kontextu okolních jednotek.

Algoritmus pak do množiny kandidátů zahrnul pouze realizace uvedené v připraveném seznamu nejčastěji se vyskytujících trifonů. V případě, že se jednotka v seznamu nenacházela, byl použit standardní mechanismus základního Viterbiova algoritmu.

Urychlení výběru jednotek za pomoci této metody udávají autoři zhruba dvojnásobné a na základě poslechových testů vychází, že kvalita generované syntetické řeči se při srovnání se základním Viterbiovým algoritmem nezhoršila. Pro úplnost je v tabulce 6.3 uvedené srovnání se standardním algoritmem bez použití optimalizace.

Tabulka 6.3: Výsledky experimentů s omezením množiny realizací jednotek na základě četnosti jejich výskytu

	Základní Viterbiův algoritmus	Optimalizovaný algoritmus
Celkový počet kandidátů	212 miliónů	14 miliónů
Počet vyhodnocených cen řetězení	333 miliónů	112 miliónů
Celkový čas syntézy	1	0,5

6.1.5 Omezení počtu vyhodnocení cen řetězení s využitím F_0

V článku Conkie & Syrdal, 2011 je představen heuristický postup jak omezit počet vyhodnocení cen řetězení C^{c} mezi kandidáty s využitím znalosti o průběhu základní frekvence F_0 u jednotlivých segmentů. Autoři uvádějí, že za pomoci experimentů na množině testovacích vět zjistili, že průměrné hodnoty F_0 u řečových segmentů z optimální sekvence se ve většině případů liší o méně než 50Hz. Také je faktem, že čím více se F_0 u dvou segmentů liší, tím je jejich spojení více problematické a hrozí výskyt nespojitostí slyšitelných posluchačem.

Publikovaná metoda optimalizace Viterbiova algoritmu (tj. procesu výběru jednotek) je založena na myšlence, že lze bez významné ztráty kvality řeči omezit množinu kandidátů v kroku k tak, že se při hledání nejlepšího předchůdce vyhodnotí cena spojení pouze s kandidáty kroku $k-1$ s určeným minimálním rozdílem frekvence ${}_{\max}\Delta F_0$. Ostatní kandidáti se vynechají, protože s vysokou pravděpodobností nepatří mezi nejvhodnější segmenty k řetězení s realizací jednotky v kroku $k-1$.

6.1.5.1 Implementace optimalizace

Autoři implementovali optimalizaci úpravou základního Viterbiova algoritmu v bodě, v němž se vyhodnocuje nejlepší předchůdce pro segment $u_k(i)$ řečové jednotky u_k v kroku k. Kandidáti předchozího kroku byli nejprve seřazeni podle velikosti hodnoty F_0 kvůli rychlejšímu vyhledání vhodných realizací. Nejlepší předchůdce pro kandidáta $u_k(i)$ je pak hledán pouze v množině kandidátů kroku $k-1$ splňujících podmínku (schéma viz obrázek 6.4):

$$|F_0(u_{k-1}(j)) - F_0(u_k(i))| \leq {}_{\max}\Delta F_0,\ j = 1, \ldots, N(k-1). \tag{6.2}$$

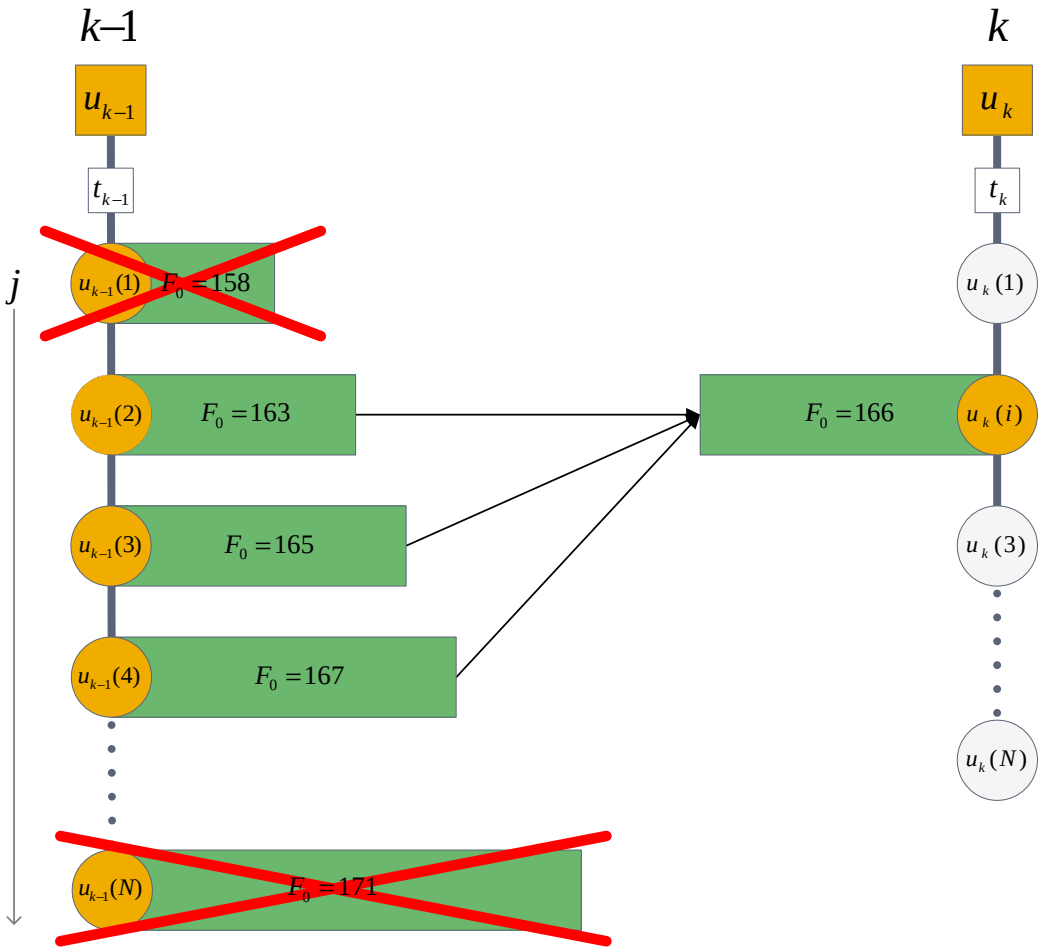
Problémem této optimalizace jsou neznělé hlásky, u kterých nelze frekvenci F_0 vyhodnotit, protože jsou tvořeny šumem. Jednou možností je ponechat hodnotu nulovou a příslušným způsobem upravit hledání nejlepšího předchůdce. Druhý způsob, s nímž autoři prováděli experimenty, je použít hodnotu F_0 získanou interpolací této frekvence nejbližších sousedních znělých hlásek.

6.1.5.2 Alternativní přístup

Kromě výše uvedeného postupu byla v publikaci navržena i alternativa, která nepracuje pouze s průměrnou hodnotou F_0 segmentů, ale s hodnotami F_0 na jejich začátku a na konci. Autoři totiž uvádějí, že v naprosté většině případů naměřili u testovacích vět mezi segmenty z optimální sekvence maximální odchylku do 10Hz, a proto by mohla být množina kandidátů ještě více omezena, neboť by se pracovalo s přesnějšími hodnotami. Tuto metodu zatím autoři nevyzkoušeli, ale je v článku popsána jako cesta, kterou chtějí dále rozvíjet.

6.1.5.3 Zhodnocení optimalizace s využitím F_0

Z publikované práce není zcela zřejmé, jaké zvolili autoři konkrétní hodnoty parametrů pro realizaci experimentů s touto optimalizací. Uvádí se zde testování v několika variantách, které kombinují různé hodnoty omezení ${}_{\max}\Delta F_0$ a způsob vyhodnocení F_0 u neznělých hlásek. Jak je patrné z tabulky 6.4, pohybovala se míra snížení počtu výpočtů cen řetězení v rozsahu 50 – 90 % ve srovnání se základním Viterbiovým algoritmem. Tím se dostáváme na zhruba dvojnásobné urychlení algoritmu výběru jednotek s tím, že u konzer-



Obrázek 6.4: Optimalizace hledání nejlepšího předchůdce s využitím F_0 ($_{\max}\Delta F_0 = 5$Hz, znázorněno pro kandidáta kroku k s indexem $i = 2$).

vativnějších hodnot parametrů nebyla pozorována významná ztráta kvality řeči. Ovšem jak i autoři dále uvádějí, u nejagresivnější verze již docházelo ke zhoršení kvality syntetické řeči a potvrzují to i poslechové testy.

Optimalizace samotná nezaručuje nalezení globálně optimální sekvence řečových segmentů, ale její implementace je poměrně jednoduchá a při vhodném nastavení parametrů by mohla vést k zajímavé míře omezení výpočtů

[3] Údaje nejsou zcela přesné, protože v článku byly tyto výsledky prezentovány pouze formou grafu bez přesných číselných hodnot.

[4] Nastavení parametrů nebylo v článku uvedeno.

Tabulka 6.4: Výsledky měření počtu vyhodnocených kandidátů při použití modifikací Viterbiova algoritmu s využitím znalosti F_0

Varianta optimalizovaného algoritmu		Úspora počtu vyhodnocení cen řetězení[3] [%]
$_{\max}\Delta F_0$	F_0 neznělých hlásek	
střední (konzervativnější)[4]	nulová	45
	interpolovaná	65
malá (agresivní)[4]	nulová	60
	interpolovaná	85

cen řetězení i bez znatelné ztráty kvality.

6.1.6 Omezení realizací řečových jednotek pomocí shlukování

Jeden z často využívaných způsobů urychlení výběru jednotek je založen na technice *shlukování*, kdy se v přípravné fázi uspořádají realizace řečových jednotek do menších skupin (shluků) na základě podobnosti, a algoritmus výběru jednotek pak pro každou jednotku kroku k omezí množinu kandidátů podle vhodně vybraného shluku (Black & Taylor, 1997).

Existuje množství přístupů k řešení problému shlukování objektů, nicméně nejčastěji jsou používané klasifikační a regresní rozhodovací stromy[5] – CART (z angl. *Classification And Regression Tree*). Obecně se u rozhodovacích stromů pracuje se sadou příznaků, z jejichž hodnot lze pomocí stanovené funkce vypočítat vzdálenost mezi dvěma libovolnými objekty. Na počátku konstrukce rozhodovacího stromu jsou všechny objekty součástí jednoho shluku. K objektům ve shluku se nalezne vhodná otázka (podmínka) pracující s jedním vybraným příznakem, která rozdělí objekty do dvou shluků tak, aby byla minimalizována vzdálenost mezi objekty zařazenými do stejného shluku. Tento postup dělení shluků se postupně opakuje, dokud není splněna podmínka na zastavení, jako je například minimální stanovený počet objektů ve shluku nebo limit na minimální rozdíl ve vzdálenosti mezi objekty. Takto vytvořený rozhodovací strom je tedy tvořen kořenovým shlukem, který se

[5] Detailnější popis klasifikačních a rozhodovacích stromů je uveden např. v knize Breiman *et al.*, 1984.

hierarchicky člení na podřízené shluky pomocí otázek vztažených k hodnotám příznaků. Listy stromu reprezentují shluky, v nichž jsou si objekty velmi podobné, tj. leží velmi blízko sebe v prostoru definovaném příznaky.

V případě užití této metody pro omezení realizací řečových jednotek při výběru optimální sekvence jsou realizace řečových jednotek ještě před zahájením procesu shlukování rozděleny podle své třídy, tj. základního typu jednotky používané TTS systémem (difony, trifony, slabiky, apod.), a pro každou třídu je vytvořen zcela samostatný rozhodovací strom.

6.1.6.1 Určení vzdálenosti realizací jednotek

Klíčovým prvkem optimalizace výběru jednotek pomocí shlukování je definice funkce k určení vzdálenosti mezi dvěma realizacemi jednotek. Ve většině publikovaných prací se pracuje s různými kombinacemi příznaků vycházejících z fonetického kontextu, prozodických nebo spektrálních charakteristik.

S ohledem na rychlost vyhledání vhodného shluku se typicky používají příznaky tvořící specifikaci cíle, tedy příznaky, jejichž hodnota není zatížená složitým výpočtem nebo je vypočítaná dopředu ve fázi přípravy TTS systému. Příklady příznaků, které jsou používané k měření vzdálenosti dvou realizací jednotek z řečového korpusu, jsou:

- typ (třída) předchozí nebo následující jednotky,
- základní perioda hlasivkového tónu, resp. základní frekvence F_0,
- MFCC koeficienty,
- délka trvání,
- energie,
- pozice ve slabice, slově, prozodické klauzi nebo frázi.

Hodnoty vybraných příznaků tvoří vektor popisující danou realizaci řečové jednotky a vzdálenost dvou realizací je vypočtena jako vzdálenost mezi těmito vektory. Protože se rozsah hodnot jednotlivých prvků vektorů liší, lze k určení vzdálenosti mezi nimi využít například *Mahalanobisovu vzdálenost* (popsána v Mahalanobis, 1936 a použita např. v Black & Taylor, 1997).

6.1.6.2 Zhodnocení omezení realizací řečových jednotek pomocí shlukování

Metody shlukování umožňují omezit v reálném čase množinu kandidátů v každém kroku k procesu výběru realizací jednotek, a tím dosáhnout jeho urychlení. Podobně jako u jiných metod popsaných v této práci, nezaručuje omezení kandidátů shlukováním nalezení globálně optimálního řetězce. Jedná se o implementačně náročnější přístup, zejména v přípravné fázi vývoje TTS systému, protože je nutné natrénovat rozhodovací strom na testovací množině vět. Jedním z klíčových parametrů této metody je počet realizací řečových jednotek ve shluku, jenž zároveň určuje míru omezení počtu kandidátů v daném kroku k procesu výběru jednotek.

Metody shlukování k omezení kandidátů lze obtížně hodnotit jak z hlediska urychlení, tak i kvality výsledné řeči, protože ve většině publikovaných prací nejsou udané přesné nebo porovnatelné výsledky. Jako příklad výsledků lze uvést článek Ling *et al.*, 2002, kde se uvádí 3 až 4 násobné urychlení proti základnímu algoritmu bez omezení kandidátů. Srovnání kvality generované promluvy bylo řešeno poslechovými testy typu MOS[6)] a bylo zde dosaženo velmi dobrého výsledku 4,18, přičemž výsledek u původní metody bez shlukování dosáhl hodnoty 4,23.

Určitou nevýhodou většiny technik omezení množiny kandidátů s využitím shlukování pomocí specifikace cíle je to, že nijak nezohledňují cenu řetězení, tj. jak dobře bude možné realizace řečových jednotek navázat. V publikaci Nishizawa & Kawai, 2007 je prezentován způsob shlukování, jenž se snaží tuto nevýhodu odstranit tím, že vzdálenost mezi dvěma realizacemi určuje na základě míry zhoršení ceny řetězení jednotek v porovnání s optimální sekvencí. Postup je takový, že se nejprve provede syntéza velkého množství vět bez omezení kandidátů. Po dokončení procesu výběru jednotek jednotlivých vět je pro každého kandidáta mimo optimální sekvenci vypočítáno zhoršení kumulativní ceny řetězení, když se v kroku k dosadí daný kandidát místo optimální realizace. Takto se postupně vypočítá průměrná míra zhoršení kvality pro všechny kandidáty a zahrne se do vztahu pro určení vzdálenosti dvou realizací. Autoři dosáhli touto metodou lepších výsledků při měření průměrné odchylky kumulativní ceny od syntézy bez užití omezení kandidátů,

[6)] MOS je zkratka z angl. *Mean Opinion Score* a označuje se tak způsob hodnocení vět, typicky na stupnici 1 – 5. Detaily viz např. Tihelka & Matoušek, 2004.

a to zejména v případech velmi nízkého počtu ponechaných jednotek ve shlucích. Bohužel se v publikaci pouze uvádí graf rozdílů průměrných hodnot cen vybraných realizací jednotek vůči metodě postavené na shlukování pomocí příznaků cen cíle. Úroveň urychlení procesu výběru jednotek z hlediska času nebo výpočetního výkonu autoři neuvádějí.

Kapitola 7

Způsob hodnocení algoritmů

Výkonnost v této práci uváděných algoritmů výběru optimální sekvence řečových jednotek je nezbytné hodnotit ze dvou úhlů pohledu. Prvním z nich je rychlost, resp. výpočetní náročnost celého procesu ve srovnání se základním Viterbiovým algoritmem. Druhým je pak dopad zrychlení na celkovou kvalitu syntetizované řeči.

7.1 Množina testovacích promluv

Pro potřeby srovnání bylo nejprve nutné připravit testovací text, jehož syntézou budou algoritmy porovnávány. Pro tyto účely byla náhodným výběrem z novinových článků vytvořena množina 20 promluv. Přesněji se jedná o prozodické klauze, tedy nejmenší samostatné části syntetizovaného textu, které začínají a končí pauzou (viz Romportl, 2006). Úplný seznam textů je uveden v příloze A.

7.2 Míra urychlení

V souladu se stanovenými cíli této práce je míra urychlení navržených algoritmů klíčovým ukazatelem výkonnosti algoritmu. Jeho výhodou je, že ho lze hodnotit objektivně pomocí matematického srovnání.

V oddíle 4.1.2.1 popisujícím výpočetní náročnost Viterbiova algoritmu a v něm odkazované publikaci Beutnagel *et al.*, 1999 je uváděno, že výpočtů cen řetězení C^c je takové množství, že zabírají 78 % výpočetního času nutného ke kompletní syntéze požadované promluvy.

Pro systém ARTIC, v němž se používá verze Viterbiova algoritmu značená kódem *VITBASE* (viz oddíl 8.1.1), bylo provedeno detailní měření profilací zdrojového kódu. Kompletní proces výběru optimální sekvence jednotek zabral 99,83 % z celkové doby syntézy řeči z textu. Do tohoto času spadá i doba nezbytná k vypočtení cen cíle C^{t} všech kandidátů, ovšem její podíl byl téměř zanedbatelný. Naopak na výpočty samotných cen řetězení bylo spotřebováno dokonce 89,29 % z času výběru jednotek, což v poměru k celkové době syntézy řeči z textu odpovídá hodnotě 89,14 %.

Tabulka 7.1: Poměr spotřebovaného času vybraných podprocesů syntézy řeči u systému ARTIC vůči celkovému času syntézy řeči z textu

Podproces	Spotřeba času [%]
Zpracování vstupního textu	0,0001
Výběr kandidátů z databáze řečových jednotek	0,0540
Výběr optimální sekvence jednotek	99,8348[1)]
↳ Výpočet cen řetězení C^{c}	89,1413[2)]
↳ Výpočet cen cíle C^{t}	0,2234[2)]
Generování výsledné řeči řetězením vybraných segmentů	0,0003

Ve zbývajících 0,17 % času je prováděno například převedení textu do fonetické podoby, výběr kandidátů jednotek z databáze nebo samotné řetězení segmentů řeči. Jedná se tedy o čas, který již nelze nijak významně zkrátit, navíc by ani snaha v tomto směru neměla výrazný efekt na celkovou rychlost. Zde by připadala v úvahu pouze redukce databáze řečových jednotek, ale tou by došlo pouze k implicitnímu zvýšení rychlosti v důsledku menšího počtu kandidátů a nikoliv o vnitřní optimalizaci prováděných úkonů. Statistika poměru naměřeného času pro vybrané podprocesy syntézy řeči je uvedena v tabulce 7.1.

Na základě uvedené statistiky byla pro účely této práce stanovena míra urychlení (případně zpomalení) jako poměr celkového počtu vyhodnocení cen řetězení C^{c} mezi referenčním Viterbiovým a zkoumaným algoritmem.

[1)] Souhrnný čas výběru optimální sekvence jednotek proti celkovému času syntézy, jenž zahrnuje i výpočet cen řetězení, výpočet cen cíle a další prováděné výpočetní operace.

[2)] Přestože je podproces prováděn jako součást výběru optimální sekvence, je hodnota uváděna v poměru k celkovému času syntézy.

Důvodem zvolení tohoto způsobu hodnocení rychlosti před prostým měřením časuje je také to, že experimenty byly realizovány pomocí skriptovacího jazyka Python, který se řadí mezi interpretované jazyky, takže rychlost výsledného programu nelze srovnávat s produkčním kódem napsaným v jazyce C. Navíc byly všechny experimenty s ohledem na svou výpočetní náročnost spouštěny na gridové infrastruktuře, kde běžely jednotlivé úlohy v rámci výpočetního clusteru na různě výkonných procesorech. Testovací kód také obsahoval množství ladících výpisů, statistik a některé jeho části nebyly ideálně optimalizované, což by naměřený čas také zkreslovalo.

Pro konkrétní syntetickou promluvu z testovací množiny označenou indexem i lze míru urychlení zapsat vztahem:

$$s(i) = \frac{{}_{base}N^c(i)}{N^c(i)}, \tag{7.1}$$

kde $N^c(i)$ je funkce vyjadřující počet vyhodnocení cen řetězení daného algoritmu. Označení ${}_{base}N^c(i)$ pak nese speciální případ této funkce vyhrazený pro referenční hodnotu verze Viterbiova algoritmu VITBASE.

K porovnání rychlosti algoritmů je používána průměrná míra urychlení nad souborem testovacích promluv:

$$S = \frac{1}{n}\sum_{i=1}^{n} s\,(i)\,, \tag{7.2}$$

kde n je počet promluv a $s(i)$ je míra urychlení pro každou z nich. Význam získané hodnoty S je patrný z přehledové tabulky 7.2.

Tabulka 7.2: Význam hodnot míry urychlení algoritmu vůči základnímu Viterbiovu algoritmu

Míra urychlení	Význam
$S > 1$	algoritmus je rychlejší
$S < 1$	algoritmus je pomalejší
$S = 1$	algoritmus je stejně rychlý

Míru urychlení s hodnotou $S > 1$ lze interpretovat také tak, že daný

algoritmus výběru optimální sekvence řečových jednotek je S krát rychlejší než algoritmus VITBASE.

7.3 Kvalita syntetické řeči

Většina prezentovaných algoritmů výběru jednotek ze své podstaty nemusí dospět ke globálně minimální sekvenci ve smyslu kumulativní ceny C a může tak dojít k výběru odlišné sekvence s vyšší hodnotou C než u základního Viterbiova algoritmu. Na druhou stranu však nemusí mírné navýšení kumulativní ceny nutně vést ke zhoršení kvality a přirozenosti výsledné promluvy, protože zejména u rozsáhlé databáze řečových jednotek existuje často i více sekvencí, které jsou při poslechovém testu v podstatě nerozlišitelné. Alternativní suboptimální sekvence může znít dokonce i lépe v případě, kdy se v sekvenci s globálně minimální kumulativní cenou vyskytne jedna nebo více realizací problémových jednotek, které vznikly např. v důsledku chyby řečníka, nepřesnosti při segmentaci řečového korpusu nebo nesouladu frekvence F_0. Takovéto problémové segmenty mohou významně narušit přirozenost promluvy (více viz také oddíl 9). Definice ceny cíle a ceny řetězení je v obecné rovině velmi složitý problém zahrnující výběr mnoha příznaků společně s vhodným nastavení jejich vah a kumulativní cena tedy bohužel nemusí přesně korelovat s lidským vnímáním kvality generované promluvy (i když se o to všechny výzkumné týmy zabývající se syntézou řeči s výběrem jednotek snaží – viz např. práce Hunt & Black, 1996, Bjørkan *et al.*, 2005, Vepa & King, 2003, Vepa *et al.*, 2002, Kirkpatrick *et al.*, 2006, Blouin *et al.*, 2002, Lakkavalli *et al.*, 2010, Zhang *et al.*, 2010 nebo Legát, 2012).

7.3.1 Matematické hodnocení kvality

7.3.1.1 Míra zhoršení kumulativní ceny

Algoritmy výběru jednotek jsou konstruovány s cílem minimalizovat kumulativní cenu C výsledné sekvence řečových jednotek. Jedná se o číselnou hodnotu, kterou lze použít právě k matematickému porovnání kvality. Etalonem pro tento způsob hodnocení je opět referenční Viterbiův algoritmus VITBASE, jenž zaručuje nalezení globálně minimální hodnoty C. Pro hodnocený algorit-

mus bude jako $C(i)$ označována funkce odpovídající kumulativní ceně[3)] vítězné sekvence jednotek pro danou promluvu i z testovací množiny. Hodnocení kvality v matematické rovině je založeno na míře odchylky kumulativní ceny $q(i)$ určené vztahem:

$$q(i) = \frac{C(i) -_{base} C(i)}{{}_{base}C(i)} \cdot 1000\ [‰], \tag{7.3}$$

kde ${}_{base}C(i)$ odpovídá globálně minimální hodnotě získané referenčním Viterbiovým algoritmem a $C(i)$ je kumulativní cena nejlepší cesty vybrané hodnoceným algoritmem.

Podobně jako u míry urychlení, je ke vzájemnému porovnání algoritmů počítána výsledná hodnota Q jako průměr ze všech promluv z testovací množiny:

$$Q = \frac{1}{n} \sum_{i=1}^{n} q\,(i)\,, \tag{7.4}$$

kde n je celkový počet promluv a $q(i)$ odchylka kumulativní ceny pro každou z nich.

Výsledná hodnota Q udává míru odchylky kumulativních cen syntetické řeči v promile[4)] ve vztahu k referenčnímu Viterbiovu algoritmu. Hodnota $Q = 0$ ‰ pak znamená, že vybrané posloupnosti jednotek jsou zcela shodné s referenčním algoritmem a že bylo dosaženo nejvyšší kvality z pohledu matematického měření. Čím je hodnota Q vyšší, tím je rozdíl kumulativních cen větší a zároveň s tím je i kvalita horší.

7.3.1.2 Míra fragmentace promluvy

Konkatenační syntéza řeči generuje výslednou promluvu složením vhodných úseků řeči vybraných z databáze, která byla vytvořena segmentací řečového korpusu namluveného lidským řečníkem. I když se u TTS systémů pracuje s jednotlivými segmenty, nahrává způsob výběru jejich optimální sekvence (viz

[3)] Kumulativní cena vítězné sekvence pro daný algoritmus nemusí být nutně globálně minimální.

[4)] V počátku experimentů bylo srovnání uváděno v procentech, ale vzhledem k velmi malým hodnotám je odchylka kvůli přehlednosti uváděna v promile.

oddíl 4.1.1) tomu, že zcela přirozeně dochází k preferenci kratších či delších řetězců segmentů, jež spolu přímo sousedily v původní nahrávce (navazovaly na sebe). Pokud syntetizovaný text jako celek odpovídá některé z původně nahraných promluv, pak dojde zřetězením příslušných segmentů k rekonstrukci původní zdrojové nahrávky a výsledek syntetizéru zní naprosto přirozeně. Tyto případy však nejsou u obecného TTS systému časté a generovaná promluva se většinou skládá z různě dlouhých úseků (*fragmentů*) z odlišných částí řečového korpusu.

Místa spojení jednotlivých fragmentů budou v dalším textu nazývány *body nespojitosti*, protože koncový segment prvního a počáteční segment druhého fragmentu na sebe nenavazují zcela přirozeně a musí se u nich provést korekce zvukového signálu neboli vyhlazení. Pokud jsou prozodické vlastnosti hraničních segmentů velmi odlišné, nemusí být vyhlazení úspěšné a může dojít ke vzniku nežádoucích artefaktů. Je tak zřejmé, že čím více bodů nespojitosti se v promluvě vyskytuje, tím je výsledná řeč více *fragmentována*[5)] a v důsledku toho může kvalita výstupu klesat[6)]. Míru fragmentace lze objektivně změřit jako poměr počtu potenciálně problematických bodů nespojitosti a celkového počtu všech bodů řetězení vybraných realizací jednotek promluvy. Tento ukazatel bude značen jako *CD* (z angl. *Concatenation Density*[7)]) a k výpočtu jeho hodnoty bude použita funkce $cd(i)$ definovaná vztahem:

$$cd(i) = \frac{N^d(i)}{N^{\mathrm{c}}(i)} \cdot 100 \ [\%], \tag{7.5}$$

kde i je index promluvy v testovací množině, $N^d(i)$ je počet bodů nespojitosti a $N^u(i)$ celkový počet všech bodů řetězení v promluvě. Konečná podoba výpočtu míry fragmentace *CD* je opět určena průměrnou hodnotou nad souborem všech testovacích promluv:

[5)] Pojem *fragmentace* byl zvolen jako analogie ke stejnému pojmu používaného v souvislosti s uložením dat na paměťovém médiu, např. pevném disku.

[6)] Jedná se o teoretickou kvalitu v matematickém měřítku. Bod nespojitosti nemusí vždy být problematický v percepční rovině.

[7)] Pojem *Concatenation Density* byl pro tento způsob hodnocení kvality syntetické řeči zaveden v publikaci Hanzlíček *et al.*, 2013.

$$CD = \frac{1}{n} \sum_{i=1}^{n} cd\,(i)\,. \tag{7.6}$$

Pro takto definované matematické hodnocení platí, že nejlepší možnou kvalitu mají promluvy s nulovou hodnotou *CD*, které lze dosáhnout pouze tehdy, pokud je celá promluva zároveň součástí řečového korpusu a je tak složena pouze ze segmentů, jejichž spojení není nutné nijak vyhlazovat. S rostoucí hodnotou *CD* se kvalita generované řeči z pohledu míry fragmentace zhoršuje až po krajní případ $CD = 100$ %, který by znamenal, že žádné dva za sebou jdoucí segmenty nebyly sousedy v původní nahrávce řečníka.

7.3.2 Hodnocení poslechovými testy

V případě matematického hlediska porovnání algoritmů je kvalita postavena na hodnocení sekvence kandidátů funkcemi pro výpočet cen cíle C^{t} a cen řetězení C^{c} definovaných v oddíle 4.1.2 popisujícím Viterbiův algoritmus. Pokud by byly funkce ideální, pak by globálně minimální cesta grafem kandidátů řečových jednotek dávala vždy objektivně nejkvalitnější promluvu, a to matematicky i percepčně. Bohužel tomu tak není a i když jsou obě kriteriální funkce zkonstruované na základě dlouhodobého výzkumu a řady experimentů, k ideálu se pouze přibližují. Z tohoto důvodu se při hodnocení kvality syntetické řeči provádějí i poslechové testy, jejichž výsledky lze považovat za nejvýznamnější a nejvíce vypovídající ukazatel.

Existuje řada typů poslechový testů, které jsou zaměřeny na různé aspekty hodnocení syntetické řeči a které jsou detailně popsány např. v ITU-T Recomendation P.800, 1996 nebo v publikaci se zaměřením na češtinu Tihelka & Matoušek, 2004. V rámci této práce je vždy porovnávána kvalita dvou různých algoritmů výběru jednotek, a proto je zde používán test typu *CCR*[8)] (angl. *Comparison Category Rating*). Test je konstruován tak, že se nejprve provede syntéza všech promluv z testovací množiny pomocí obou algoritmů, takže ke každé z nich vzniknou dvě verze (A a B). Promluvy jsou pak postupně předkládány posluchači, a ten posuzuje, která z verzí zní přirozeněji, a to na

[8)] Tento typ poslechových testů bývá také označován jako ABX test, případně preferenční test.

stupnici se třemi hodnotami uvedenými v tabulce 7.3. Pro větší objektivitu je pořadí verzí v příslušných dvojicích náhodné.

Tabulka 7.3: Význam hodnot při posouzení kvality řeči v poslechových testech

Hodnocení	Význam
−1	promluva A zní lépe
0	obě verze zní stejně dobře (nebo špatně)
1	promluva B zní lépe

V rámci práce bylo prováděno množství experimentů pro několik algoritmů, resp. jejich variant s různým nastavením parametrů a testy byly zaměřeny na zhodnocení, zda dává jedna verze algoritmu lepší výsledky než druhá. Navíc jde většinou o srovnání algoritmu se základním Viterbiovým algoritmem a zjištěním, jestli se kvalita nezhoršila. Proto, i když se u CCR testů někdy používá k hodnocení pěti nebo dokonce až sedmi bodová stupnice, byly zde zvoleny pouze tři hodnoty, neboť více nutí posluchače zaměřit se na výběr přirozenější promluvy spíše než na určení velikosti kvalitativního rozdílu, jenž je v tomto kontextu méně podstatný.

K zajištění objektivity se u klasického poslechového testu pracuje s hodnocením většího počtu posluchačů a závěry jsou určeny statistickými metodami. Vzhledem k počtu variant algoritmů, které bylo nutno porovnat, nebylo možné rozsáhlé poslechové testy vždy provést. Například při volbě nejlepší kombinace parametrů jednotlivých algoritmů byly používány jednoduché poslechové testy prováděné jednou osobou. I takový test má však svou váhu, zejména pokud slouží jako doplněk předvýběru pomocí matematického srovnání. Pokud se totiž kumulativní cena C pro vybrané posloupnosti jednotek lišila minimálně, pak byl při poslechu hodnocen zejména výskyt artefaktů, a tomu vyhoví i poslechový test provedený jedním zkušeným posluchačem.

Kapitola 8

Algoritmy a experimenty

V kapitole 6 bylo popsáno množství technik zaměřených na urychlení Viterbiova algoritmu a na jejich základě bylo v rámci této práce nejprve vytvořeno několik odvozených verzí. Dále pak byla navržena řada dalších algoritmů výběru jednotek, jež jsou sice postaveny na jiném základu, ale v některých momentech jsou kombinovány s původním algoritmem nebo jeho modifikacemi. Některé optimalizace z předchozí kapitoly byly použity i v jiném než uváděném kontextu. U mnoha algoritmů se například mohou výpočty cen řetězení pro dvě jednotky opakovat, a proto je v nich využívána mezipaměť pro uchování již vypočtených hodnot, avšak platná pouze po dobu syntézy dané promluvy.

Náplní následující části je popis všech testovaných algoritmů, včetně postupu jejich návrhu a porovnání s ostatními algoritmy. Z pohledu této práce je pak stěžejní část 8.2, v níž jsou popsány algoritmy výběru jednotek založené na původní myšlence využití tzv. ZCC řetězců.

8.1 Viterbiův algoritmus a jeho modifikace

Viterbiův algoritmus výběru jednotek byl již v této práci představen v oddíle 4.1.2, stejně tak i detailní popis dosud publikovaných optimalizací v části 6.1. V následujícím textu jsou varianty Viterbiova algoritmu uvedeny z důvodu posouzení jejich výkonnosti a kvalitativního hodnocení.

8.1.1 Základní Viterbiův algoritmus a jeho referenční verze

Základní Viterbiův algoritmus (dále bude označován zkratkou *VITORIG*) představuje standardní metodu výběru optimální posloupnosti řečových jednotek, která se běžně používá v TTS systémech založených na principu konkatenační syntézy. Na rozdíl od ostatních metod byl detailnímu popisu algoritmu již věnován oddíl 4.1.2, a proto nebude postup hledání v tomto místě práce znovu uváděn.

Základní Viterbiův algoritmus VITORIG je zástupcem skupiny technik založených na hledání cesty grafem kandidátů do šířky. Díky tomu, že jsou vyhodnoceny všechny možné cesty, je zaručeno nalezení sekvence s globálně minimální kumulativní cenou (tj. v matematickém měřítku kvality je nalezená cesta zároveň i nejlepší), ovšem za cenu extrémních výpočetních nároků.

V původní verzi systému ARTIC byl výběr jednotek řešen modifikovanou verzí algoritmu VITORIG s jednoduchou optimalizací při vyhodnocování nejlepšího předchůdce každého kandidáta. V této vnitřní smyčce je vždy lokálně uchovávána hodnota dosud nejnižší nalezené kumulativní ceny, a pokud ji kumulativní cena předchůdce přesahuje již před výpočtem ceny řetězení, pak se zpracování tohoto kandidáta vynechá, protože se již nemůže stát nejlepším předchůdcem. Díky této úpravě se podařilo v systému ARTIC snížit množství výpočtů cen řetězení přibližně 7×, přičemž zůstala zachována záruka nalezení cesty s globálně minimální cenou. Všechny ostatní algoritmy jsou v menší či větší míře založeny na heuristických postupech, které většinou znamenají ztrátu této záruky, a proto byl původní algoritmus implementovaný v systému ARTIC použit jako referenční a byl mu přidělen kód VITBASE. Výkon ostatních algoritmů je v této práci vztažen právě k tomuto algoritmu, jehož výkonnost je pro úplnost uvedena v tabulce 8.1.

Tabulka 8.1: Souhrnné hodnocení referenční verze Viterbiova algoritmu (VITBASE)

Vlastnost	Hodnota
Kód algoritmu	VITBASE
Míra urychlení (S)	1,00
Matematická míra zhoršení kvality (Q)	0,00 ‰
Míra fragmentace (CD)	29,39 %

Pokud by se základní Viterbiův algoritmus VITORIG hodnotil stejným

způsobem jako ostatní algoritmy v této práci, pak by matematická kvalita generovaných promluv byla shodná s referenčním algoritmem VITBASE ($Q = 0{,}00$ ‰), ale rychlost by byla nižší (experimentálně změřená míra urychlení činí $S = 0{,}13$).

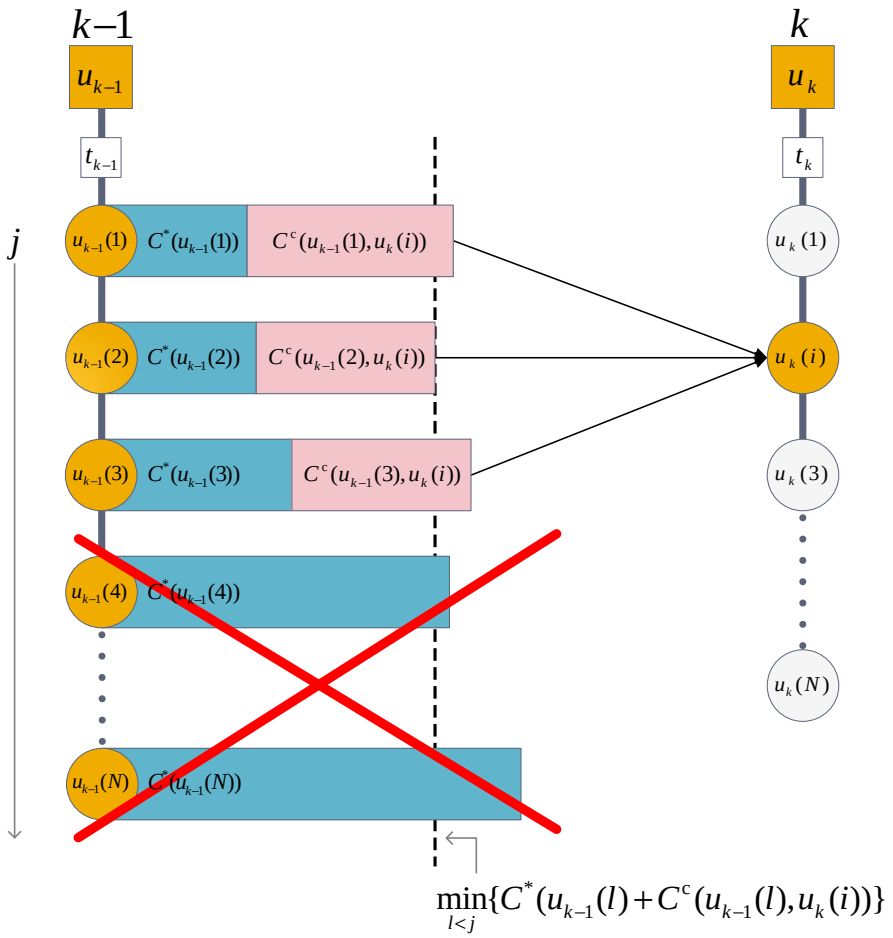
8.1.2 Optimalizovaný Viterbiův algoritmus

První experimenty s urychlením výběru jednotek vycházejí zejména z publikace Sakai *et al.*, 2008, v níž byly uvedeny celkem tři modifikace algoritmu VITBASE. Jedna z těchto modifikací, popsaná v oddíle 6.1.1.2, nenarušuje záruku nalezení globálního minima. Ve zmíněné práci je tato úprava pouze kombinována s dalšími úpravami a není hodnocena samostatně. Proto byla použita k vytvoření prvního experimentálního algoritmu nazvaného optimalizovaný Viterbiův algoritmus (dále *VITOPT*). Jedná se o techniku prořezávání grafu při hledání nejlepšího předchůdce daného kandidáta, kdy je možné při vhodném řazení kandidátů tento proces přerušit, aniž by mohlo dojít ke zhoršení kumulativní ceny vítězné posloupnosti jednotek.

V původním článku pracovali autoři s předem vypočítanými hodnotami minimální ceny řetězení C^c mezi všemi možnými kandidáty dvou sousedních jednotek (difonů). Z podstaty této úpravy platí, že čím je minimální hodnota vyšší, tím je větší pravděpodobnost, že se hledání nejlepšího předchůdce dříve přeruší. Nejnižší možnou hodnotou je nulová cena řetězení (odpovídá případům, kdy se daná kombinace jednotek nachází přímo v původním řečovém korpusu). Nulová hodnota se také použije tehdy, když není minimální cena známá.

Výpočet minimálních cen řetězení mezi všemi typy jednotek zahrnuje obrovské množství výpočtů a ne zcela triviální způsob uložení získaných hodnot a zajištění jejich rychlého vyhledání v době syntézy. Nakonec bylo od vypočtení minimálních cen řetězení upuštěno a byla použita nulová hodnota pro všechny kombinace. K tomuto kroku vedla úvaha nad možným množstvím hodnot, které by byly nenulové. U systémů s velmi rozsáhlou databází řečových jednotek, mezi něž se řadí i systém ARTIC, je velmi malý počet kombinací typů jednotek, které by nebyly obsaženy v nahrávkách řečového korpusu. Tuto hypotézu potvrzuje i analýza uvedená později v textu v oddíle 8.2.2, při níž bylo zjištěno syntézou 10 000 náhodně vybraných promluv, že pouze 4,55 % všech použitých realizací řečových jednotek odpovídalo

samostatným jednotkám, tj. nebyly součástí souvislého řetězce z řečového korpusu. To znamená, že v databázi řečových segmentů s opravdu vysokým počtem realizací by bylo možné nalézt pouze přibližně 5 % nenulových hodnot minimální ceny řetězení ve smyslu popisované optimalizace. Vyhodnocení a uchování hodnot minimální ceny řetězení by patrně mohlo mít výraznější vliv pouze tehdy, pokud by základem systému byl velmi malý řečový korpus nebo v případě systému ARTIC by se provedla výrazná redukce původně obsáhlé databáze řečových segmentů. Upravená verze původního schématu ze strany 32 odpovídající algoritmu VITOPT je pak znázorněna na obrázku 8.1.



Obrázek 8.1: Schéma prořezávání při vyhledávání nejlepšího předchůdce kandidáta bez znalosti minimální ceny mezi všemi kandidáty sousedních jednotek (znázorněno pro kandidáta kroku k s indexem $i = 2$).

Přestože byla za minimální cenu řetězení mezi všemi typy jednotek použita nulová konstanta, bylo dosaženo urychlení $S = 1{,}5185$ (detailní výsledky viz

tabulka 8.2). Zároveň bylo experimentem i ověřeno, že výsledné posloupnosti jsou naprosto shodné s výsledky získanými algoritmem VITBASE, takže kvalita je také totožná a není potřeba ji prověřovat pomocí poslechového testu.

Algoritmus byl dále ještě doplněn o jednu úpravu, která nijak nemůže ovlivnit výslednou sekvenci, a proto je používána i u všech ostatních algoritmů. Modifikace využívá faktu, že cena řetězení je nulová pro dva řečové segmenty, které na sebe navazují v původním řečovém korpusu. Mechanismus výpočtu cen řetězení byl upraven tak, že pro tyto případy výpočet ceny řetězení vynechá a automaticky vrací nulovou hodnotu. Touto optimalizací bylo uspořeno dalších přibližně 0,0074 % všech výpočtů cen řetězení, takže konečná míra urychlení se pro algoritmus VITOPT se sice velmi nepatrně, ale přeci jen zvýšila na $S = 1{,}5187$. Vzhledem k tomu, jak je změna triviální, však nebyl důvod ji do systému nezavést.

Tabulka 8.2: Souhrnné hodnocení optimalizovaného Viterbiova algoritmu (VITOPT)

Vlastnost	Hodnota
Kód algoritmu	VITOPT
Míra urychlení (S)	1,52
Matematická míra zhoršení kvality (Q)	0,00 ‰
Míra fragmentace (CD)	29,39 %

POZNÁMKA KE SROVNÁNÍ S ALGORITMEM VITBASE

Protože se v konečné podobě algoritmu VITOPT používá nulová minimální cena řetězení mezi dvěma jednotkami, liší se od referenčního algoritmu VITBASE pouze tím, že jsou kandidáti předchozího kroku seřazeni[1)] podle kumulativní ceny C^*. Díky tomu dochází k efektivnějšímu prořezávání kandidátů při vyhodnocení nejlepšího předchůdce, neboť vnitřní cyklus lze přerušit okamžitě po nalezení prvního kandidáta s vyšší kumulativní cenou než dosud nejlepší nalezenou. U algoritmu VITBASE se řazení nepoužívá, takže jsou kandidáti zpracováváni v náhodném pořadí a míra snížení množství výpočtů cen řetězení C^{c} tak závisí na tom, jak brzy je nalezen nejlepší předchůdce nebo jiný kandidát, který má velmi nízké ohodnocení blížící se lokálnímu minimu.

[1)] Řazení kandidátů se provádí jednorázově před zahájením hledání a navýšení výpočetní náročnosti je tak vůči celkovému času výběru jednotek zanedbatelné.

8.1.3 Optimalizovaný Viterbiův algoritmus s prořezáváním typu BEAM

Další z vytvořených algoritmů označený jako *VITBEAM* přímo navazuje na předcházející algoritmus VITOPT a rozšiřuje jej o další dvě modifikace původně popsané v oddílech 6.1.1.3 a 6.1.1.4. Jak již bylo uvedeno, není zde využívána znalost minimální ceny řetězení mezi dvěma jednotkami, resp. se v kontextu původních optimalizací používá nulová hodnota.

Algoritmus zavedením první modifikace mění logiku prohledávání z úplného hledání do šířky na hledání typu *BEAM*. Po vyhodnocení všech kandidátů daného kroku dochází k jejich prořezání a dále se pracuje pouze s realizacemi jednotek s nejnižší kumulativní cenou v množství omezeném limitem BW.

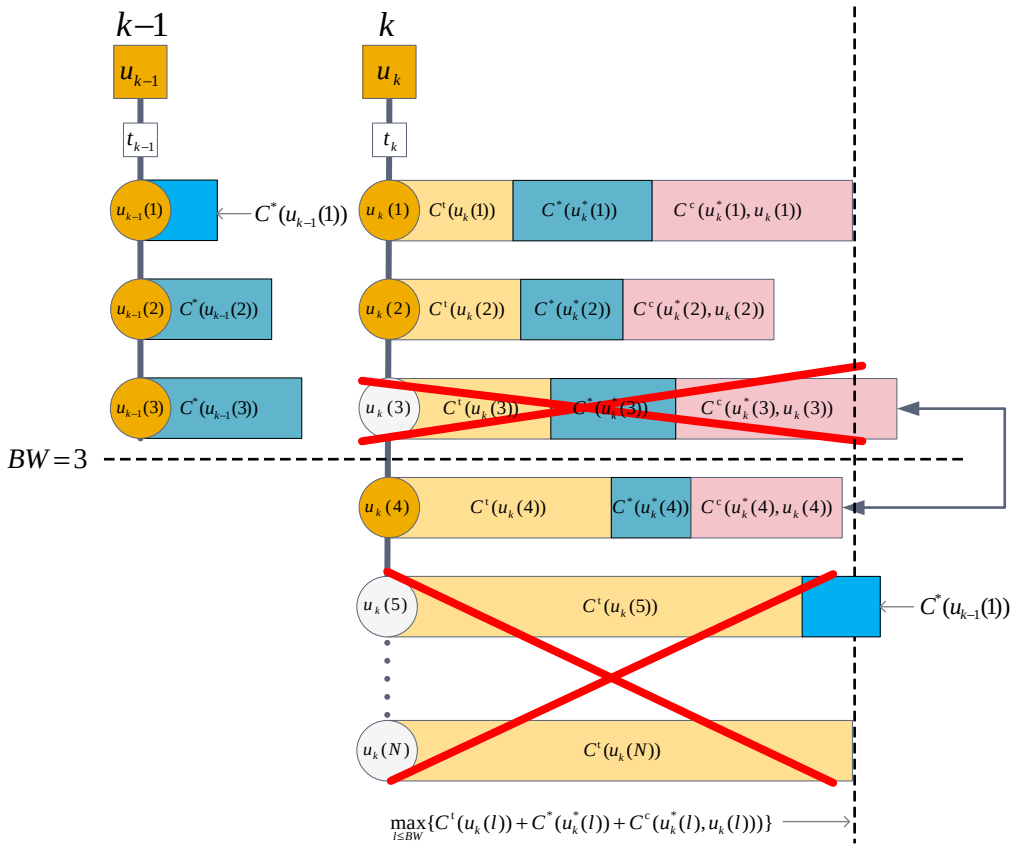
Druhá z úprav je optimalizací, která bez dalšího zhoršení kumulativní ceny prořezává kandidáty, kteří se již nemohou zařadit do množiny BW nejlepších realizací, a to bez nutnosti vypočítávat ceny řetězení C^{c} s jejich předchůdci. Předpokladem je to, že kandidáti předchozího kroku $k-1$ jsou seřazeni podle hodnoty kumulativní ceny C^{*} a také že kandidáti aktuálního kroku k jsou seřazeni podle ceny cíle C^{t}. U kandidátů kroku k se postupně provádí vyhledání nejlepšího předchůdce z kroku $k-1$. Pokud platí, že je již nalezeno BW realizací s nejnižší kumulativní cenou C^{*}, pak lze hledání přerušit, jestliže součet ceny cíle C^{t} některého z dalších kandidátů a kumulativní ceny nejlepší realizace předchozího kroku $k-1$ je vyšší než nejhorší kumulativní cena realizací v limitu BW.

Pozměněné schéma obou uvedených modifikací se zohledněním nedostupnosti minimální ceny řetězení mezi odpovídajícími jednotkami je ilustrováno na obrázku 8.2.

Tímto zásahem však dochází k významné změně, kdy již není zaručeno nalezení cesty grafem kandidátů s globálně minimální kumulativní cenou. Zároveň jde o první algoritmus, jehož vlastnosti závisí na nastavení proměnlivých parametrů. Nebylo tak možné změřit jednu konkrétní hodnotu míry urychlení ani matematické kvality, ale bylo nutné provést množství experimentů s různými kombinacemi parametrů (viz tabulka B.1 v příloze B) a nakonec vybrat optimální nastavení algoritmu.

HODNOCENÍ VÝKONU

Graf na obrázku 8.3 znázorňuje vztah mezi dosaženou rychlostí S a odpovídající matematickou kvalitou Q pro algoritmus VITBEAM. Zároveň je zde

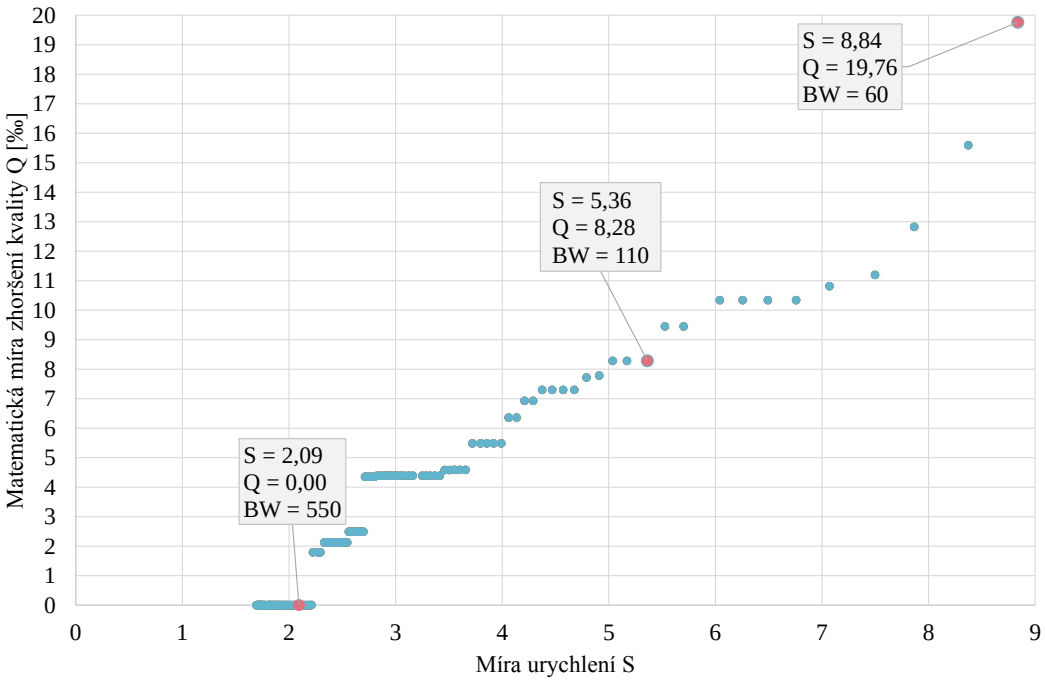


Obrázek 8.2: Optimalizace při vyhodnocování kumulativních cen kandidátů bez znalosti minimální ceny mezi všemi kandidáty sousedních jednotek.

vynesena i hodnota jediného parametru algoritmu BW. Zcela dle očekávání roste rychlost výběru jednotek se snižující se hodnotou limitu BW. Zrychlení u Algoritmu VITBEAM je založeno na prořezávání kandidátů jednotek po zpracování každého kroku, a tím dochází logicky i ke zhoršení matematické kvality. V případě již velmi agresivního nastavení prořezávání $BW = 61$ bylo sice dosaženo vyšší míry urychlení $S = 8{,}84$, ale již za cenu zhoršení matematické kvality $Q = 19{,}76$, přičemž toto zhoršení bylo u některých promluv při jejich poslechu patrné.

Konfigurace algoritmu, která ještě bezpečně zajišťuje výběr jednotek s globálně minimální kumulativní cenou (tj. $Q = 0$), pak odpovídala nastavení limitu $BW = 550$. Míra zrychlení byla ovšem v tomto případě poměrně nízká a rovnala se hodnotě $S = 2{,}09$ (viz tabulka 8.3).

Výkonnost algoritmu VITBEAM lze hodnotit tak, že lze sice dosáhnout



Obrázek 8.3: Matematická kvalita ve vztahu ke zrychlení pro různé parametry algoritmu VITBEAM.

při nízkých hodnotách parametru BW vyšších rychlostí, ale za cenu zhoršení kvality. Bezpečné nastavení algoritmu sice zajistí nalezení sekvencí shodných s výstupem základního Viterbiova algoritmu, ale míra urychlení je velmi nízká a výrazně zaostává za očekáváním této práce.

Tabulka 8.3: Souhrnné hodnocení optimalizovaného Viterbiova algoritmu prořezáváním typu BEAM (VITBEAM) pro bezpečné nastavení parametrů

Vlastnost	Hodnota
Kód algoritmu	VITBEAM
Míra urychlení (S)	2,09
Matematická míra zhoršení kvality (Q)	0,00 ‰
Míra fragmentace (CD)	29,39 %
Parametry: $BW = 550$	

8.1.4 Optimalizovaný Viterbiův algoritmus s prořezáváním kandidátů po vyhodnocení ceny cíle

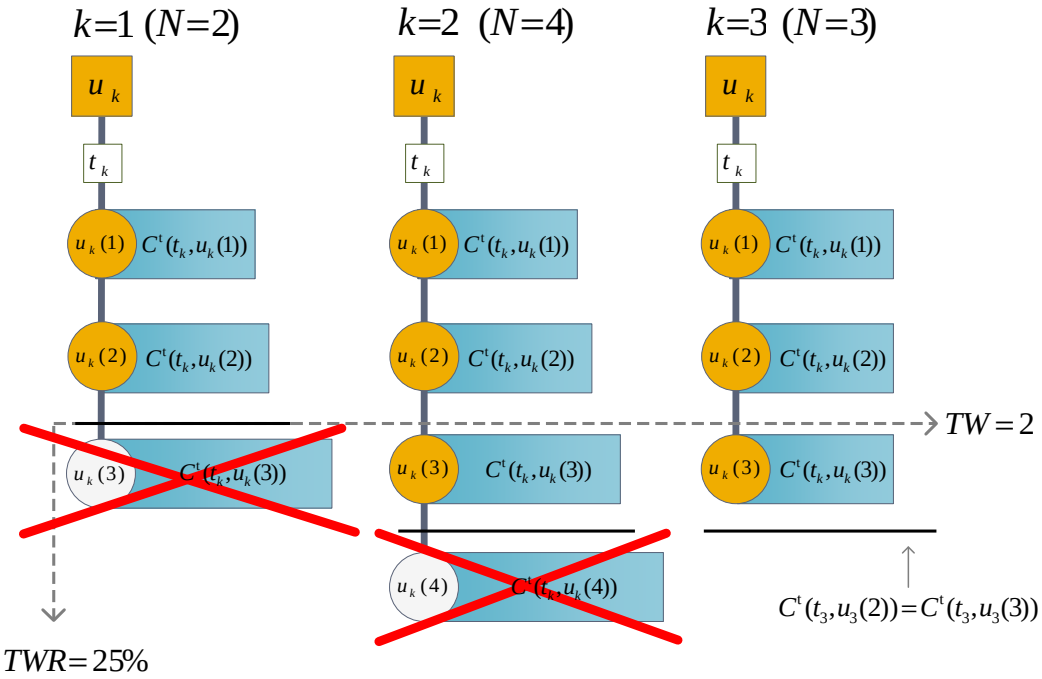
Záměrem této práce je snížení výpočetních nároků u TTS systémů, zejména v případech, kdy je využívána rozsáhlá databáze řečových segmentů. Množství kandidátů k požadované jednotce je v takovém případě vysoké a jednou z možností, jak jejich počet snížit, je provést předvýběr na základě některého kritéria, jehož vyhodnocení nebude zatěžovat algoritmus dalšími výpočty (nebo pouze minimálně). Tato myšlenka z určitého úhlu pohledu odpovídá technice použité v publikaci Conkie *et al.*, 2000, kde se využívala pomocná hodnoticí funkce, která byla založena na typu okolních jednotek v původním řečovém korpusu a která určovala odchylku kontextu dané jednotky při jejím použití v syntetizované promluvě (blíže viz oddíl 6.1.3).

U systémů s výběrem jednotek již samotná funkce pro výpočet ceny cíle C^{t} zprostředkovaně zahrnuje kontext umístění realizace jednotky v požadované promluvě. S ohledem na to a na skutečnost, že ohodnocení všech kandidátů cenou cíle zabírá statisticky velmi malou část výpočetního času celé syntézy (u systému ARTIC činí 0,2234 %), nabízí se možnost spočítat cenu cíle u všech kandidátů a pracovat pak pouze s omezeným počtem realizací s nejnižší hodnotou.

Algoritmus VITBEAM byl tedy doplněn o předvýběr po vyhodnocení cen cíle C^{t} a vznikl tak nový algoritmus označený *VITPRUNE*, u něhož se graf kandidátů omezí pouze na realizace jednotek s nejnižší hodnotou ještě před započetím jeho prohledávání. Vzhledem k tomu, že počet dostupných kandidátů pro různé jednotky je velmi proměnlivý, byla zvolena dynamická hranice, která vychází z počtu kandidátů množiny daného kroku. Hranice prořezání je určena parametrem TW (z angl. *Target Cost Width*), který je dále doplněn o rezervu TWR specifikovanou v procentech. Pokud je počet kandidátů $N(k)$ kroku k menší než hodnota TW, pak se množina kandidátů neprořezává. Když je však množina větší, jsou kandidáti seřazeni dle ceny cíle C^{t} a v grafu se ponechá W nejlepších kandidátů, kde $W = TW + N(k) \cdot TWR\,\%$. Schéma prořezávání je znázorněno na obrázku 8.4.

HODNOCENÍ VÝKONU

Algoritmus VITPRUNE v sobě spojuje všechny dosud uváděné a testované optimalizace základního Viterbiova algoritmu. Z algoritmu VITOPT byly



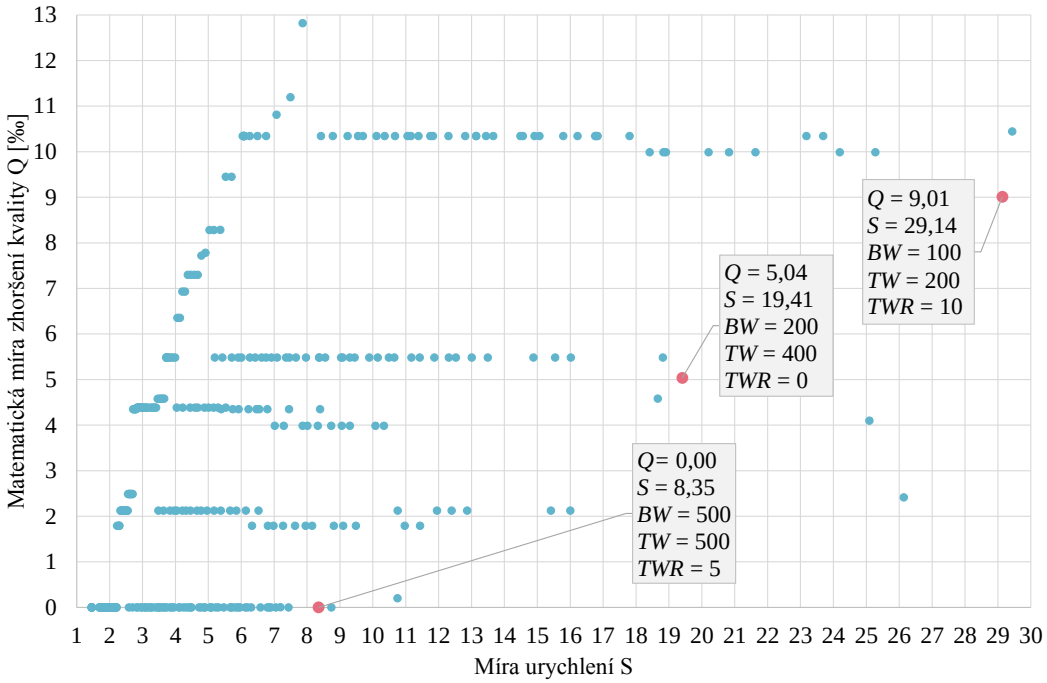
Obrázek 8.4: Schéma prořezávání kandidátů dle ceny cíle C^{t}. V kroku $k = 3$ má realizace $u_3(3)$ shodnou cenu cíle jako u hraniční realizace $u_3(2)$, a proto byla hranice posunuta (viz vysvětlení níže – oddíl 8.1.5 strana 69).

převzaty optimalizace, které sice nejsou nijak významné ($S = 1{,}52$), ale na druhou stranu nijak nemohou zhoršit matematickou kvalitu. Parametr BW byl přejat z algoritmu VITBEAM a určuje se jím míra prořezávání po nalezení nejlepších předchůdců a vyhodnocení kumulativních cen C^* u všech kandidátů každého kroku. Poslední dva parametry TW a TWR umožňují provést omezení uzlů z celého grafu již po vypočtení cen cíle C^{t}.

Jak již bylo popsáno v předchozím textu, byly experimenty prováděny nad databází řečových segmentů systému ARTIC, která obsahovala 656 406 realizací jednotek. Díky tomu je množství kandidátů pro výběr jednotek syntetizovaných promluv opravdu vysoké (až jednotky tisíc) a tím se otevírá prostor k provedení optimalizací prořezáváním.

V této fázi experimentů bylo cílem nalézt takovou kombinaci hodnot parametrů algoritmu, která povede na co nejvyšší míru urychlení S, ale u níž bude ještě zaručeno nalezení cesty s globálně minimální cenou. K tomuto účelu byla provedena syntéza množiny testovacích promluv s různými kombinacemi

hodnot[2)] v rozsahu uvedeném v tabulce B.2 v příloze B.



Obrázek 8.5: Matematická míra zhoršení kvality ve vztahu ke zrychlení pro různé parametry algoritmu VITPRUNE se zaměřením na nízké hodnoty Q. Maximální hodnoty ukazatelů S a Q byly omezeny s ohledem na přehledné znázornění výsledků experimentů dávajících nulovou míru zhoršení matematické kvality ($Q = 0{,}00$ ‰).

Po provedení experimentu byly všechny výstupy porovnány s algoritmem VITBASE a výsledky byly zpracovány do grafu na obrázku 8.5, jenž znázorňuje vztah mezi hodnotou urychlení S a mírou zhoršení kvality Q pro každou testovanou konfiguraci parametrů. Z grafu je patrné, že při zachování kvality shodné s globálně minimální cenou vítězných cest ($Q = 0$), je dosahováno urychlení v řádu jednotek. Krajní hodnotou je v tomto případě zrychlení $S = 8{,}74$, avšak ke stanovení vhodné bezpečné hodnoty bylo zvoleno nastavení parametrů se nižším ziskem zrychlení $S = 8{,}35$.

Algoritmus VITPRUNE lze tedy hodnotit tak, že umožnil navýšit rychlost hledání přibližně 5× ve srovnání s algoritmem VITOPT ($S = 1{,}52$). Sice

[2)]Speciálním případem je nastavení parametru $TW = \infty$, při němž chování algoritmu odpovídá verzi VITBEAM. Pokud se navíc použije hodnota $BW = \infty$, pak se způsob hledání mění na algoritmus VITOPT.

jím není zaručeno, že bude vždy dosaženo globálního minima, ale vhodným nastavením parametrů bylo riziko zhoršení kvality minimalizováno.

Tabulka 8.4: Souhrnné hodnocení optimalizovaného Viterbiova algoritmu VITPRUNE pro bezpečné nastavení parametrů

Vlastnost	Hodnota
Kód algoritmu	VITPRUNE
Míra urychlení (S)	8,35
Matematická míra zhoršení kvality (Q)	0,00 ‰
Míra fragmentace (CD)	29,39 %
Hodnoty parametrů: BW= 500, TW= 500, TWR= 5	

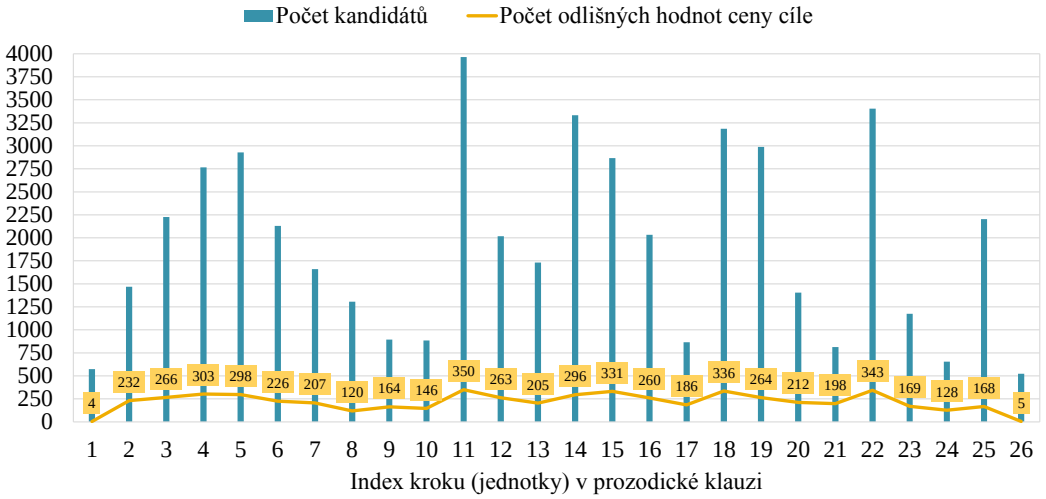
8.1.5 Poznámky ke strategii Viterbiova algoritmu a jeho modifikacím

V předchozím textu byl představen algoritmus VITBEAM, jenž umožňuje zredukovat počet nutných vyhodnocení cen řetězení C^{c} pomocí prořezávání kandidátů na základě průběžné kumulativní ceny. Jeho rozšířená verze VITPRUNE pak přidává prořezávání prohledávaného grafu, kdy se v něm ponechává pro každou jednotku pouze specifikované množství realizací s nejnižší cenou cíle C^{t}. Oba tyto algoritmy lze svou povahou z určitého úhlu pohledu označit za techniky hledání nejlepší cesty grafem do hloubky. Důvodem je fakt, že na základě heuristické úvahy preferují jednotky s průběžně nízkou kumulativní cenou C nebo v druhém případě dokonce již dle velikosti ceny cíle C^{t}. Se strategií prohledání do hloubky pak sdílí sklon ke zvýhodnění lokálních minim, pokud se v grafu kandidátů vyskytují.

ROZLOŽENÍ HODNOT CENY CÍLE U KANDIDÁTŮ

Jedna z dalších slabin algoritmů VITBEAM a VITPRUNE je způsobena typickým rozložením počtu kandidátů u jednotek syntetizovaných promluv. U obecného TTS systému se totiž většinou provádí syntéza vstupního textu po klauzích, tj. větných úsecích ohraničených pauzami. Analýzou provedenou na množině testovacích promluv byl potvrzen předpoklad, že počet realizací

jednotek na začátku a na konci promluv[3)] je nižší než u jednotek uvnitř promluv (viz graf na obrázku 8.6). Ještě větší rozdíl lze vysledovat u statistiky struktury hodnot cen cíle C^{t}. Příčinou je to, že krajní jednotky (začínající pauzou) se vyskytují v původní nahrávce řečníka i v syntetizovaných větách ve stejném kontextu, a proto bývají kandidáti v těchto pozicích ohodnoceni malými a také málo odlišnými cenami.



Obrázek 8.6: Počty kandidátů jednotek doplněné o statistiku počtu odlišných hodnot.

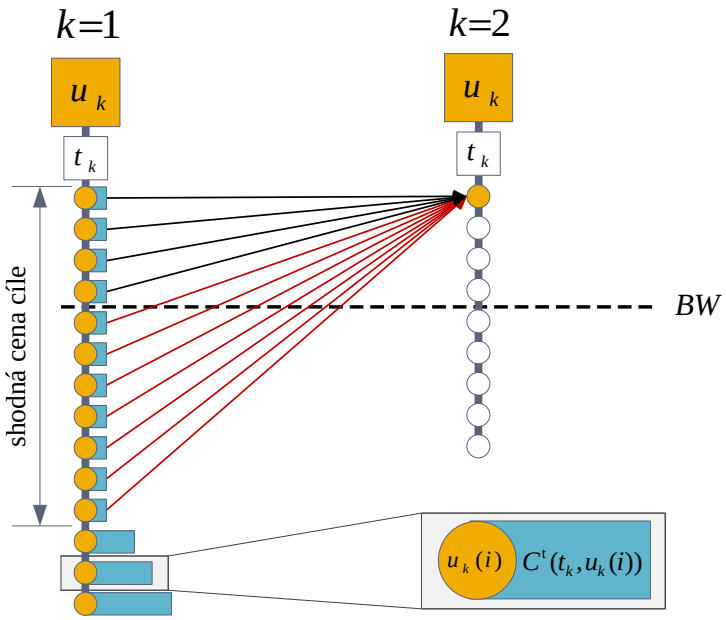
Tento fakt ve svém důsledku zhoršuje matematickou míru zhoršení kvality Q u algoritmů VITBEAM a VITPRUNE, neboť při prořezávání kandidátů na začátku nebo na konci dochází k redukci[4)] kandidátů se stejnými hodnotami. Lze tak říci, že se jedná o náhodný výběr, protože chybí znalost toho, jak se realizace jednotek prvního a druhého kroku budou vhodně řetězit.

Z tohoto důvodu bylo do algoritmů VITBEAM i VITPRUNE zavedeno opatření, kdy se u všech typů prořezávání ponechávají v redukované množině i všichni další kandidáti, jejichž cena je rovna hodnotě zkoumané ceny na stanovené hranici BW nebo $TW + TWR\%$ (viz obrázek 8.4 krok $k = 3$). Tím byl odstraněn problém náhodného výběru, avšak na druhou stranu je

[3)] U systému ARTIC se jedná o difony jejichž úvodní, resp. koncová polovina je tvořena pauzou. Při použití jiných typů jednotek (např. trifony, hlásky) bude sice jejich struktura odlišná, ale i tak lze pro krajní jednotky předpokládat obdobné statistické hodnoty.

[4)] V případě algoritmů VITBEAM aVITPRUNE se tento efekt projevuje zejména na začátku, protože u kandidátů prvního kroku odpovídá kumulativní cena C^{*} ceně cíle C^{t}.

nutné vypočítat více cen řetězení C^{c}. Toto zpomalení se nejvíce projeví právě na začátku a na konci prohledávaného grafu, a to kvůli nepříznivému rozložení hodnot cen cíle C^{t} (schéma problému je znázorněno na obrázku 8.7).

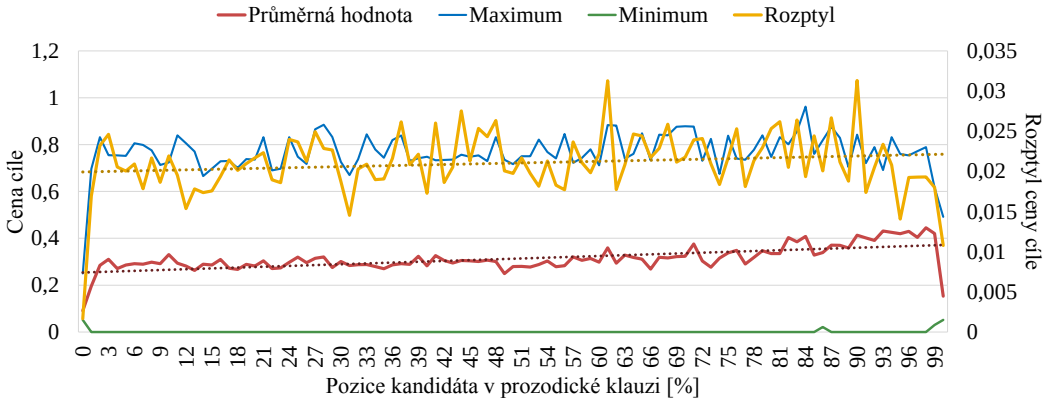


Obrázek 8.7: Schematické znázornění problému s nízkým rozptylem cen cíle u kandidátů na první pozici. Červené šipky představují spojení, jejichž cenu řetězení je nutné vyhodnotit, aby nedošlo k náhodnému výběru kandidátů z kroku $k = 1$ v důsledku prořezání pevnou hranicí BW. Hranice se musí posunout tak, aby se vyhodnotili všichni kandidáti se shodnou cenou řetězení.

Další ze statistik, která byla při experimentech zpracována, prezentuje průměrnou cenu cíle v závislosti na pozici kandidáta ve větě doplněnou o rozptyl hodnot.

Z grafu na obrázku 8.8 vyplývá, že průměr ceny cíle kandidátů a stejně tak i rozptyl mají vzestupnou tendenci. Jsou zde patrné i výrazné změny průběhu křivky na samém počátku a konci promluv, což se dalo předpokládat s ohledem na rozložení absolutních hodnot ceny cíle a komentáře uvedené k předchozí statistice.

Průběh křivky průměrných cen cíle vede k úvaze, zda by nebylo vhodnější pro algoritmy VITBEAM a VITPRUNE prohledávat graf kandidátů jednotek v opačném směru, tj. od konečného kroku K směrem ke kroku 1. Vzhledem k vyšším hodnotám průměrné ceny cíle i rozptylu u jednotek umístěných blíže ke konci promluvy, by mohlo být rozlišení kumulativní ceny jednotlivých průběžných cest grafem významnější. Při prořezávání by tak mohly být v grafu



Obrázek 8.8: Průměrná hodnota ceny cíle a její rozptyl v závislosti na pozici kandidátů v promluvě resp. prozodické klauzi.

ponecháni spíše kandidáti, jež více korespondují s optimální cestou, než při standardním prohledáváním. Tato úprava nemá žádný efekt pro základní Viterbiův algoritmus VITBASE, kde bude rychlost nalezení optimální cesty S i výsledná matematická kvalita Q vždy shodná. U algoritmů VITBEAM a VITOPT by mohla být rychlost S nepatrně vyšší díky rychlejšímu nárůstu kumulativních cen cest, takže by se měly více projevit optimalizace popsané v oddílech 6.1.1.2 a 6.1.1.4. Ze stejného důvodu byl významnější efekt očekáván u kvality, protože by se měly rychleji projevit cesty, které by byly s větší pravděpodobností patřit mezi nejlepší.

K ověření či vyvrácení této hypotézy byl uskutečněn experiment, při němž se provedla syntéza testovací množiny vět ve všech kombinacích parametrů, jak bylo uvedeno v oddíle 8.1.4 v části hodnocení výkonu, ovšem s tím rozdílem, že prohledání grafu kandidátů probíhalo v opačném směru[5)]. Následně pak byly získané výsledky porovnány s výstupy hledání v dopředném směru. U některých nastavení hodnot parametrů bylo sice inverzním hledáním dosaženo mírného zlepšení alespoň u ukazatele kvality Q nebo u rychlosti S, ale celkově se rozdíl ukázal být nepatrný. V průměru nad množinou všech 330 kombinací parametrů však byly oba ukazatele mírně horší, a to o $\overline{\Delta Q} = 1{,}28$ ‰ a $\overline{\Delta S} = -0{,}03$. Lze tak konstatovat, že technika zpětného prohledávání grafu kandidátů u algoritmů VITBEAM a VITPRUNE nevede k významnému urychlení nebo zlepšení matematické kvality.

[5)] Při prohledávání v opačném směru byl používán revertovaný graf a v opačném směru musely být vyhodnocovány i ceny řetězení.

8.2 Algoritmy využívající řetězců s nulovou cenou řetězení (ZCC)

V průběhu prvotních experimentů zaměřených na optimalizaci Viterbiova algoritmu výběru jednotek bylo pozorováno, že velká část syntetizované promluvy se skládá z různě dlouhých řetězců, které odpovídají úsekům původní nahrané promluvy z řečového korpusu. Vzhledem k tomu, že segmenty řeči tvořící tyto řetězce na sebe navazují v původní nahrávce řečníka, je možné je při syntéze řeči zřetězit bez jakýchkoliv modifikací a rizika vzniku nežádoucích artefaktů. Zároveň platí, že v takových případech je cena řetězení C^{c} mezi nimi automaticky nulová, aniž by ji bylo nutné počítat. U algoritmů výběru jednotek (včetně VITBASE a jeho modifikací) se pak sklon k použití řetězců s nulovou cenou řetězení jeví jako přirozený důsledek snahy minimalizovat celkovou kumulativní cenu, zejména při použití rozsáhlé databáze řečových segmentů, kde lze předpokládat dostatečné množství realizací jednotek schopných takové řetězce vytvořit.

Další experimenty jsou směrovány k vytvoření nového algoritmu výběru jednotek, jehož základní myšlenkou je pokusit se využít vlastností řetězců s nulovou cenou řetězení, jež budou dále označovány zkratkou *ZCC* (z angl. *Zero Concatenation Cost*).

8.2.1 Vlastnosti ZCC řetězců

Kumulativní cenu izolovaného ZCC řetězce lze zapsat zjednodušením vztahu 4.3 původně definovaného na straně 20:

$$ {}_{zcc}C(t_m^M, u_m^M) = \sum_{k=m}^{M} C^{\mathrm{t}}(t_k, u_k), \tag{8.1} $$

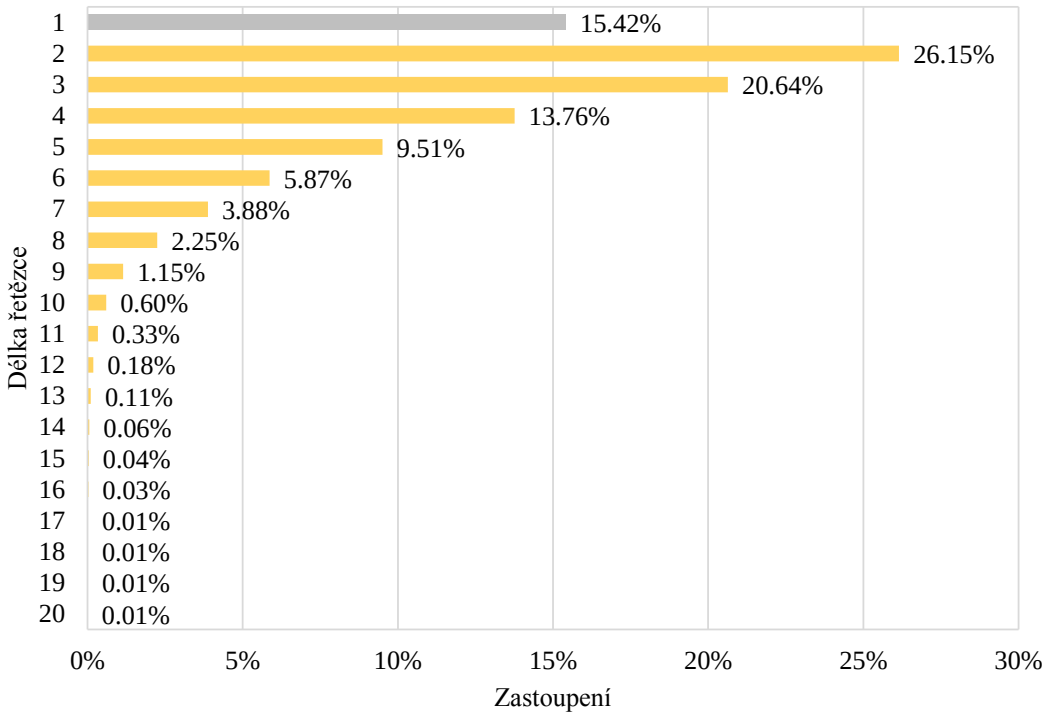
kde m je počáteční a M koncová pozice jednotek v rámci celé syntetizované promluvy. Cena řetězení C^{c} realizací jednotek je nulová a je tak ze vztahu vypuštěna.

V případě, že by byl ZCC řetězec vhodně umístěn v syntetizované promluvě a pozice v něm zahrnutých realizací jednotek by přesně odpovídala specifikaci cíle, byla by i cena cíle C^{t} pro všechny kandidáty nulová a tedy i kumulativní cena celého řetězce ${}_{zcc}C$ by také byla rovna nule.

Pokud by takový ZCC řetězec dokonce obsáhl celou požadovanou promluvu (klauzi), byla by posluchači v důsledku přehrána původní nahrávka z řečového korpusu bez jakýchkoliv úprav, tj. v objektivně nejvyšší možné kvalitě, které je možné konkrétním TTS systémem dosáhnout.

8.2.2 Analýza výskytu ZCC řetězců v syntetizovaných větách

Před zahájením návrhu algoritmů byla provedena analýza použitých realizací řečových jednotek při syntéze základním Viterbiovým algoritmem VITBASE, a to nad množinou 10 000 náhodně vybraných promluv z novinových článků.

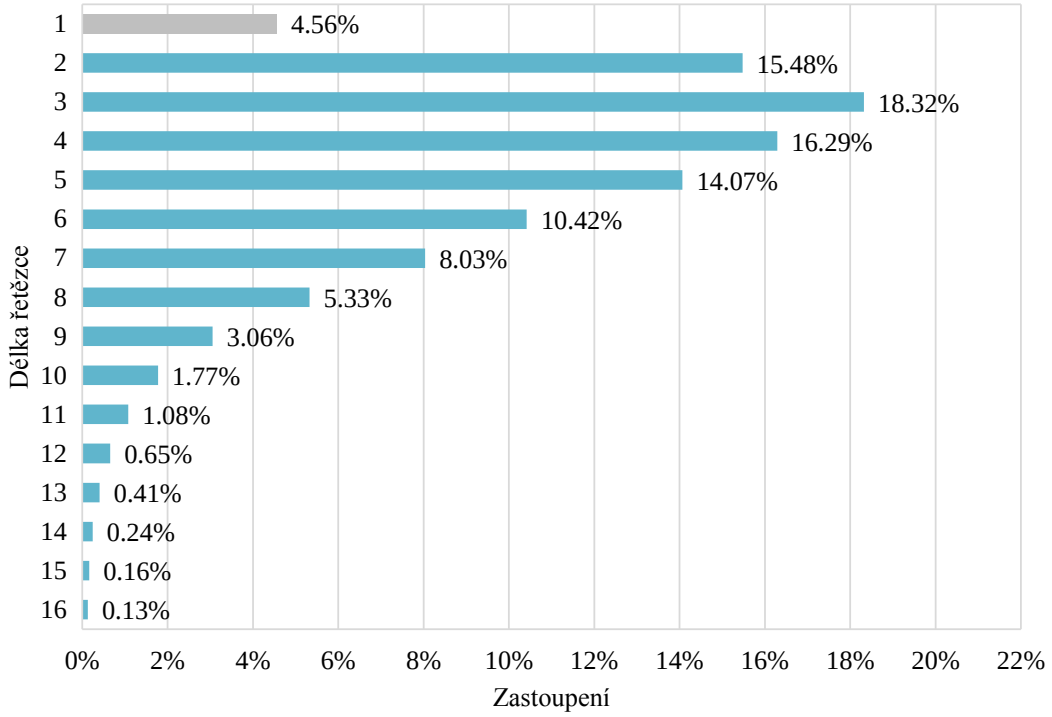


Obrázek 8.9: Graf rozložení počtu ZCC řetězců.

Nejprve byl zkoumán poměr zastoupení ZCC řetězců obsažených ve vybraných optimálních sekvencích s globálně minimální kumulativní cenou a ukázalo se, že tvoří přibližně 85 % z celkového počtu a že tedy jednoznačně převažují (viz graf na obrázku 8.9). Zbývajících 15 % připadá na samostatné

jednotky, které lze ve své podstatě chápat jako speciální případ ZCC řetězců o délce jedna. V grafu jsou počty ZCC řetězců rozděleny podle jejich délky a je z něj zřejmé, že nejvíce jsou zde zastoupeny řetězce délky 2, 3 a 4. Dále se ukazuje, že se v syntetizovaných větách vyskytuje nemalé množství ještě delších ZCC řetězců, jejichž počet, jak se dalo očekávat, s rostoucí délkou klesá.

Patrně více vypovídající statistika je uvedena na obrázku 8.10, na níž je vyjádřen počet realizací jednotek v závislosti na jejich příslušnosti do ZCC řetězců dané délky. Z ní vyplývá, že samostatně stojící řečové segmenty mimo ZCC řetězce zaujímají pouze 5 % z celkového množství použitých realizací jednotek.

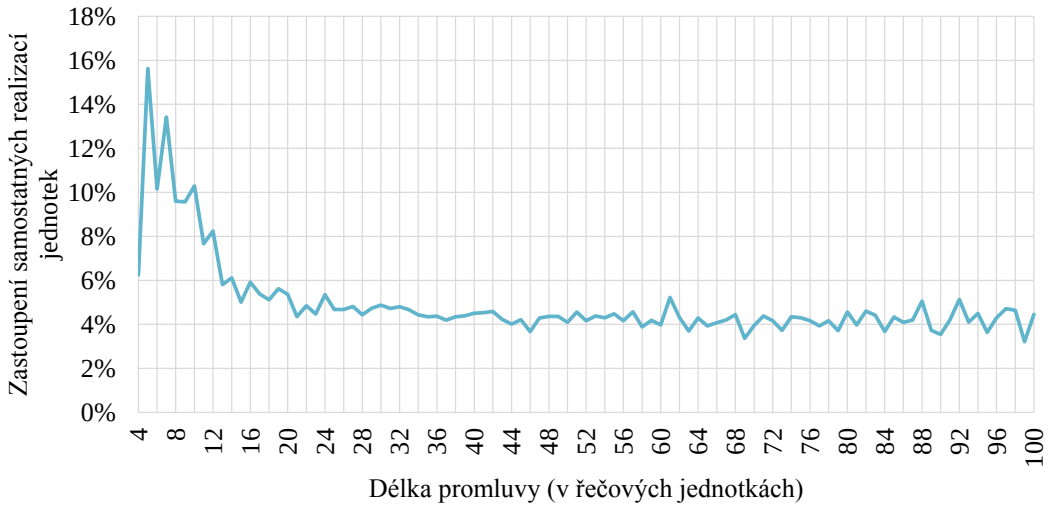


Obrázek 8.10: Graf zastoupení ZCC řetězců se zohledněním jejich délky.

Ze získaných údajů tedy vychází, že více než 95 % použitých realizací jednotek je součástí ZCC řetězců různých délek a tím i bylo potvrzeno úvodní tvrzení, že velká část syntetizované promluvy je tvořena souvislými úseky z původního řečového korpusu. Zároveň to také znamená, že pravděpodobně

nebude možné využít pouze ZCC řetězce, ale bude nutné při konstrukci algoritmů počítat s tím, že k dosažení ideálních hodnot v matematickém měřítku kvality, bude nutné pracovat i se samostatnými jednotkami. Důvodem výskytu samostatných jednotek může být buď to, že daný kontext jednotek v promluvě není žádným ZCC řetězcem pokryt nebo že cena spojení či cíle byla v dané pozici pro dostupné řetězce moc vysoká.

Další zpracovaná statistika slouží k ověření toho, zda nějakým způsobem nekoreluje množství samostatných realizací jednotek s délkou generovaných promluv.



Obrázek 8.11: Zastoupení samostatných realizací jednotek v závislosti na délce syntetizovaných promluv.

Graf na obrázku 8.11 ukazuje, že křivka se rychle přibližuje hranici 5 % a kolem ní pak osciluje. Hodnota tak odpovídá zjištěnému počtu samostatných segmentů řeči na celé testovací množině syntetizovaných klauzí bez ohledu na jejich délku. Vyšší zastoupení bylo naměřeno pouze u promluv složených z méně než 16 jednotek, což lze připsat spíše statistické chybě, protože počet takto krátkých promluv byl velmi nízký (např. promluva o délce pět jednotek byla pouze jedna). Navíc u vět s nízkým počtem jednotek navyšuje každý samostatný segment měřenou hodnotu velmi významným způsobem (např. u věty o délce 5 jednotek má 1 samostatná jednotka zastoupení 20 %). I přesto křivka nepřesáhla hodnotu 16 % u žádné z délek.

Z průběhu křivky lze tedy odvodit, že míra zastoupení samostatných

řečových segmentů není ovlivněna délkou syntetizované klauze. Při návrhu algoritmu založeného na ZCC řetězcích tak nebude nutné počet jednotek ve výsledné promluvě zohledňovat.

8.2.3 Strategie algoritmů na bázi ZCC řetězců

U původního Viterbiova algoritmu VITBASE, případně u jeho vylepšené verze VITOPT, dochází v podstatě k vyhodnocení všech přípustných cest grafem tvořeným všemi možnými kandidáty řečových jednotek. Při úvaze nad možnostmi využít vlastností ZCC řetězců se formulace řešeného problému nijak nemění. I nadále se pracuje s grafem tvořeným uzly, které jsou uspořádané do K množin odpovídajících jednotkám syntetizované promluvy. Z každé množiny uzlů s indexem k vedou orientované hrany směrem ke všem uzlům množiny $k + 1$, přičemž každé hraně je přiřazena cena řetězení kandidátů, jež spojuje. Vzniká tak acyklický orientovaný graf s ohodnocenými uzly i hranami, a cílem algoritmu je nalézt cestu s nejnižší kumulovanou cenou, která začíná některým uzlem z množiny s indexem 1 a končí jiným uzlem z množiny s indexem K.

Zásadním problémem je právě ohodnocení všech hran, resp. obrovské množství potřebných výpočtů cen řetězení C^{c}. Cílem při návrhu algoritmů založených na ZCC řetězcích je to, aby se v maximální míře omezila nutnost těchto výpočtů a snížil se počet zkoumaných cest grafem. Strategie takového hledání je postavena na předpokladu, že optimální cesta bude procházet údolími tvořenými některými ze ZCC řetězců (tj. úseky s minimální kumulativní cenou).

Jednou z významných výhod všech algoritmů využívajících ZCC řetězce je to, že pokud by databáze řečových jednotek obsahovala vhodný řetězec schopný obsáhnout celou požadovanou promluvu (tzv. *úplný ZCC řetězec*), pak dojde buď okamžitě, nebo alespoň velmi rychle, k jeho nalezení a použití pro syntézu. Kromě téměř nulových nároků na výpočetní výkon daného zařízení, bude posluchači přehrána promluva v ideální kvalitě. Test na výskyt takového řetězce kandidátů by samozřejmě bylo možné zapracovat i do ostatních algoritmů, nicméně u metod na bázi ZCC řetězců jde ve své podstatě o zajímavý vedlejší efekt a implicitní vlastnost.

Ať již je vyhledání úplného ZCC řetězce použito v libovolném algoritmu, je nezbytné vždy hodnotit i jeho vhodnost z hlediska umístění v promluvě a z něj plynoucích prozodických charakteristik[6]. Podle definice stanovené v této práci má ZCC řetězec nulovou cenu řetězení mezi všemi realizacemi jednotek. To ovšem nic nevypovídá o vhodnosti umístění v něm obsažených kandidátů, tj. o ceně cíle C^{t}. V praxi je nejvíce problematický koncový úsek promluvy, kde klesá nebo naopak stoupá intonace podle toho, zda se například jedná o konec věty oznamovací nebo konec klauze uprostřed souvětí, za níž následuje další část promluvy. Pokud by se použil ZCC řetězec, u něhož by byla intonace v původní promluvě nahrané řečníkem v rozporu s požadovanou promluvou, měli by kandidáti v tomto místě ZCC řetězce velmi vysoké hodnoty ceny cíle C^{t} a v konečném důsledku by vygenerovaná řeč zněla nepatřičně nebo by mohlo dojít i ke změně jejího smyslu.

Řešení tohoto problému je testovat u kandidátů v ZCC řetězci soulad s typem prozodému určujícím intonační vlastnosti daného úseku promluvy. Například v systému ARTIC se rozlišují celkem 4 možné typy prozodému (Romportl, 2006) a u všech realizací jednotek v databázi řečových segmentů je uložen příznak s touto informací. V době samotné syntézy řeči je určení typu prozodému součástí specifikace cíle, což lze využít k přímému porovnání s hodnotou u kandidátů uvnitř ZCC řetězce, a to ještě před vyhodnocením ceny cíle. O nevyhovující kandidáty by pak byl ZCC řetězec pokrácen, tudíž by již nepokrýval celou promluvu a pracovalo by se s ním jako ostatními kratšími řetězci.

8.2.4 Algoritmy hledání rozvojem nejslibnější cesty s využitím ZCC řetězců

První zástupce metod výběru optimální posloupnosti kandidátů s využitím ZCC řetězců používá, na rozdíl od předchozích algoritmů, techniku prohledávání grafu rozvojem cesty s nejslibnějším ohodnocením. Bude značen kódem *ZCCBEFS* vytvořeným z užívaného angl. názvu *Best–First Search*. Základní strategie algoritmu je postavená na tom, že se vždy rozvíjí cesta s aktuálně nejnižší hodnotou kumulativní ceny C^* a pokud na ni v daném místě navazuje

[6]Systém ARTIC se řadí mezi systémy, u nichž je prozodie (průběh frekvence F_0) modelována implicitně prostřednictvím hodnoticí funkce ceny cíle C^{t}, jež také porovnává pozici řečových segmentů v požadované promluvě a v promluvě z řečového korpusu.

některý ze ZCC řetězců, pak se automaticky použije k jejímu rychlejšímu rozvoji, tj. k přeskočení kroků (pozic) grafu, jež jsou takovým řetězcem pokryty. Předtím než bude algoritmus ZCCBEFS detailněji popsán a hodnocen, bude následující oddíl věnován obecnému prohledávání rozvojem nejslibnější cesty v kontextu řešené úlohy.

8.2.4.1 Klasický algoritmus hledání rozvojem nejslibnější cesty

Při použití Viterbiova algoritmu se postupovalo v krocích a v každém z nich se vždy zpracovávala celá množina kandidátů na dané pozici. V případě algoritmu hledání cesty grafem nazývaného v anglickém jazyce *Best–First Search* (zkráceně *BEFS*[7]), se vždy rozvíjí pouze nejslibnější dosud nalezená cesta, a to bez ohledu na to, zda již byly ostatní cesty rozvinuty do stejné délky.

U obecného algoritmu BEFS je každá cesta ohodnocena svou cenou, typicky určenou heuristickou funkcí H, která je definována tak, aby co nejlépe vystihovala řešený problém. Klasický postup prohledávání grafu rozvojem nejslibnější cesty má následující podobu:

> Algoritmus používá dvě množiny. První z nich je označená *OPEN* a obsahuje všechny stavy (uzly), které ještě nebyly zpracovány. Pokud se uzel zpracuje, je přesunut do druhé množiny značené *CLOSED*, aby se zabránilo opakované expanzi. U každého uzlu je kromě ohodnocení uchováván i odkaz na nejlepšího předchůdce ve smyslu minima hodnoticí funkce.
>
> 1. Uzly odpovídající počátečnímu stavu jsou ohodnoceny a zařazeny do množiny OPEN. Množina CLOSED je na počátku prázdná.
> 2. Z množiny OPEN se vybere uzel u^* s nejnižší cenou H.
> 3. Pokud bylo uzlem u^* dosaženo cíle, pak se hledání ukončí a zpětným trasováním přes odkazy na nejlepší předchůdce (angl. *backtracking*) je zrekonstruována vítězná cesta.

[7] Zkratka byla zvolena kvůli odlišení od zkratky pro hledání do šířky – angl. Breadth-First Search (*BFS*).

4. Pokud není u^* koncovým uzlem, pak se provede jeho expanze směrem k následným uzlům v grafu a uzel u^* se přesune do množiny CLOSED.
5. Pro všechny uzly, do nichž byla cesta expandována:
   (a) Pokud není v množině CLOSED, pak se provede ohodnocení uzlu funkcí H, uloží se k němu odkaz na uzel u^* jako nejlepšího předchůdce a je zařazen do množiny OPEN.
   (b) Pokud je v množině CLOSED, pak se provede také jeho ohodnocení funkcí H a pokud je výsledná cena nižší než dosud nalezená, změní se u něj odkaz na nejlepšího předchůdce na uzel u^*.
6. Algoritmus se vrací zpět na bod 2, dokud není nalezeno řešení nebo dokud není množina OPEN prázdná (v takovém případě algoritmus končí bez nalezení řešení).

8.2.4.2 Algoritmus BEFS v úloze výběru jednotek

I obecný algoritmus BEFS lze použít k řešení úlohy výběru jednotek u TTS systému, i když u něj nelze předpokládat výrazné zrychlení proti referenčnímu Viterbiovu algoritmu.

HODNOTICÍ FUNKCE
K hodnocení cest se nabízí kumulativní cena C^*. Jejím použitím by však byl algoritmus poměrně neefektivní, protože přírůstky hodnot cen jednotlivých cest při rozvoji o jeden další krok jsou si velmi podobné a docházelo by k vyhodnocení velké části ze všech cen řetězení jako u algoritmu VITBASE. Proto dává smysl zahrnout do hodnocení cest i vzdálenost od počátku grafu a zvýhodnit delší cesty. Tím se charakter hledání blíží algoritmu označovanému jako *A-star* (A^*), u něhož se používá hodnoticí funkce zohledňující vzdálenost od začátku a zároveň i vzdálenost zbývající do konce. U algoritmu BEFS byla použita hodnoticí funkce ve tvaru:

$$H(u_p(i)) = C^*(u_p(i)) + \alpha(P - p), \tag{8.2}$$

kde $u_p(i)$ je realizace jednotky s indexem i z množiny realizací na pozici p[8)] v promluvě. Konstanta P udává celkovou délku promluvy v počtu používaných jednotek. Parametr α udává míru zvýhodnění delších cest proti cestám kratším. Čím bude hodnota α vyšší, tím rychleji dospěje algoritmus k výsledku, ale zároveň bude mít větší sklon k upřednostnění lokálních minim v neprospěch celkové ceny vítězné cesty.

ZÁRUKA NALEZENÍ CESTY

S ohledem na rozložení kandidátů jednotek v prohledávaném grafu lze algoritmu BEFS přisoudit některá další specifika. U zcela obecného grafu nemusí algoritmus BEFS ve své nejjednodušší verzi (bez detekce toho, zda byl již daný uzel expandován) nalézt vůbec žádné řešení, a to kvůli sklonu k preferenci lokálního zisku, kdy může hledání skončit v nekonečné smyčce. Graf tvořený kandidáty jednotek a hranami umožňujícími spojení s libovolným kandidátem na vyšší pozici je acyklický, takže k takové situaci nikdy nemůže dojít, protože z každého uzlu existují vždy pouze cesty do uzlů na vyšší pozici. Při prohledávání grafu algoritmem BEFS v řešené úloze je tedy zaručeno nalezení alespoň jedné cesty, a to i při agresivním prořezání grafu například dle ceny cíle C^{t}. Pokud se navíc nepoužije některá z níže popsaných optimalizací, je dokonce zaručeno i nalezení cesty s globálně minimální kumulativní cenou C.

OPTIMALIZACE PŘI EXPANZI

Další specifické chování algoritmu BEFS se vztahuje k volbě parametru α hodnoticí funkce. Pokud by se použilo nastavení $\alpha = 0$ (důraz na kvalitu), stává se funkce H monotónní – neklesající. Z toho plyne, že do uzlů, jež byly v danou chvíli nejslibnější, a tedy byla provedena jejich expanze, již nemůže vést žádná jiná cesta s nižším ohodnocením. Nemůže tedy nastat podmínka 5b z výše uvedeného postupu algoritmu BEFS a jinými slovy platí, že při expanzi nejslibnějšího uzlu lze vynechat přechod k následujícím uzlům zařazených do množiny CLOSED, aniž by došlo ke zhoršení ceny vítězné cesty.

V případě použití parametru $\alpha > 0$ (důraz na rychlost a ztráta záruky nalezení globálního minima) již uvedené specifikum algoritmu neplatí ve všech případech, neboť funkce H ztrácí monotónnost. Protože však platí, že expanzí libovolného uzlu se lze posunout v grafu pouze o jednu pozici,

[8)] Pozice p odpovídá svým významem kroku k užívaného u předchozích algoritmů.

pak také platí, že při expanzi nejslibnějšího uzlu lze vynechat následující uzly zařazené do množiny CLOSED, jejichž hodnota H je nižší než součet hodnocení expandovaného uzlu a ceny cíle následujícího uzlu mínus hodnota α.

Výše uvedenou vlastnost algoritmu BEFS lze použít k jeho optimalizaci, v jejímž důsledku dochází ke snížení celkového počtu vyhodnocení cen řetězení, a to bez dalšího navýšení kumulativní ceny vítězné cesty.

PROŘEZÁVÁNÍ

Doplněním prořezávání do optimalizovaného Viterbiova algoritmu vznikl algoritmus VITPRUNE, jenž dosahuje poměrně zajímavé míry urychlení při zachování kvality. Zároveň umožňuje jednoduchou změnou parametrů prořezávání zvýšit rychlost, i když při agresivním nastavení doprovázenou zhoršenou kvalitou. Do algoritmu BEFS v řešené úloze lze snadno zapracovat optimalizaci kombinující stejné techniky, jaké byly popsané v oddílech 8.1.3 a 8.1.4. Při expanzi každého uzlu by se následující kandidáti seřadili podle ceny cíle C^{t} a rozvoj cesty by se provedl pouze pro specifikovaný počet nejlepších uzlů. Algoritmus je tak obohacen o nový parametr určující hloubku expanze ($DEEP$), jenž při hodnotách $DEEP > 0$ do něj zavádí prvky hledání typu $BEAM$.

Tímto prořezáním kandidátů však vzniká nevhodný efekt, kdy se v podstatě na každé pozici grafu pracuje pouze s kandidáty s nejnižší cenou cíle C^{t} v počtu daným parametrem $DEEP$[9)]. Tomu lze předejít tak, že se expanze uzlu provádí tak dlouho, dokud není skutečně zpracováno $DEEP$ nejlepších kandidátů, mezi něž se nepočítají uzly vynechané optimalizací při expanzi popsané v předcházejícím textu. Takto upravený algoritmus pak provádí rozvoj nejslibnější cesty do specifikované hloubky z množiny všech přípustných následovníků v dané chvíli. Znamená to také zvýšení počtu vyhodnocovaných uzlů, ale to lze také zefektivnit prořezáváním popsaném v oddíle 8.1.3, tj. vyhledávání $DEEP$ nejlepších kandidátů lze ukončit, pokud již bylo zpracováno $DEEP$ uzlů a cena cíle C^{t} zkoumaného uzlu již přesahuje dosud nalezenou minimální cenu řetězení C^{c}.

KOMENTÁŘ K EXPANZI UZLŮ NA POČÁTKU GRAFU

Statistika uvedená na straně 69 v oddíle 8.1.5 ukazuje, že ceny cíle C^{t}

[9)] Stejného výsledku by se dosáhlo aplikací prořezání grafu dle ceny cíle popsaného v oddíle 8.1.4 s nastavením parametrů $TW = DEEP$ a $TWR = 0$.

přiřazené kandidátům na první pozici mají velmi nízký počet odlišných hodnot. Z toho také plyne malá míra odlišení cen jednotlivých cest na začátku hledání, zejména pak při přechodu z první na druhou pozici grafu. Jinými slovy to znamená, že v množině cest na počátku hledání má velká část z nich rovnocenné ohodnocení a nelze rozhodnout, která cesta je nejslibnější.

Pokud by se při expanzi cest vždy zpracovaly všechny následující uzly (tj. parametr $\alpha = 0$ a „$DEEP = \infty$“), pak se v průběhu hledání dříve či později dostane na první místo cesta s globálně minimální cenou. Když se však upřednostní delší cesty nebo se omezí hloubka expanze, může dojít k tomu, že se již na počátku hledání upřednostní cesta, jejíž konečná kumulativní cena bude vyšší, protože jiné vhodnější uzly nebyly ani rozvíjeny. Jde o podobný problém popsaný u optimalizovaných verzí základního Viterbiova algoritmu VITORIG, kdy je nejslibnější cesta vybrána v podstatě náhodně. Jako řešení by se nabízelo upravit algoritmus BEFS tak, že se na počátku provede expanze všech uzlů na pozici $p = 1$ s nejnižším ohodnocením a až pak se zahájí postupný rozvoj nejslibnější cesty. Tím by sice došlo ke zmírnění rizika použití lokálního minima s nevýhodným pokračováním, ale zároveň by se významně omezila výhoda rychlejšího prohledání grafu daná podstatou hledání rozvojem nejslibnější cesty (mezi kandidáty na první a druhé pozici by došlo k výpočtu všech přípustných cen řetězení stejně jako u základního Viterbiova algoritmu).

Způsob řešení tohoto problému nebyl nakonec detailně zkoumán a vyhodnocován, protože algoritmus BEFS není vhodný k rychlému prohledávání celého grafu, nicméně je velmi efektivní k prohledání kratších úseků jako součást jiných algoritmů (viz algoritmy popsané v oddílech 8.2.5 a 8.2.9).

IMPLEMETACE ALGORITMU BEFS

Přestože algoritmus označovaný zkratkou BEFS nemá předpoklady významně urychlit proces výběru jednotek, byl v rámci experimentů implementován a byla otestována jeho výkonnost (rozsah hodnot parametrů viz tabulka B.3 v příloze B), protože je základem jeho optimalizované verze využívající ZCC řetězce.

Výsledný postup hledání optimální cesty grafem pomocí algoritmu BEFS je při zohlednění předchozích komentářů následující[10)]:

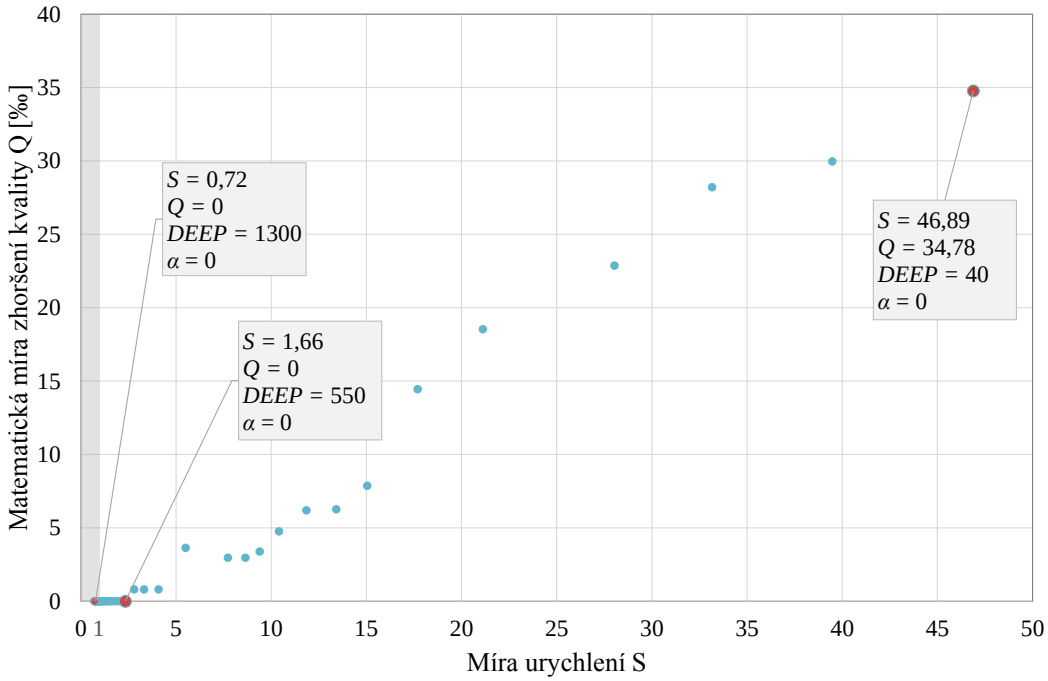
[10)] Algoritmus používá množiny OPEN a CLOSED se stejným významem jako u obecného algoritmu.

1. Ke všem uzlům grafu se vypočítá cena cíle C^{t}.
2. Pro uzly odpovídající kandidátům na pozici $p = 1$ se cena cíle C^{t} stává i hodnotou kumulativní ceny C^*. Uzlům se následně přiřadí ohodnocení vypočtené pomocí funkce H (viz vztah 8.2) a zařadí se do množiny OPEN.
3. Z množiny OPEN se vybere uzel u^* s nejnižší hodnotou H.
4. Pokud bylo uzlem u^* dosaženo konce grafu na pozici P, pak se hledání ukončí a zpětným trasováním přes odkazy na nejlepší předchůdce je zrekonstruována vítězná cesta.
5. Pokud není u^* koncovým uzlem, pak se provede jeho expanze směrem k uzlům na následující pozici v grafu a uzel u^* se přesune do množiny CLOSED.
6. Pro všechny uzly u, do nichž byla cesta expandována, se provádí vnitřní cyklus:
   (a) Pokud je uzel u v množině CLOSED a zároveň platí podmínka $[H(u^*) + C^{\mathrm{t}}(u) - \alpha] > H(u)$, pak se další zpracování uzlu u přeskočí, a to bez jeho započítání mezi zpracované.
   (b) Provede se ohodnocení uzlu funkcí H, což zahrnuje i výpočet ceny řetězení C^{c} s předcházejícím uzlem u^*.
   (c) Pokud uzel u není v množině CLOSED, pak se vloží do množiny OPEN a uloží se k němu odkaz na uzel u^* jako nejlepšího předchůdce.
   (d) Pokud je uzel u již v množině CLOSED, pak pokud je výsledná cena $H(u)$ nižší než dosud nalezená, změní se u něj odkaz na nejlepšího předchůdce na uzel u^*.
   (e) Pokud již bylo zpracováno $DEEP$ kandidátů, pak je vnitřní cyklus ukončen za podmínky, že ohodnocení H zkoumaného uzlu u přesahuje ohodnocení dosud nejhorší hodnoty H všech zpracovaných uzlů.

7. Algoritmus se vrací zpět na bod 3, dokud není nalezeno řešení nebo dokud není množina OPEN prázdná (to by znamenalo ukončení bez nalezení řešení, ale tato situace u popisované úlohy nemůže nastat).

HODNOCENÍ ALGORITMU BEFS

Pokud je algoritmus BEFS využit k vyhledání celé cesty grafem testovacích promluv, pak jeho výkonnost[11)] je nízká (viz obrázek 8.12) a při bezpečném nastavení parametrů ($Q = 0{,}00$ ‰) bylo docíleno zrychlení pouze $S = 1{,}66$ (viz tabulka 8.5). Pokud byla hloubka expanze (parametr *DEEP*) vysoká, pak byla rychlost dokonce nižší než u algoritmu VITBASE ($S < 1$). Naopak vyšších rychlostí bylo dosaženo při velmi nízkých hodnotách parametru *DEEP*, které však znamenají zhoršení matematické kvality Q.



Obrázek 8.12: Matematická míra zhoršení kvality ve vztahu ke zrychlení pro různé parametry algoritmu BEFS. Na vodorovné ose je vyznačena i hodnota $S = 1$, protože pro některé kombinace hodnot parametrů byla rychlost $S < 1$, tj. algoritmus byl pomalejší než referenční algoritmus VITBASE.

[11)] Rozsahy hodnot parametrů použitých při testování výkonu algoritmu BEFS jsou uvedeny v tabulce B.3 v příloze B.

Tabulka 8.5: Hodnocení výkonu algoritmu hledání rozvojem nejslibnější cesty pro bezpečné hodnoty parametrů

Vlastnost	Hodnota
Kód algoritmu	BEFS
Míra urychlení (S)	1,66
Matematická míra zhoršení kvality (Q)	0,00 ‰
Míra fragmentace (CD)	29,39 %
Hodnoty parametrů: $\alpha = 0,\ \mathrm{DEEP} = 550$	

8.2.4.3 Optimalizace hledání rozvojem nejslibnější cesty za pomoci ZCC řetězců

První z algoritmů využívající ZCC řetězce k urychlení procesu výběru jednotek bude označován zkratkou ZCCBEFS. Vznikl změnou chování algoritmu BEFS při expanzi uzlu grafu, kdy v případě, že rozvíjeným uzlem začíná ZCC řetězec, pak se přednostně provede i expanze všech kandidátů, které jej tvoří.

Strategií algoritmu ZCCBEFS je pokusit se zvýšit rychlost hledání na základě předpokladu, že pokud kandidáti navazující na nejslibnější uzel byli zároveň přímými sousedy v řečovém korpusu, pak se u nich zvyšuje pravděpodobnost, že budou součástí vítězné cesty díky vzájemné nulové ceně řetězení C^{c}.

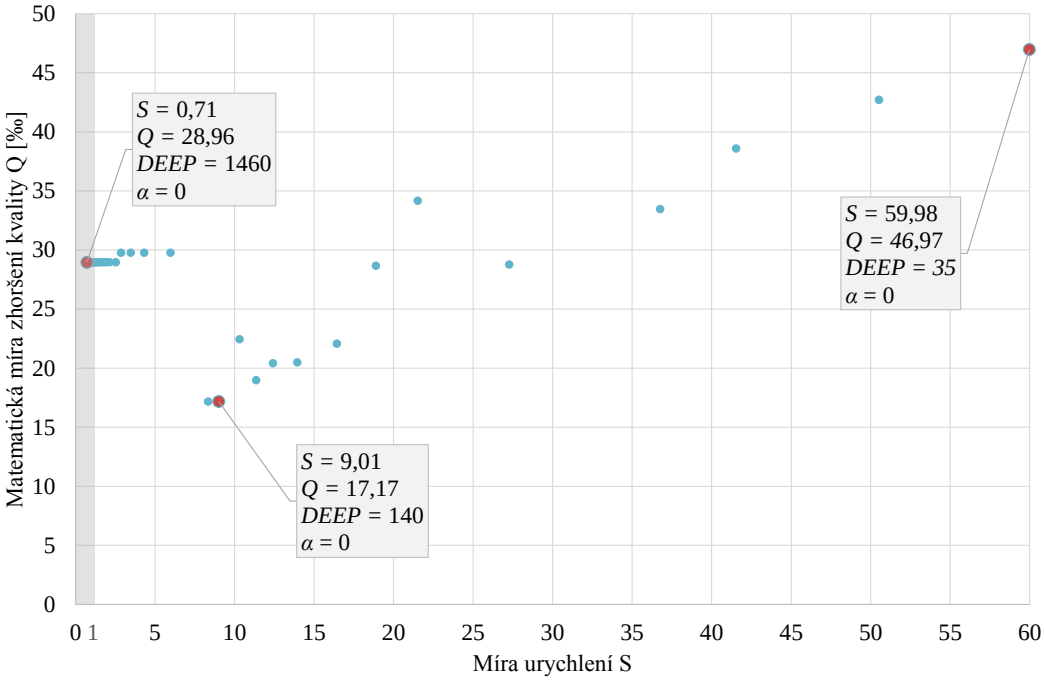
Zápis postupu algoritmu BEFS ze strany 83 by tedy byl rozšířen v bodě 6 takto:

> 6. Pro všechny uzly, do nichž byla cesta expandována:
>
> (a) Pokud uzel pokračuje ZCC řetězcem, pak pro všechny v něm obsažené navazující uzly provést expanzi.
>
> (b) – (e) *Algoritmus je v těchto částech shodný s původními body 6a až 6e.*

HODNOCENÍ ALGORITMU ZCCBEFS

Z grafu na obrázku 8.13 je patrné, že při použití algoritmu ZCCBEFS nebylo ani v jednom případě dosaženo výsledků shodných s referenčním algoritmem VITBASE (rozsah hodnot parametrů viz tabulka B.4 v příloze B), tj. nebylo

dosaženo matematické kvality $Q = 0{,}00$ ‰. Nejnižší míra zhoršení matematické kvality $Q = 17{,}70$ ‰ byla získána u varianty dávající zrychlení $S = 9{,}01$ (viz tabulka 8.6). Zajímavým výsledkem u této varianty je míra fragmentace $CD = 27{,}99$ %, což je nižší hodnota, než u referenčního algoritmu VITBASE, kde byla hodnota $CD = 29{,}39$ %. Zlepšení tohoto ukazatele potvrzuje strategii algoritmu ZCCBEFS, kdy je upřednostňována expanze uzlů, jež jsou součástí ZCC řetězců, a algoritmus má tak sklon k výběru souvislejších celků.



Obrázek 8.13: Matematická míra zhoršení kvality ve vztahu ke zrychlení pro různé parametry algoritmu ZCCBEFS. Na vodorovné ose je vyznačena i hodnota $S = 1$, protože pro některé kombinace hodnot parametrů byla rychlost $S < 1$, tj. algoritmus byl pomalejší než referenční algoritmus VITBASE.

Nevýhodou algoritmu ZCCBEFS je, že při zvyšování hodnoty parametru $\alpha > 0$ se rychleji zhoršuje ukazatel matematické kvality, než u neoptimalizované verze BEFS. Takové chování algoritmu lze vyložit větším sklonem k prohledávání do hloubky díky přednostní expanzi uzlů, jež leží vedle sebe v řečovém korpusu.

Zisk zrychlení je vykoupen vysokou mírou zhoršení matematické kvality, která byla na první poslech slyšitelná. To, že nebyla dosažena v žádné z variant nastavení parametrů kvalita $Q = 0{,}00$ ‰, lze vysvětlit tím, algoritmus

v některých částech cesty grafem kandidátů upřednostní navazující realizace jednotek v ZCC řetězcích před vhodnějšími realizacemi (např. na konci grafu). Celkově lze říci, že algoritmus ZCCBEFS sám o sobě nevyhovuje cílům této práce a dále nebyl zkoumán, protože ani nenabízí další možnosti optimalizace, která by vedla k předpokládané míře urychlení procesu výběru jednotek.

Tabulka 8.6: Hodnocení výkonu algoritmu hledání rozvojem nejslibnější cesty s využitím ZCC řetězců pro nejlepší dosažený výsledek

Vlastnost	Hodnota
Kód algoritmu	ZCCBEFS
Míra urychlení (S)	9,01
Matematická míra zhoršení kvality (Q)	17,17 ‰
Míra fragmentace (CD)	29,32 %
Hodnoty parametrů: $\alpha = 0,\ \mathrm{DEEP} = 140$	

8.2.5 Algoritmus hledání heuristickým předvýběrem cest složených ze ZCC řetězců

Algoritmy BEFS a ZCCBEFS popsané v předchozím textu nedosahují očekávaných výsledků, a proto byly hledány jiné způsoby využití znalostí o struktuře prohledávaných grafů k urychlení procesu výběru jednotek. Další z testovaných algoritmů je založen na úvaze, že velmi dlouhé ZCC řetězce v grafu kandidátů jsou s nejvyšší pravděpodobností i součástí vítězné cesty.

8.2.5.1 Strategie algoritmu ZCCFRAME

Strategie hledání má v tomto případě podobný základ jako u hledání rozvojem nejslibnějšího kandidáta, ale místo s jednotlivými kandidáty se zde pracuje na úrovni ZCC řetězců s minimální specifikovanou délkou.

Nejprve se v grafu kandidátů vyhledají všechny ZCC řetězce delší než nastavené minimum. Poté jsou vhodným způsobem řazeny za sebe tak, aby na sebe mohly navazovat, čímž vznikne množina cest grafem. V těchto cestách je maximum pozic obsazeno kandidáty ze ZCC řetězců, avšak ne všechny pozice se podaří zaplnit. Cesty vytvořené v této fázi jsou tedy neúplné, a proto pro ně

bude v dalším textu užíváno pojmenování *kostra cesty*[12)] grafem. Algoritmus poté pokračuje postupným dohledáváním vhodných kandidátů k vyplnění prázdných pozic v kostrách a tím dochází ke spojení chybějících úseků mezi ZCC řetězci.

Algoritmus samotný je označen kódem *ZCCFRAME*, protože v první fázi vzniká množina koster (rámců – angl. frame) tvořících základ možných optimálních cest grafem. Hlavním cílem algoritmu je opět v maximální míře redukovat počet nutných výpočtů cen řetězení C^{c} mezi kandidáty. První fáze hledání, kdy se vytvářejí kostry cest, se dokonce obejde zcela bez jakéhokoliv výpočtu ceny řetězení. V druhé fázi postupného zaplňování prázdných pozic v kostrách je již nutné ceny řetězení vyhodnocovat a je tak potřeba vhodně zvolit techniku, jak chybějící úseky zaplnit s minimálními výpočetními nároky. Zde připadá v úvahu buď použít některou z variant Viterbiova algoritmu (VITPRUNE) nebo algoritmus BEFS, protože se jedná o algoritmy pracující po jednotlivých jednotkách.

8.2.5.2 Fáze hledání koster cest grafem

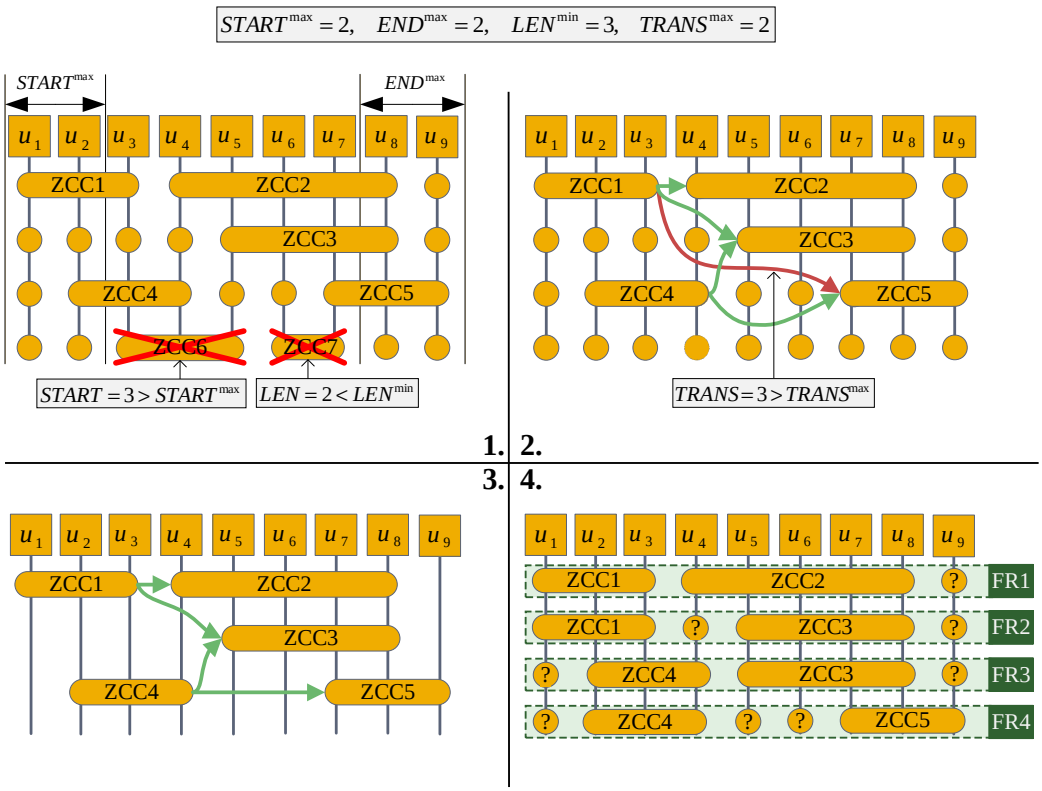
První fáze algoritmu ZCCFRAME, v rámci níž se v grafu kandidátů vyhledávají kostry cest, je řízena pomocí následujících parametrů:

- $LEN^{\min}$ – minimální délka ZCC řetězce kostry,
- $TRANS^{\max}$ – maximální vzdálenost mezi řetězci kostry,
- $START^{\max}$ – maximální vzdálenost prvního ZCC řetězce kostry od začátku grafu,
- $END^{\max}$ – maximální vzdálenost posledního ZCC řetězce kostry od konce grafu,
- *CHAINDEEP* – hloubka expanze ZCC řetězců koster,
- $FRAMECOUNT^{\max}$ – maximální počet rozvíjených koster,
- $CANDCOUNT^{\max}$ – maximální počet úplných resp. kandidátních koster (viz dále).

Schéma hledání koster cest grafem, včetně znázornění významu uvedených parametrů, je pak uvedeno na obrázku 8.14.

Při sestavování koster se nejdříve v grafu kandidátů vyhledá množina všech ZCC řetězců, jejichž délka v počtu jednotek splňuje kritérium definované

[12)] V této práci zavedený termín *kostra cesty* nemá žádnou souvislost s pojmem *kostra grafu*, který je užíván v teorii grafů.



Obrázek 8.14: Schéma hledání koster cest grafem. V části č. 1 je znázorněno vyhledání ZCC řetězců v grafu kandidátů a jejich prořezání, pokud nesplňují podmínky kladené na minimální délku nebo vzdálenost od počátku grafu. V části č. 2 jsou naznačena možná spojení mezi ZCC řetězci (zelenou barvou) a jedno spojení, která nesplňuje podmínku na maximální vzdálenost mezi ZCC řetězci (červená barva). Část č. 3 prezentuje schéma zjednodušeného grafu pouze se zpracovávanými ZCC řetězci. Poslední část č. 4 znázorňuje vzniklé kostry, které jsou značené FR1 až FR4 (z angl. názvu frame), a jsou zde s otazníky naznačeny pozice koster, které je nutno doplnit.

parametrem $LEN^{\min}$ a ostatní ZCC řetězce s kratší délkou jsou ignorovány (viz řetězec ZCC7 na obrázku 8.14.1). Prvky takto vybrané množiny lze uspořádat do pomyslného zjednodušeného grafu, jehož uzly reprezentují jednotlivé ZCC řetězce (viz obrázek 8.14.2 resp. 8.14.3). Vzdálenost mezi uzly je dána počtem jednotek mezi koncem a začátkem jim odpovídajících ZCC řetězců. Hrany grafu pak vedou pouze mezi uzly, které od sebe leží nejvýše ve vzdálenosti specifikované hodnotou parametru $TRANS^{\max}$ (omezení ilustrováno spojením mezi řetězci ZCC1 a ZCC5 na obrázku 8.14.2).

V grafu, který tímto postupem vznikl, se poté provádí hledání cest rozvojem nejslibnějšího kandidáta podobně jako u algoritmu BEFS a tím postupně vzni-

kají kostry cest původním grafem kandidátů (viz obrázek 8.14.4). Struktura grafu ZCC řetězců i cíle jeho prohledávání jsou však odlišné, a tomu bylo nutné přizpůsobit i další vlastnosti algoritmu.

První rozdíl je v konstrukci hodnoticí funkce H, kterou bylo nutné definovat tak, aby byla v souladu se strategií algoritmu, tj. upřednostňovala kostry s větším počtem vyplněných pozic a složených z delších ZCC řetězců. Po několika experimentech se nejvhodnější ukázala být kombinace počtu nevyplněných pozic kostry, počtu úseků mezi ZCC řetězci s nenulovou délkou a součtu cen cíle C^{t} všech kandidátů v kostře (na počátku pouze kandidátů uvnitř ZCC řetězců). Definice hodnoticí funkce H je následující:

$$H(f_i) = [E_u(f_i), E_p(f_i), CF^*(f_i)]\,, \tag{8.3}$$

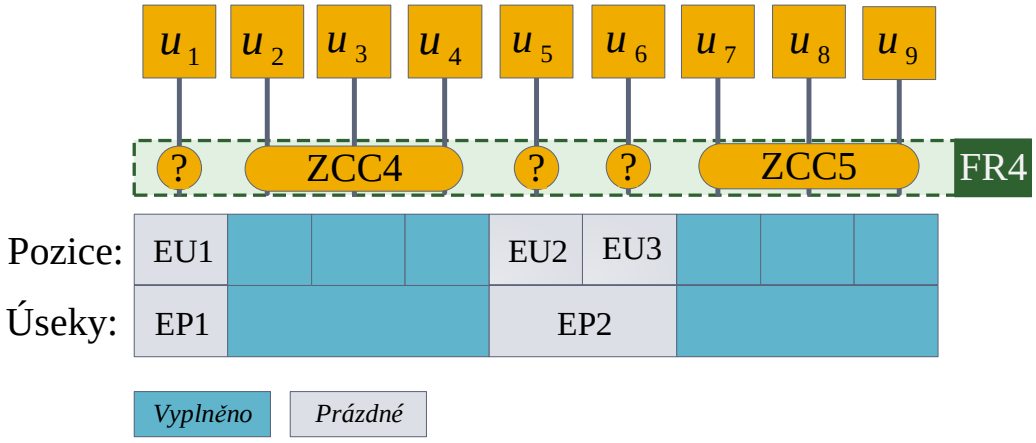
kde f_i je kostra (z angl. frame) s indexem i, $E_u(f_i)$[13)] je funkcí vracející počet nevyplněných pozic kostry a hodnota funkce $E_p(f_i)$[14)] odpovídá počtu souvislých úseků prázdných pozic (viz obrázek 8.15). Označení $CF^*(f_i)$ nese speciální verze funkce pro výpočet kumulativní ceny cesty grafem, jež byla upravena pro strukturu koster. Rozdíl proti standardní funkci C^* definované vztahem 4.5 na straně 21 je ten, že cena cíle C^{t} je u nevyplněných pozic nulová a stejně tak i cena řetězení C^{c} v místech, kde nebyla dosud vyhodnocena. Protože se při hledání koster v první fázi algoritmu ZCCFRAME nevyhodnocují žádné ceny řetězení, a to ani mezi přímo navazujícími ZCC řetězci, odpovídá hodnota funkce CF^* prostému součtu cen cíle C^{t} všech jednotek na vyplněných pozicích, tj. jednotek uvnitř použitých ZCC řetězců.

Výstup hodnoticí funkce H není jedno konkrétní číslo, ale vektor složený ze tří prvků. K určení nejslibnější kostry se provádí seřazení jejich množiny tak, že se nejprve porovnají hodnoty první složky hodnoticích vektorů a při jejich rovnosti se rozhoduje porovnáním druhé, případně i třetí složky.

Dalším rozdílem proti algoritmu BEFS je to, že se nehledá jedna vítězná cesta, ale všechny možné (neúplné) cesty – kostry, protože dokud nejsou doplněni kandidáti do všech prázdných pozic, nelze rozhodnout, která z koster bude mít nakonec nejlepší kumulativní cenu C.

[13)] Název funkce E_u vychází z angl. *empty units*.

[14)] Název funkce E_p vychází z angl. *empty parts*.



Obrázek 8.15: Obecné schéma kostry používané v algoritmu ZCCFRAME. Vyplněné jsou pozice pokryté řetězci ZCC4 a ZCC5. Samostatné nevyplněné pozice jsou značeny prefixem EU (celkem 3) a souvislé úseky prázdných pozic prefixem EP (celkem 2).

Na začátku hledání se v grafu ZCC řetězců vyhledají uzly, které nemají žádného předchůdce a zároveň je jejich vzdálenost od počátku původního grafu menší nebo rovna hodnotě parametru $START^{\max}$. Z nich se vytvoří počáteční množina koster, v níž každá obsahuje pouze jeden ZCC řetězec. Kostry se ohodnotí funkcí CF^* a provede se expanze té nejslibnější. Rozvoj kostry je řešen tak, že se v grafu ZCC řetězců vyhledají navazující uzly, pro každý z nich se vytvoří kopie právě expandované kostry a do ní se vloží odpovídající ZCC řetězec. Původně rozvíjená kostra se pak odstraní a dále se s ní již nepracuje. Tento cyklus rozvoje nejslibnější kostry se opakuje až do okamžiku, kdy se nová kostra vzniklá expanzí přiblíží konci původního grafu kandidátů na maximální vzdálenost určenou parametrem $END^{\max}$. Pokud kostra tuto podmínku splní, stává se z ní *kandidátní kostra* na vítěznou cestu grafem.

OMEZENÍ HLOUBKY EXPANZE A POČTU KOSTER

U algoritmu BEFS jsou cesty grafem uchovávány pomocí odkazu na nejlepšího zjištěného předchůdce daného uzlu a po dokončení hledání se vítězná cesta rekonstruuje zpětným řetězením. V případě algoritmu ZCCFRAME tento způsob není možné použít, protože cílem prohledávání v první fázi je nalézt všechny možné kostry (v daný okamžik nejslibnější), jejichž nevyplněné pozice budou doplněny až později. Kostry také slouží k průběžnému ukládání dopočítaných prázdných úseků a každá kostra má u sebe uloženo

i aktuální hodnocení funkcí CF^*, které lze při jakékoliv změně jednoduše a rychle přepočítat. Pokud by se používaly odkazy na nejlepší předchůdce, vedlo by dopočítání úseku na začátku kostry na složitou aktualizaci všech následujících uzlů resp. koster, u nichž byly prázdné pozice zaplněny. Nevýhodou tohoto přístupu uchovávání koster jsou vyšší paměťové nároky. Aby nedošlo v tomto směru k problémům, je hloubka expanze koster omezena parametrem *CHAINDEEP*. Při rozvoji nejslibnější kostry se použije pouze určený počet z množiny přípustných navazujících uzlů, které jsou seřazeny postupně podle vzdálenosti od kostry, délky příslušného ZCC řetězce a až posledním kritériem je součet cen cíle C^{t} kandidátů v ZCC řetězci. Tímto způsobem lze významně omezit exponenciální nárůst počtu vytvořených koster, nicméně i tak docházelo při experimentech k problémům s nedostatkem paměti. Proto byl algoritmus doplněn o další parametr $FRAMECOUNT^{\max}$, kterým je omezován celkový počet koster. Používá se tak, že po expanzi každé kostry se v paměti ponechá pouze $FRAMECOUNT^{\max}$ nejlepších koster ve smyslu hodnoticí funkce H.

Podobný problém vyvstává s velkým množstvím koster, jež dosáhnou specifikované vzdálenosti od konce grafu a přesouvají se do množiny kandidátních koster. Velikost této množiny byla také omezena, a to pomocí parametru $CANDCOUNT^{\max}$. V okamžiku nalezení stanoveného počtu kandidátních koster se fáze hledání koster ukončuje a algoritmus pokračuje druhou fází postupného doplňování jejich prázdných pozic.

8.2.5.3 Fáze doplnění prázdných pozic v kostrách

V rámci druhé fáze algoritmu ZCCFRAME se v množině kandidátních koster postupně doplňují chybějící úseky mezi ZCC řetězci. Proces probíhá v cyklu, jehož základem je opět algoritmus BEFS, tj. dochází ke zpracování vždy nejslibnějšího prvku, v tomto případě kostry. Hodnoticí funkce je v této fázi shodná s funkcí H definovanou vztahem 8.3 a pomocí ní se vždy vybere kostra s nejlepším hodnocením, ovšem místo její expanze se u ní provede doplnění jejího prvního dosud prázdného úseku. Dojde tak k vyplnění mezery mezi dvěma ZCC řetězci kostry cestou, jež je tvořena vhodně zvolenými kandidáty. Současně s tím se dopočítají i všechny ceny řetězení C^{c} jak mezi doplňovanými kandidáty, tak i mezi krajními kandidáty nové cesty, jež sousedí s příslušnými ZCC řetězci. Pokud na sebe ZCC řetězce přímo navazují, tj. vzdálenost mezi

nimi má hodnotu nula, pak se pouze vyhodnotí cena řetězení mezi krajními kandidáty. Po vyplnění prvního prázdného úseku nejslibnější kostry se znovu přepočte její hodnocení, množina kandidátních koster se opět seřadí dle hodnocení funkcí H a celý cyklus se opakuje, dokud nejsou všechny kostry vyplněné.

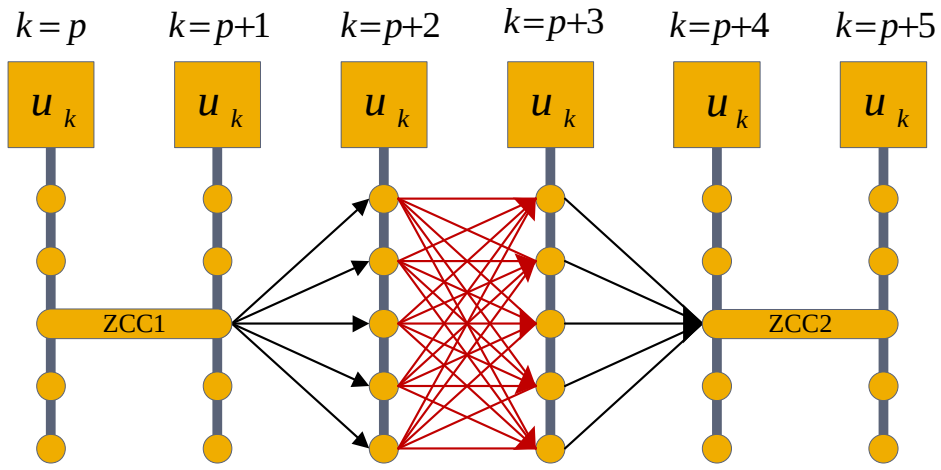
V okamžiku, kdy se v libovolné kandidátní kostře vyplní poslední prázdný úsek, přesune se příslušná kostra do množiny *úplných cest* a třetí složka její hodnoticí funkce H pak odpovídá kumulativní ceně celé cesty grafem. Závěrečným krokem druhé fáze algoritmu je prohledání množiny vytvořených úplných cest grafem a nalezení cesty s nejnižší kumulativní cenou C, která se pak stává vítěznou cestou, a v ní obsažené realizace jednotek se použijí k vygenerování výsledné promluvy.

OPTIMALIZACE PŘI DOPLŇOVÁNÍ PRÁZDNÝCH ÚSEKŮ KOSTER
Do hlavního cyklu doplňování dosud neobsazených pozic v kostrách byla doplněna obdobná optimalizace, která byla použita u optimalizovaného Viterbiova algoritmu VITOPT (viz oddíl 8.1.2). Předtím než dojde k dopočítání prvního prázdného úseku nejslibnější kandidátní kostry, testuje se, zda její částečná kumulativní cena (třetí prvek hodnoticí funkce) je nižší než kumulativní cena dosud nejlepší úplné cesty. Pokud tomu tak není, pak se kostra odstraní a dále se s ní již nepracuje, protože se již nemůže stát vítěznou cestou.

POSTUP HLEDÁNÍ KANDIDÁTŮ DO PRÁZDNÝCH POZIC KOSTER
Na výběr vhodných jednotek k doplnění koster lze nahlížet jako na hledání optimální cesty zúženým (pod)grafem, který vznikne vyjmutím odpovídajících kandidátů a jejich vzájemných spojení z původního grafu pro celou promluvu. Podgraf má na první pozici pouze koncový uzel prvního ZCC řetězce, pak pokračuje všemi uzly kandidátů na nevyplněných pozicích a je ukončen prvním uzlem navazujícího druhého ZCC řetězce.

Takto omezený podgraf lze pak prohledat libovolnou metodou, jako kdyby se jednalo o celou promluvu, ovšem s ohledem na strukturu kandidátů (resp. uzlů) v něm obsažených, je vhodnější použít algoritmus BEFS. Zásadní důvod, proč se vyhnout Viterbiovu algoritmu nebo jeho modifikacím (např. VITPRUNE), je ten, že pokud je dopočítávaná mezera mezi ZCC řetězci dlouhá dvě nebo více jednotek, pak se na vnitřních pozicích vždy vyhodnotí ceny řetězení mezi všemi kandidáty stejně jako u původního úplného grafu

(viz obrázek 8.16). To znamená, že pokud by při doplňování koster existovalo větší množství mezer delších než dvě jednotky, resp. pozice, pak by mohlo dojít k vyhodnocení stejného počtu cen řetězení C^c, jako kdyby se celý graf rovnou prohledal některou z variant Viterbiova algoritmu. Nejhorší možný scénář je dokonce takový, že pokud by se některé mezery v různých kostrách opakovaly, mohlo by množství výpočtů C^c být dokonce ještě vyšší než počet všech hran celého původního grafu.



Obrázek 8.16: Znázornění problému při použití Viterbiova algoritmu při hledání cest mezi řetězci ZCC1 a ZCC2. Při vzdálenosti dvou a více pozic je nutno vypočítat ceny řetězení mezi všemi kandidáty na vnitřních pozicích (znázorněno červenými spojnicemi).

Při experimentech s algoritmem ZCCFRAME byl tedy používán algoritmus BEFS, který používá strategii rozvoje nejslibnější cesty (s omezenou hloubkou expanze) a je s ohledem na výše uvedený problém vhodnější. Struktura prohledávaných podgrafů také nahrává tomuto algoritmu, neboť na jeho počáteční pozici je vždy pouze jeden jediný kandidát, a díky tomu dojde i k rychlejšímu nárůstu rozdílů ohodnocení (cen) rozvíjených cest.

Výjimkou jsou pouze případy, kdy se dopočítává úsek vedoucí z počátku původního grafu k prvnímu ZCC řetězci kostry (tj. pokud řetězec nezačíná také na první pozici grafu). Tento problém byl po řadě experimentů vyřešen tak, že se vhodné jednotky prohledávají v revertovaném podgrafu, čímž se minimalizuje problém s nutností vyhodnocení velkého množství kandidátů na první pozici s velmi podobnými cenami cíle (viz také komentář ke strategii

Viterbiova algoritmu na straně 68). Technika hledání cesty grafem v opačném směru byla již zmíněna v předchozím textu na straně 70 v oddíle 8.1.5, kde byl popsán i experiment s prohledáváním takto upraveného grafu Viterbiovým algoritmem, ale dosažené výsledky nevedly ani k urychlení a paradoxně u prořezávaných variant hledání došlo navíc ke zhoršení matematické kvality Q. Při hledání optimálních cest od počátku grafu k uzlu na začátku prvního ZCC řetězce se naopak kombinace algoritmu BEFS a revertovaného podgrafu ukázala být efektivnější než při hledání v dopředném směru. Vyšší rychlost hledání byla ověřena pomocí simulace, při níž se pro všechny promluvy testovací množiny vybralo z grafu kandidátů vždy 30 uzlů s nejnižší cenou cíle C^{t} na pozici č. 3, byl vytvořen odpovídající podgraf od první pozice až k vybranému uzlu a algoritmem BEFS byla nalezena nejlepší cesta. Poté se celý test opakoval s tím rozdílem, že podgrafy byly revertované. Následným porovnáním množství výpočtů cen řetězení C^{c} bylo zjištěno, že při hledání v opačném směru bylo potřeba 148,18 krát méně výpočtů cen řetězení (při nastavení parametrů $DEEP = \infty$ a $\alpha = 0$). Zde je ovšem nutno poznamenat, že jde sice o významnou míru urychlení, ale pouze ve specifických případech, když první ZCC řetězec kostry nezačíná na první pozici. Technika hledání počátečních cest koster v revertovaném grafu tedy přispěje k celkovému urychlení, nicméně ne tak významně jako při uvedené simulaci.

MEZIPAMĚŤ CEST MEZI ZCC ŘETĚZCI

Vzhledem ke způsobu vytváření kandidátních koster v první fázi algoritmu, dochází k tomu, že se některé jejich části shodují a je nutné opakovaně vyhledat cestu mezi stejnými ZCC řetězci. Z tohoto důvodu byl algoritmus ZCCFRAME doplněn o optimalizaci, která všechny dosud nalezené cesty k vyplnění prázdných úseků koster ukládá do mezipaměti. Při opakovaném požadavku na vyhledání cesty stejným úsekem se neprovádí prohledávání subgrafu algoritmem BEFS, ale použije se výsledek uložený v mezipaměti. Mezipaměť se nijak neuchovává a je platná pouze po dobu syntézy jedné promluvy. Není tak potřeba řešit úsporný způsob uložení nebo výkonnostní problémy spojené s obrovským množstvím záznamů.

MEZIPAMĚŤ CEN ŘETĚZENÍ

Při hledání pomocí Viterbiova algoritmu (nebo jeho optimalizovaných variant), stejně tak u algoritmu BEFS, se nikdy neopakovaly výpočty cen řetězení mezi

stejnými kandidáty. U algoritmu ZCCFRAME to již neplatí, a kromě možného opakování hledání cest mezi ZCC řetězci, může dojít i k opakovanému vyhodnocení cen řetězení mezi jednotlivými kandidáty. Další úspory výpočetního výkonu bylo dosaženo použitím cache hodnot cen řetězení C^{c} mezi kandidáty jednotek promluvy. Nejedná se o mezipaměť všech možných nebo nejčastějších kombinací z celé databáze řečových segmentů, jak je popisováno v oddíle 6.1.2, ale opět jde o cache, která je platná pouze v rámci syntézy jedné promluvy.

Nejprve byla zvolena cache, která formou hash tabulky indexovala globálně všechny výpočty, kde klíčem byly identifikátory kandidátů v grafu. Tato cache sice nečítala milióny záznamů, jak je uváděno v např. publikacích Beutnagel *et al.*, 1999 nebo Čepko *et al.*, 2008, ale i tak obsahovala až desítky tisíc kombinací a začínaly se projevovat problémy s rychlostí vyhledávání. Tento problém byl vyřešen tak, že se mezipaměť ukládala lokálně ke každému kandidátovi a byly zde uchovány již vyhodnocené ceny řetězení s předcházejícími kandidáty grafu. Tím se sice nijak neuspořila ani paměť, ani počet uchovaných hodnot, nicméně došlo k urychlení vyhledání cen, protože se nejprve velmi efektivně nalezne lokální cache spojená s kandidátem a poté se hodnota v hash tabulce hledá pouze v omezeném počtu možných spojení v počtu maximálně stovek hodnot.

8.2.5.4 Implementace algoritmu ZCCFRAME

Předpis hledání algoritmem ZCCFRAME je následující:

> I. FÁZE – HLEDÁNÍ KANDIDÁTNÍCH KOSTER
> V této části předpisu se pracuje se dvěma množinami PARTS a CANDS. Množina PARTS obsahuje částečné kostry, které jsou postupně rozvíjeny a doplňovány. V množině CANDS jsou uloženy všechny kandidátní kostry, které již dosáhly konce grafu.
>
> 1. Ke všem uzlům grafu kandidátů se vypočítá cena cíle C^{t}.
> 2. V grafu kandidátů se vyhledají všechny ZCC řetězce s minimální délkou $LEN^{\min}$.
> 3. Ke každému ZCC řetězci se doplní seznam odkazů na navazující ZCC

řetězce, jež začínají na vyšší pozici, přičemž musí být splněna podmínka na maximální povolenou vzdálenost $TRANS^{\max}$. Vytvoří se tak graf ZCC řetězců, v němž uzly tvoří nalezené ZCC řetězce a hrany všechna přípustná spojení.

4. Každý ZCC řetězec, začínající do vzdálenosti $START^{\max}$ od počátku grafu kandidátů, se vloží do nově vytvořené kostry a ta se zařadí do množiny PARTS.

5. Z množiny PARTS se vybere kostra s nejlepším hodnocením funkcí H a označí se f^*.

6. Pokud poslední řetězec kostry f^* končí maximálně ve vzdálenosti $END^{\max}$ od konce grafu kandidátů, pak se kostra f^* přesouvá do množiny CANDS.

7. Pokud množina CANDS obsahuje již $CANDCOUNT^{\max}$ kandidátních koster, pak první fáze končí a algoritmus pokračuje bodem 11.

8. Pokud kostra f^* nedosahuje limitu na vzdálenost ke konci grafu, pak se provede její expanze:

   (a) Množina navazujících ZCC řetězců posledního ZCC řetězce kostry f^* se seřadí podle vzdálenosti (od nejbližších ZCC řetězců), podle délky (od delších) a podle součtu cen cíle C^{t} kandidátů ZCC řetězce (od nejnižší hodnoty).

   (b) Ke každému z prvních $CHAINDEEP$ navazujících ZCC řetězců se vytvoří kopie kostry f^*, do ní se doplní příslušný ZCC řetězec a nově vytvořená kostra se zařadí do množiny PARTS.

   (c) Kostra f^* se odstraní z množiny PARTS a dále se s ní již nepracuje.

9. Kostry v množině PARTS se seřadí podle ohodnocení funkcí H a v množině se ponechá pouze $FRAMECOUNT^{\max}$ nejlepších koster.

10. Algoritmus se vrací zpět na bod 5, dokud není množina PARTS prázdná.

II. FÁZE – VYPLŇOVÁNÍ PRÁZDNÝCH ÚSEKŮ KOSTER
Tato část algoritmu vychází z množiny CANDS a zavádí novou množinu FINAL, která slouží k uchování úplných cest.

11. Z množiny CANDS se vybere kandidátní kostra s nejlepším ohodnocením funkcí H a označí se f^*.

12. Pokud je hodnota částečné kumulativní ceny CF^* nejlepší kandidátní kostry f^* větší nebo rovna nejnižší kumulativní ceně úplné cesty z množiny FINAL, pak hlavní cyklus končí a algoritmus přejde na bod 16.

13. V kandidátní kostře f^* se vyhledá první nevyplněný úsek a provede se pro něj:

    (a) Pokud se jedná o úsek mezi dvěma ZCC řetězci, které na sebe přímo navazují, pak se v kostře dopočítá cena řetězení mezi koncovými kandidáty ZCC řetězců.

    (b) Pokud se jedná o úsek mezi dvěma ZCC řetězci vzdálených od sebe více než jednu pozici, pak se vytvoří podgraf z původního grafu kandidátů složený z koncového kandidáta prvního řetězce, všech jednotek na pozicích mezi ZCC řetězci a ukončený počáteční jednotkou druhého uzlu. Tento podgraf se prohledá algoritmem BEFS a vítězná cesta se použije k vyplnění prázdného úseku kandidátní kostry f^*.

    (c) Pokud se jedná o úsek od počátku grafu kandidátů k prvnímu ZCC řetězci kandidátní kostry f^*, pak se vytvoří podgraf z kandidátů na pozicích předcházející ZCC řetězci doplněný o počátečního kandidáta ZCC řetězce. Provede se inverze podgrafu a jeho prohledání algoritmem BEFS. Vítězná cesta se použije k vyplnění prázdného úseku.

    (d) Pokud se jedná o úsek od posledního ZCC řetězce ke konci grafu kandidátů, pak se provedou analogické kroky jako v bodě 13c, pouze s tím rozdílem, že podgraf není invertován.

14. Pokud je kandidátní kostra f^* zcela bez prázdných úseků, pak se přesune do množiny FINAL.

15. Pokud není množina CANDS prázdná, pak se algoritmus vrací zpět na bod 11.

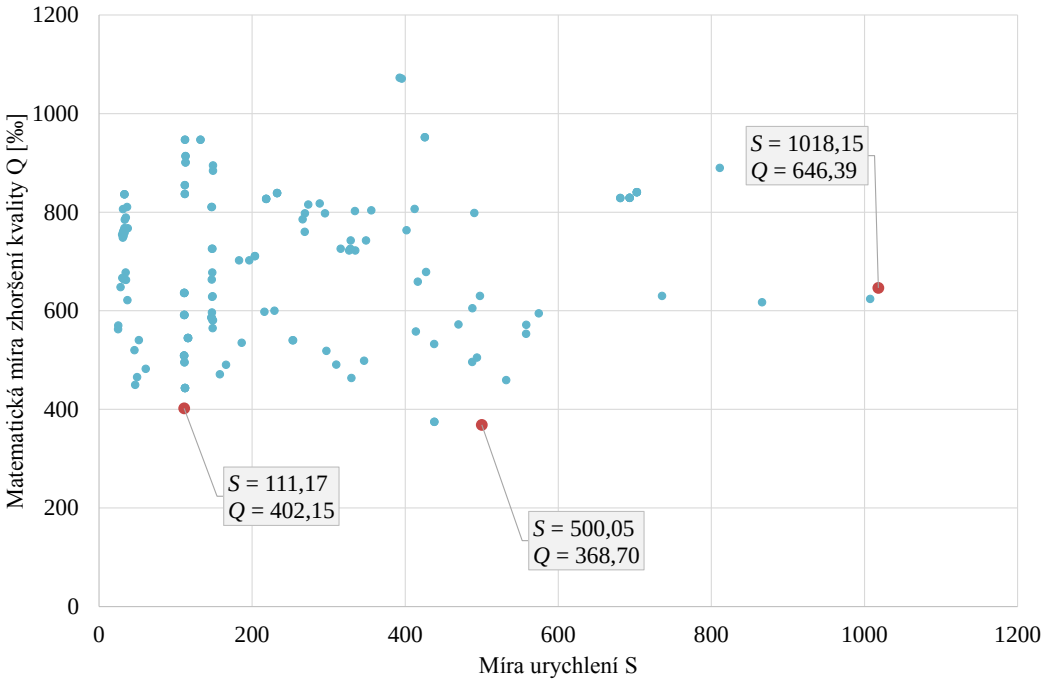
16. Z množiny FINAL se vybere úplná cesta s nejnižším hodnocením a stává se vítěznou cestou grafem kandidátů, jež se použije k syntéze výsledné promluvy.

8.2.5.5 Hodnocení algoritmu ZCCFRAME

Z hlediska výkonu byly výsledky získané užitím algoritmu ZCCFRAME rozporuplné. Sice se podařilo snížit potřebné množství výpočtů cen řetězení C^{c} řádově o stovky, a tím významně urychlit proces výběru jednotek, bohužel však míra zhoršení matematické kvality Q vykazovala velmi vysoké hodnoty u všech testovaných konfigurací (rozsah hodnot parametrů viz tabulka B.5 v příloze B). Minimální hodnoty ukazatele $Q = 368{,}70$ ‰ bylo dosaženo při zrychlení $S = 500{,}05$ (viz tabulka 8.7) a i u této varianty bylo zhoršení vygenerovaných promluv při poslechu patrné. Podobně jako u algoritmu ZCCBEFS dosáhla míra fragmentace s hodnotou $CD = 28{,}65$ % lepšího výsledku než u referenčního algoritmu VITBASE. I v tomto případě není toto zlepšení překvapením, protože algoritmus ZCCFRAME používá ZCC řetězce jako základ cest prohledávaným grafem kandidátů.

Tabulka 8.7: Hodnocení výkonu základní verze algoritmu ZCCFRAME – konfigurace dosahující nejlepšího výsledku matematické kvality

Vlastnost	Hodnota
Kód algoritmu	ZCCFRAME
Míra urychlení (S)	500,05
Matematická míra zhoršení kvality (Q)	368,70 ‰
Míra fragmentace (CD)	28,65 %
Hodnoty parametrů: $LEN^{\min} = 3$, $TRANS^{\max} = 4$, $START^{\max} = 3$, $END^{\max} = 3$, $CHAINDEEP = 10$, $FRAMECOUNT^{\max} = 2000$, $CANDCOUNT^{\max} = 1000$	



Obrázek 8.17: Matematická míra zhoršení kvality ve vztahu ke zrychlení pro různé parametry algoritmu ZCCFRAME.

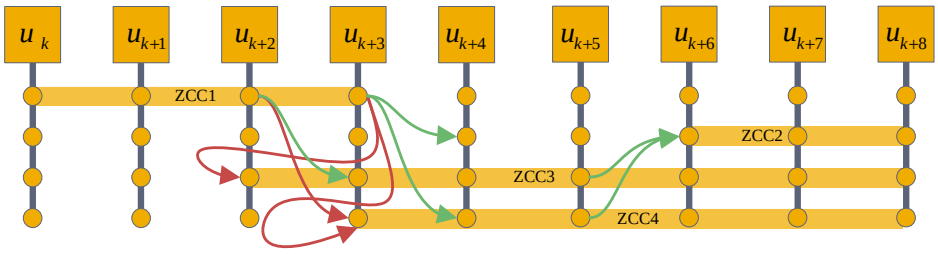
8.2.6 Analýza vlastností ZCC řetězců v optimální cestě

Výkon dosud testovaných algoritmů, zejména algoritmu ZCCFRAME, byl stále nedostatečný a míra urychlení nedosahovala hodnot, kterými by se dala opodstatnit zvýšená složitost způsobu hledání optimální cesty grafem. Větším problémem však byl fakt, že ani za cenu nastavení parametrů vedoucích na velmi vysoké výpočetní i paměťové nároky se nepodařilo získat cesty grafem kandidátů s globálně minimální kumulativní cenou C^* nebo se k této hodnotě přiblížit na přijatelnou úroveň. Při hledání příčin těchto problémů bylo pozorováno, že se velmi často ve vítězné cestě nalezené algoritmem ZCCFRAME nepoužily některé dlouhé ZCC řetězce.

Z tohoto důvodu byla provedena řada srovnání složení cest s globálně minimální kumulativní cenou s výsledky získanými algoritmem ZCCFRAME. Bylo zjištěno, že často se z velmi dlouhých ZCC řetězců použily v cestě s globálně minimální kumulativní cenou pouze jejich části – jejich podřetězce. Pokud se totiž optimální cesta vyhledává např. základním Viterbiovým algoritmem (jenž zaručuje nalezení cesty s globálně minimální kumulativní cenou), pak se pracuje s jednotlivými kandidáty. Ke vzniku řetězců s nulovou cenou řetězení

pak dochází přirozeně díky zvolenému způsobu hodnocení cest a strategii minimalizující kumulativní cenu. V případě algoritmu ZCCFRAME se však základem optimální cesty stává kombinace ZCC řetězců, které jsou nejprve vyhledány jako celky v grafu kandidátů bez ohledu na okolní jednotky nebo budoucí možné cesty grafem, a to způsobuje problémy ve vztahu k minimalizaci kumulativní ceny.

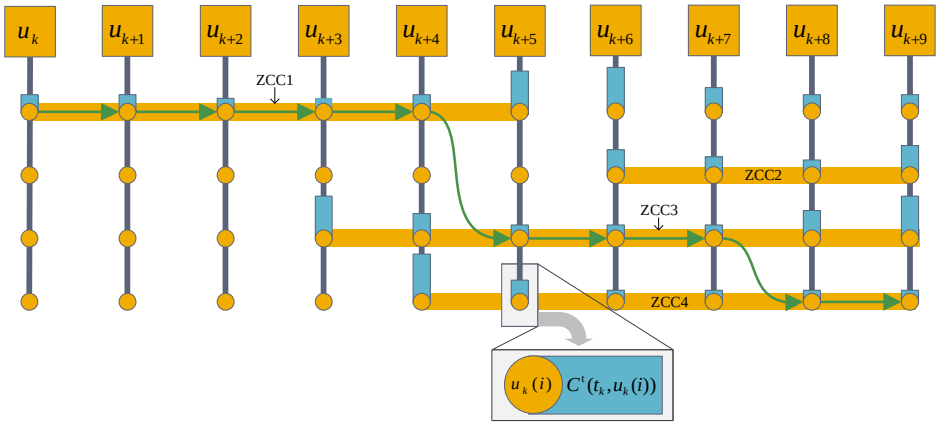
Prvním typem problémů jsou případy, kdy by dva ZCC řetězce svým spojením mohly pokrýt významnou část promluvy, ale nelze je na sebe napojit, protože koncový uzel prvního z nich je např. na stejné pozici jako první uzel druhého ZCC řetězce (viz obrázek 8.18). Při výběru optimální sekvence kandidátů po jednotkách by vznikly stejné ZCC řetězce, ale jeden z nich by byl o problematickou jednotku kratší, a tím by bylo umožněno je na sebe napojit. V některých případech je z pohledu dosažení minimální kumulativní ceny cesty dokonce výhodnější zkrátit ZCC řetězec o několik jednotek, případně zkrátit oba dva, a úsek mezi nimi vyplnit zcela samostatnými kandidáty, kteří nejsou součástí žádného řetězce, ale jejich cena cíle a ceny řetězení mají v součtu nižší hodnotu. Nejčastějším pozorovaným případem bylo vyplnění jednou samostatnou jednotkou v situacích, kdy by přímé napojení krajních kandidátů zkrácených ZCC řetězců bylo ohodnoceno velmi vysokou cenou řetězení C^{c}.



Obrázek 8.18: Znázornění problému s návazností dlouhých ZCC řetězců. Příkladem jsou ZCC řetězce ZCC1 a ZCC3, jejichž kombinací lze pokrýt všechny pozice promluvy, ale jejich napojení by bylo možné pouze tehdy, pokud by byl alespoň jeden z nich kratší minimálně o jednu jednotku (možné spoje jsou naznačeny zelenými spojnicemi). V původní délce je totiž nelze na sebe napojit (reprezentováno červenou spojnicí).

Druhý, často se vyskytující problém vzniká tehdy, když se ZCC řetězec vyskytne na ne zcela vhodné pozici v grafu z pohledu specifikace cíle. Kandidáti, kteří jej tvoří, pak mají přiřazené vyšší hodnoty ceny cíle C^{t}, přičemž se nejčastěji jedná o kandidáty na okrajích ZCC řetězce (viz obrázek 8.19). Zde mají opět výhodu algoritmy, u nichž probíhá prohledávání grafu po jednot-

kách, protože se na danou pozici mohou dostat kandidáti mimo ZCC řetězec s nižšími hodnotami ceny cíle C^{t}, pokud je výsledná kumulativní cena cesty C nižší (i přes případné zvýšené hodnoty cen řetězení C^{c}).

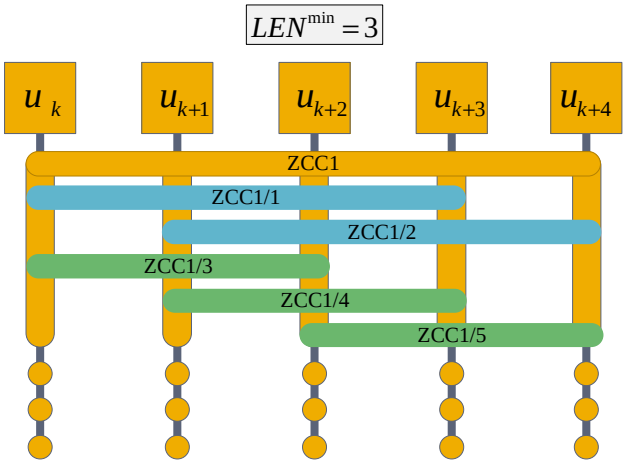


Obrázek 8.19: Znázornění problému s vyššími hodnotami ceny cíle C^{t} u kandidátů na okrajích delších ZCC řetězců. Vítězná cesta grafem (zelené spojnice) pak typicky prochází pouze částmi původních ZCC řetězců a vyhýbá se tak případným vysokým hodnotám C^{t} (ve schématu znázorňuje výška modrých obdélníků výši cen cíle).

8.2.7 Modifikace algoritmu ZCCFRAME doplněním ZCC podřetězců

Poznatky získané analýzou uvedenou v předchozím oddíle vedly k úvaze jak zvýšit matematickou kvalitu algoritmu ZCCFRAME při zachování základního principu využití ZCC řetězců. Řešení, které bylo použito, spočívalo v zapracování mechanismu, který po vyhledání ZCC řetězců doplní jejich množinu o všechny jejich přípustné podřetězce splňujících kritérium na definovanou minimální délku. Tato verze algoritmu bude značena kódem ZCCFRAME$^{\mathrm{sub}}$ (z angl. *subchains*). Z každého řetězce se vytvoří dvě kopie, přičemž první se zkrátí o počátečního kandidáta a druhá o koncového. Tento proces se pro všechny podřetězce rekurzivně opakuje, dokud délka zkrácených kopií nedosáhne hodnoty parametru $LEN^{\min}$ (viz schéma na obrázku 8.20).

Se všemi nově vytvořenými podřetězci se při prohledávání grafu (resp. při vytváření koster) pracuje naprosto stejným způsobem jako s původními ZCC řetězci. Rozdíl je pouze v tom, že počet ZCC řetězců je mnohem vyšší a stejně tak i počet možných kombinací, jak z nich složit kandidátní kostry. Díky této úpravě algoritmu ZCCFRAME byla předpokládána vyšší pravděpodobnost,



Obrázek 8.20: Schéma tvorby ZCC podřetězců. Na obrázku je znázorněno vytvoření kopií podřetězců z původního řetězce ZCC1 o délce 5 jednotek, a to při nastavení parametru $LEN^{\min} = 3$. Podřetězce délky 4 jsou vykresleny modrou barvou a podřetězce délky 3 barvou zelenou.

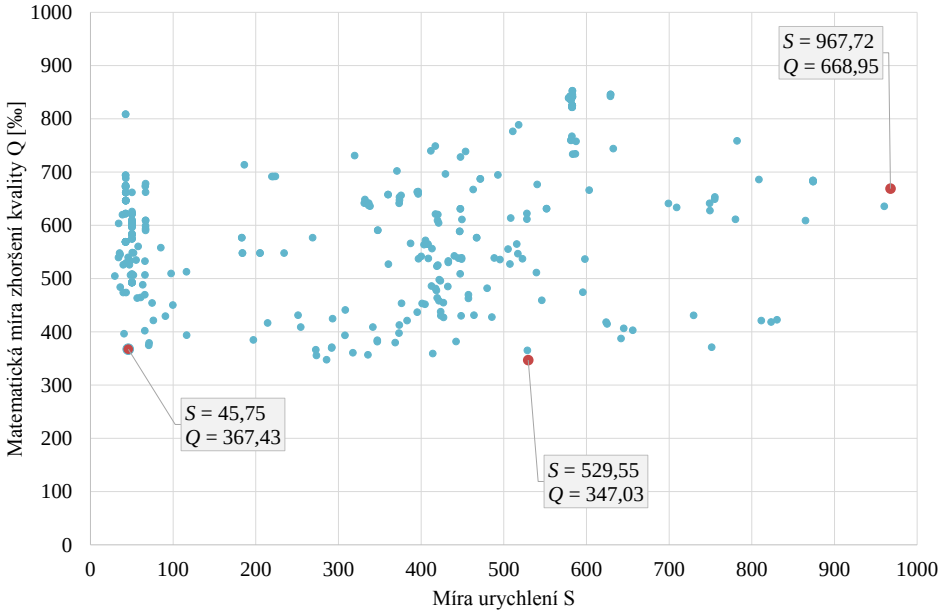
že dojde k výběru cesty s globálně minimální kumulativní cenou C, případně že se výsledky této ceně alespoň přiblíží.

HODNOCENÍ VÝKONU PO DOPLNĚNÍ ZCC PODŘETĚZCŮ

Výkonnost algoritmu ZCCFRAME$^{\text{sub}}$, jenž zahrnuje mechanismus doplňování podřetězců, je reprezentována grafem na obrázku 8.21 (rozsah hodnot parametrů viz tabulka B.5 v příloze B). U nejlepší varianty (viz tabulka 8.8) byla rychlost $S = 529{,}55$ srovnatelná s algoritmem ZCCFRAME bez podřetězců ($S = 500{,}05$). Došlo sice dle očekávání ke zlepšení ukazatele matematické kvality na hodnotu $Q = 347{,}03$ ‰ (bez podřetězců $Q = 368{,}70$ ‰), ovšem míra zlepšení byla předpokládána vyšší a zhoršení kvality ve srovnání s referenčním algoritmem VITBASE bylo i v tomto případě rozpoznatelné jednoduchým poslechovým testem. V důsledku výrazného zvýšení počtu ZCC řetězců, se tak u algoritmu ZCCFRAME$^{\text{sub}}$ zlepšila šance na výběr posloupností s vyšší kvalitou, ale patrně se naopak ještě více projevila negativní vlastnost algoritmu ve sklonu k upřednostnění lokálních minim z pohledu kumulativní ceny C.

Při snaze docílit ještě vyšší kvality byly provedeny i experimenty s nastavením minimální délky ZCC řetězců na dvě jednotky ($LEN^{\min} = 2$), ale u těchto variant konfigurace bylo ZCC řetězců již takové množství, že buď výpočetní a paměťové nároky neumožňovaly úspěšně dokončit výběr jednotek, nebo bylo nutné omezit počet koster parametrem $FRAMECOUNT^{\max}$ tak,

že kvalita byla opět matematicky i na první poslech nedostatečná.



Obrázek 8.21: Matematická míra zhoršení kvality ve vztahu ke zrychlení pro různé parametry algoritmu ZCCFRAME$^{\text{sub}}$ s doplněnými podřetězci.

Tabulka 8.8: Hodnocení výkonu základního algoritmu ZCCFRAME$^{\text{sub}}$ s doplněnými podřetězci – konfigurace dosahující nejlepšího výsledku matematické kvality

Vlastnost	Hodnota
Kód algoritmu	ZCCFRAME$^{\text{sub}}$
Míra urychlení (S)	529,55
Matematická míra zhoršení kvality (Q)	347,03 ‰
Míra fragmentace (CD)	28,82 %
Hodnoty parametrů: $LEN^{\min} = 3$, $TRANS^{\max} = 2$, $START^{\max} = 3$, $END^{\max} = 3$, $CHAINDEEP = 10$, $FRAMECOUNT^{\max} = 1000$, $CANDCOUNT^{\max} = 1000$	

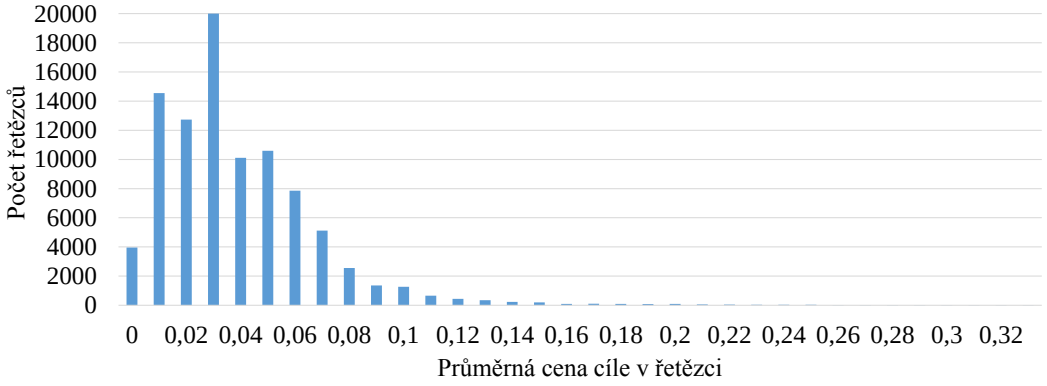
8.2.8 Prořezávání na základě analýzy vlastností vítězných ZCC řetězců

Mechanismus doplňování ZCC podřetězců sice pomohl k mírnému zvýšení matematické kvality Q u generovaných promluv, ale výsledky zaostávaly za očekávanými hodnotami matematických ukazatelů výkonu. Další z úvah

nad tím, jak algoritmus optimalizovat a zlepšit kvalitu generované řeči, byla zaměřena na vlastnosti množiny ZCC řetězců (tvořené původními řetězci i jejich podřetězci), které byly použity v cestě s globálně minimální kumulativní cenou u posloupností získaných pomocí referenčního Viterbiova algoritmu VITBASE.

Byla opět provedena analýza 10 000 promluv vygenerovaných pomocí referenčního algoritmu, podobně jako v oddíle 8.2.2 věnovanému analýze výskytu ZCC řetězců. V tomto případě byly zkoumány vlastnosti souvislých úseků promluv z řečového korpusu, jež by mohly sloužit k předvýběru ZCC řetězců, u nichž je vyšší pravděpodobnost, že budou součástí optimální cesty grafem kandidátů.

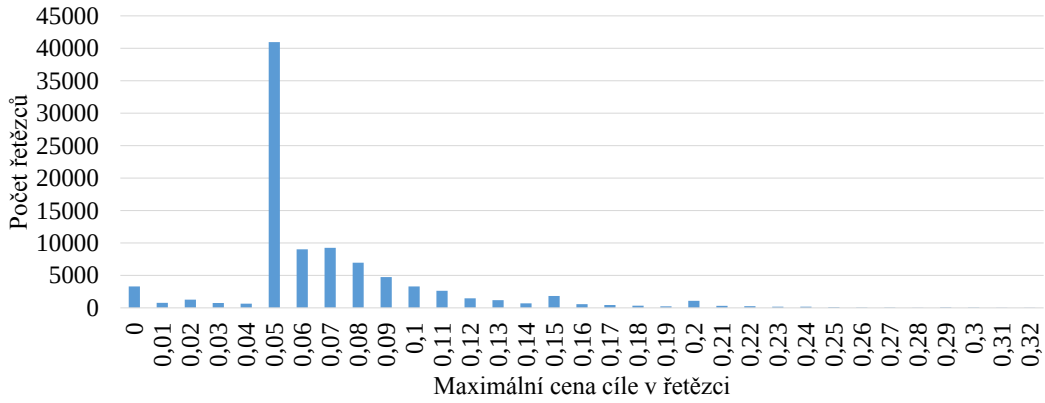
Jednou ze zkoumaných vlastností byla i cena cíle C^{t} u kandidátů v těchto úsecích. Bylo zjištěno, že průměrná hodnota C^{t} úseků delších než dvě jednotky je velmi nízká ve srovnání s množinou všech možných ZCC řetězců. Z grafu na obrázku 8.22 je patrné, že většina ZCC řetězců vítězných cest má průměrnou hodnotu menší nebo rovnu 0,1. Přesně se jednalo o 90 109 (97,14 %) z celkového počtu 92 762 ZCC řetězců. Za hraniční hodnotu lze považovat průměrnou cenu cíle o velikosti 0,3, protože počet ZCC řetězců s vyšší hodnotou byl zanedbatelný a činil pouhých 0,02 % ze všech ZCC řetězců.



Obrázek 8.22: Statistika počtu ZCC řetězců globálně minimálních cest v závislosti na průměrné ceně cíle C^{t} v nich obsažených kandidátů.

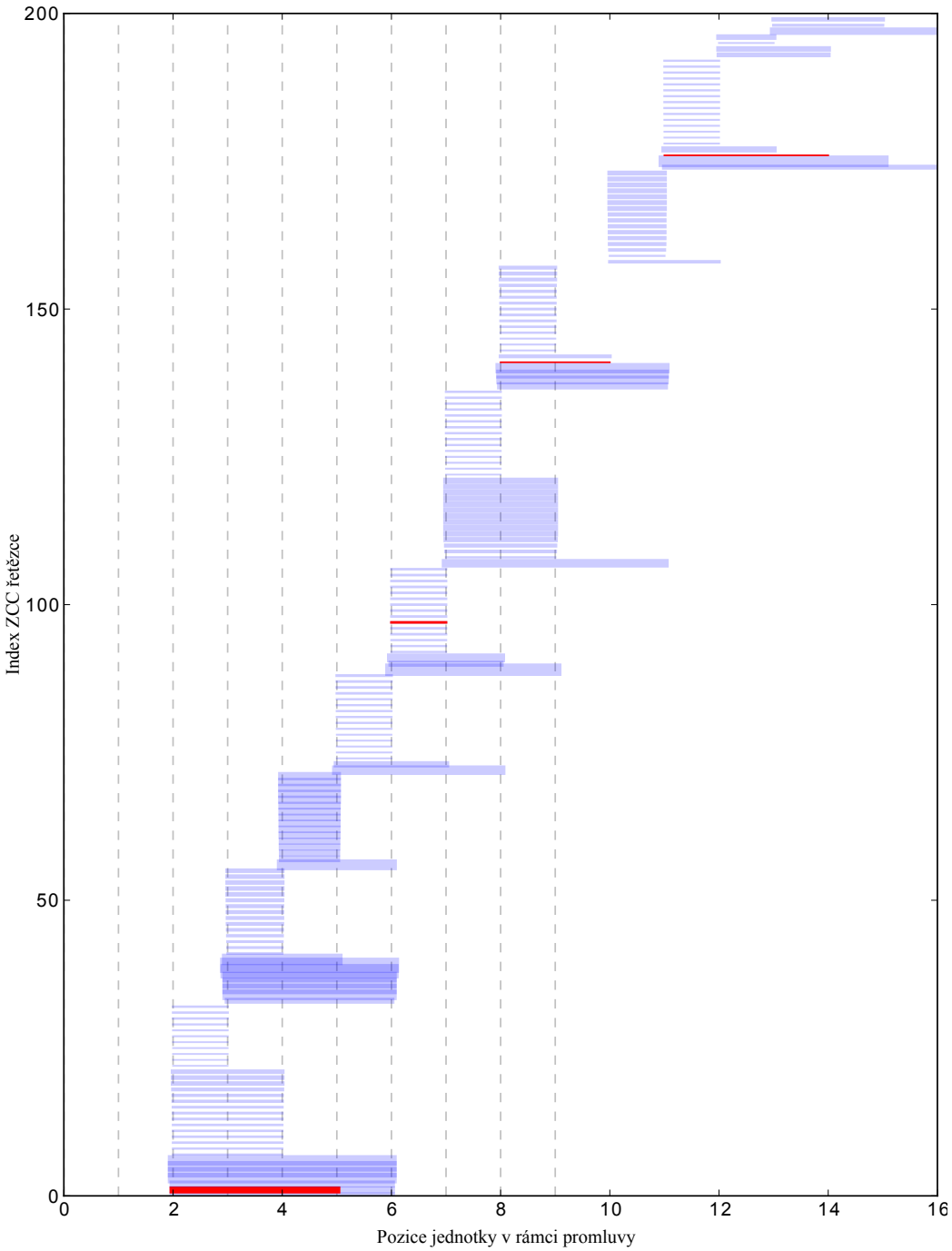
Další statistika, uvedená na obrázku 8.23, je zaměřena na maximální cenu cíle u kandidátů v ZCC řetězcích použitých v cestách s globálně minimální kumulativní cenou. Z ní vyplývá, že nejvíce byla zastoupena maximální hodnota $C^{\mathrm{t}} = 0{,}05$ (44,15 %). Počet ZCC řetězců s vyšším maximem C^{t} pak

rychle klesá a podobně jako u průměrné hodnoty je jejich množství za hranicí 0,33 zanedbatelné, a to pouhých 46 (0,05 %) výskytů.



Obrázek 8.23: Statistika počtu ZCC řetězců globálně minimálních cest v závislosti na maximální ceně cíle C^{t} v nich obsažených kandidátů.

Při zkoumání možností, jak co nejefektivněji dojít k výběru optimální cesty grafem kandidátů, byla vytvořena i řada vizualizací ZCC řetězců v promluvě s cílem posoudit možná kritéria pro jejich heuristický předvýběr. Jedna z forem prezentace byla vytvořena tak, že se ZCC řetězce seřadily podle jejich počáteční pozice a byly vykresleny v grafu, v němž osa x odpovídala pozici v grafu a osa y pořadí ZCC řetězce (viz obrázek 8.24). Aby vizualizace lépe odpovídala strategii algoritmu, tj. snaze co nejvíce vyplnit cestu grafem vhodnými ZCC řetězci, byly ZCC řetězce rozděleny do skupin podle jejich počáteční pozice. V rámci skupin byly ZCC řetězce seřazeny podle své délky (od nejdelších) a dále pak podle součtu ceny cíle C^{t} v nich obsažených kandidátů. V grafu byly ZCC řetězce reprezentovány úsečkami podle odpovídající počáteční a koncové pozice v grafu. Odlišnou barvou pak byly vykresleny úseky mezi kandidáty, jež byly součástí cesty s globálně minimální kumulativní cenou.



Obrázek 8.24: Příklad vizualizace ZCC řetězců v promluvě. Modrými úsečkami jsou vyznačeny ZCC řetězce, tj. pozice, které ZCC řetězce zaujímají. Červenou barvou jsou pak vyznačeny úsečky, jež jsou zároveň součástí globálně minimální cesty získané základním Viterbiovým algoritmem VITBASE. ZCC řetězce jsou seřazeny ve skupinách podle začátku resp. konce a dále pak od nejnižšího součtu cen cíle C^{t} v nich obsažených realizací jednotek. Z obrázku je patrné, že červené úsečky, resp. ZCC řetězce, jsou většinou umístěny ve spodní části skupin.

U takto vytvořených grafů nad množinou testovacích promluv bylo pozorováno, že ZCC řetězce složené z kandidátů globálně optimální sekvence se ve většině případů vykytovaly na začátku vytvořené skupiny, což vedlo k závěru, že skupiny ZCC řetězců začínajících na stejné pozici a majících shodnou délku lze omezit na určený maximální počet ZCC řetězců s nejnižší průměrnou cenou cíle, aniž by s vysokou mírou pravděpodobnosti došlo ke zhoršení matematické kvality u výsledných promluv.

Na základě uvedených statistik bylo do algoritmu ZCCFRAME, resp. jeho verze s doplněnými podřetězci ZCCFRAME$^{\text{sub}}$, zapracováno prořezávání množiny ZCC řetězců. Vznikla tak nová varianta algoritmu značená ZCCFRAME$^{\text{prn}}$ (z angl. *pruned*), v níž se ponechávají ke zpracování pouze ty řetězce, jež splňují následující kritéria:

- Průměrná hodnota ceny cíle C^{t} kandidátů v ZCC řetězci je menší nebo rovna hodnotě nově zavedeného parametru ${}_{zcc}C^{\text{t}}_{\text{avg}}$ [15)], tj. platí: $\frac{1}{N}\sum_{i=1}^{N} C^{\text{t}}(u_i) \leq {}_{zcc}C^{\text{t}}_{\text{avg}}$, kde N je počet kandidátů ZCC řetězce a u_i je kandidát s indexem i.

- Maximální hodnota ceny cíle C^{t} kandidátů ZCC řetězce je menší nebo rovna hodnotě nově zavedeného parametru ${}_{zcc}C^{\text{t}}_{\text{max}}$ [16)], tj. platí: $\forall i = \{1, \ldots, N\} : C^{\text{t}}(u_i) \leq {}_{zcc}C^{\text{t}}_{\text{max}}$, kde N je počet kandidátů ZCC řetězce a u_i je kandidát s indexem i.

- ZCC řetězce jsou rozděleny do skupin. V každé z nich se nacházejí ZCC řetězce se shodným počátkem i koncem v grafu kandidátů. Množství řetězců ve skupinách je pak omezeno pomocí nového parametru $GROUPCOUNT^{\max}$ na definovaný počet s nejnižší průměrnou hodnotou ceny cíle kandidátů, z nichž jsou řetězce složeny.

Je důležité poznamenat, že hodnoty použitých konstant závisí na konkrétní databázi řečových segmentů a také definici funkce ceny cíle C^{t}. Pokud by se použila databáze s odlišným počtem segmentů (např. provedením redukce) nebo databáze vytvořená jiným hlasem, pak by bylo nutné znovu provést

[15)] Pro testovanou databázi řečových segmentů mužského hlasu byla s ohledem na předchozí statistiku stanovena hodnota parametru na ${}_{zcc}C^{\text{t}}_{\text{avg}} = 0{,}3$ (pro ženský hlas ${}_{zcc}C^{\text{t}}_{\text{avg}} = 0{,}33$).

[16)] Pro testovanou databázi řečových segmentů mužského hlasu byla s ohledem na předchozí statistiku stanovena hodnota parametru na ${}_{zcc}C^{\text{t}}_{\text{max}} = 0{,}33$ (pro ženský hlas ${}_{zcc}C^{\text{t}}_{\text{max}} = 0{,}35$).

statistické měření na větším počtu promluv s globálně minimální kumulativní cenou a konstanty nastavit na odpovídající hodnoty.

HODNOCENÍ VÝKONU PO PROŘEZÁNÍ ZCC PODŘETĚZCŮ

Při použití varianty algoritmu ZCCFRAME$^{\text{prn}}$ (rozsah hodnot parametrů viz tabulka B.6 v příloze B), v níž byly doplněny podřetězce a celá množina ZCC řetězců byla následně prořezána, bylo dosaženo zlepšení zejména u hodnoty matematické kvality Q (viz obrázek 8.25). Nejnižší hodnota tohoto ukazatele činila $Q = 201{,}89$ ‰ a byla nižší než u základní verze algoritmu ZCCFRAME ($Q = 368{,}70$ ‰) i než verze s doplněnými ZCC podřetězci ZCCFRAME$^{\text{sub}}$ ($Q = 347{,}03$ ‰). Z grafu výkonnosti na obrázku 8.25 je patrný i celkový posun hodnot matematické kvality experimentů směrem k hranici $Q \approx 200$ ‰, přičemž např. u varianty ZCCFRAME$^{\text{sub}}$ začínala tato hranice přibližně u hodnoty $Q \approx 350$ ‰.

Z hlediska míry urychlení S dává algoritmus ZCCFRAME$^{\text{prn}}$ podobné výsledky jako u varianty ZCCFRAME$^{\text{sub}}$, nicméně i zde je z grafu na obrázku 8.25 zřejmý posun u většiny výsledků směrem k vyšším rychlostem (při srovnání s obrázkem 8.21).

Přestože byla nejnižší hodnota míry zhoršení matematické kvality rovna $Q = 201{,}89$ ‰ při rychlosti $S = 58{,}71$, byla při výběru optimální konfigurace, která by reprezentovala výkonnost algoritmu ZCCFRAME$^{\text{prn}}$, zvolena nejnižší hodnota matematické kvality Q v okolí bodů korespondujících se srovnatelnou rychlostí $S = 539{,}55$ jako u optimální konfigurace pro verzi ZCCFRAME$^{\text{sub}}$. Důvodem této volby byl také fakt, že se v tomto okolí nachází množství bodů, u nichž byly hodnoty matematické míry zhoršení kvality Q pouze mírně vyšší než byla celkově nejnižší dosažená hodnota Q. Ke srovnání byla tedy za optimální zvolena konfigurace s hodnotami výkonnostních ukazatelů $S = 525{,}92$ a $Q = 226{,}98$ ‰.

U varianty algoritmu ZCCFRAME$^{\text{prn}}$ bylo dosaženo nejlepších výsledků z rodiny algoritmů ZCCFRAME, zejména u ukazatele míry matematického zhoršení kvality Q. Pokud by se srovnávala míra zrychlení S, pak pro optimální konfigurace parametrů dávají všechny verze algoritmu ZCCFRAME velmi podobné výsledky, nicméně při srovnání grafů výkonnosti je zřejmé, že v souhrnu dosahuje varianta ZCCFRAME$^{\text{prn}}$ nejvyššího zrychlení (většina konfigurací ZCCFRAME$^{\text{prn}}$ je v grafu na obrázku 8.25 v pásmu za hodnotou $S \approx 400$).

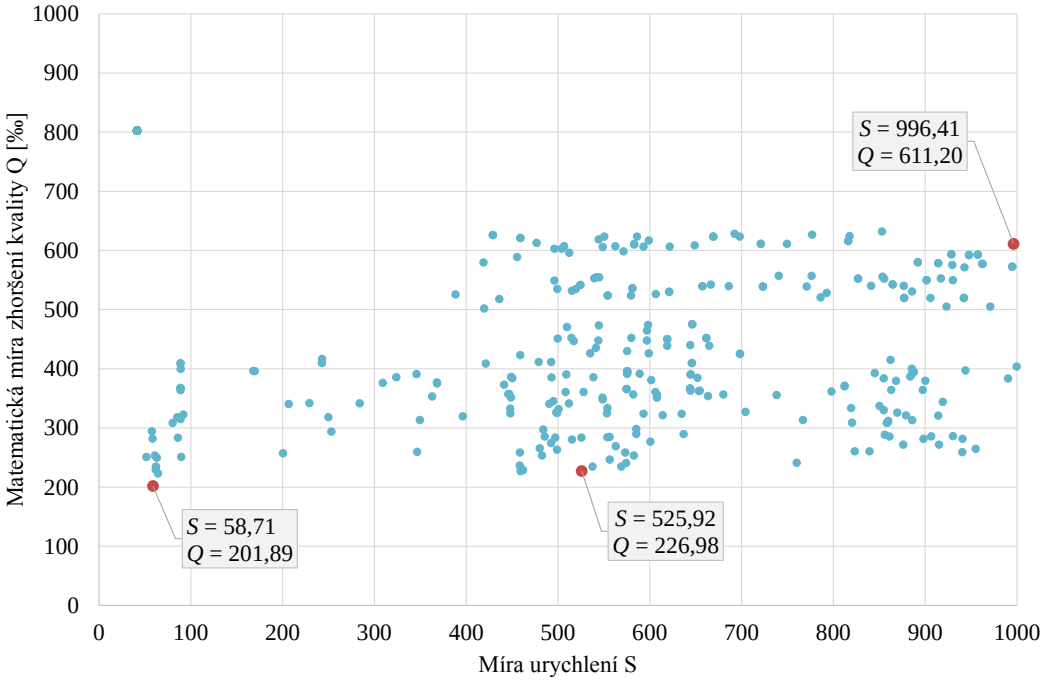
Největší nevýhodou algoritmu ZCCFRAME$^{\text{prn}}$ resp. všech variant algoritmu ZCCFRAME je však komplikovaná volba hodnot parametrů tak, aby bylo možné dosáhnout co nejlepších výsledků. Přestože byla provedena syntéza testovací množiny promluv ve stovkách různých kombinací, nebylo možné se s hodnotami matematické míry zhoršení kvality Q alespoň přiblížit očekávaným výsledkům. Lze se domnívat, že by bylo možné získat lepší výsledky, pokud by byly hodnoty parametrů určovány dynamicky pro každou větu např. podle počtu a rozmístění ZCC řetězců v grafu kandidátů, ale to by mimo jiné znamenalo navýšení výpočetní náročnosti celého procesu výběru jednotek. Dále by bylo nutné stanovit pravidla, jak parametry nastavit, což by samo o sobě již znamenalo neúměrnou složitost celého algoritmu (již tak složitého) s předpokladem nejistých výsledků a zlepšení zejména v oblasti kvality.

Při porovnání s optimalizovaným Viterbiovým algoritmem ve variantě VITPRUNE dosahuje algoritmus ZCCFRAME$^{\text{prn}}$ horších výsledků. Při srovnatelné hodnotě zrychlení $S = 524{,}99$ byla hodnota matematického zhoršení kvality u algoritmu VITPRUNE pouze $Q = 90{,}55$ ‰. Dokonce u konfigurace VITPRUNE s velmi agresivním prořezáváním, jež dávala urychlení $S = 4862{,}99$, byla hodnota matematického zhoršení kvality $Q = 191{,}68$ ‰, což je stále nižší hodnota, než nejnižší hodnota Q získaná algoritmem ZCCFRAME$^{\text{prn}}$. Kromě horších výkonnostních ukazatelů je algoritmus VITPRUNE také významně implementačně jednodušší a paměťové méně náročný.

Na základě uvedeného hodnocení a srovnání s algoritmem VITPRUNE, bylo upuštěno od dalšího zkoumání možností jak vylepšit algoritmy rodiny ZCCFRAME na úroveň, která by byla z pohledu kladených cílů této práce akceptovatelná. Poznatky získané při návrhu a testování algoritmu ZCCFRAME, zejména pak v oblasti struktury a vlastností ZCC řetězců, byly však využity při návrhu další metody výběru jednotek, u níž existovaly lepší předpoklady k dosažení vyššího výkonu při zachování kvality produkované řeči. Vznikl tak nový algoritmus, jehož popis je uveden v následujícím oddíle.

Tabulka 8.9: Hodnocení výkonu základního algoritmu ZCCFRAME s doplněnými podřetězci a provedením jejich prořezání – konfigurace dosahující nejlepšího výsledku matematické kvality

Vlastnost	Hodnota
Kód algoritmu	ZCCFRAME$^{\text{prn}}$
Míra urychlení (S)	525,92
Matematická míra zhoršení kvality (Q)	226,98 ‰
Míra fragmentace (CD)	28,35
Hodnoty parametrů: $LEN^{\min} = 3$, $TRANS^{\max} = 2$, $START^{\max} = 3$, $END^{\max} = 3$, $CHAINDEEP = 10$, $FRAMECOUNT^{\max} = 1000$, $CANDCOUNT^{\max} = 1000$, ${}_{zcc}C^{\text{t}}_{\text{avg}} = 0{,}3$, ${}_{zcc}C^{\text{t}}_{\max} = 0{,}33$, $GROUPCOUNT^{\max} = 300$	



Obrázek 8.25: Matematická míra zhoršení kvality ve vztahu ke zrychlení pro různé parametry algoritmu ZCCFRAME$^{\text{prn}}$ s doplněnými podřetězci a provedením jejich prořezání.

8.2.9 Algoritmus na bázi Viterbiova algoritmu využívající ZCC řetězce

Nevýhodou algoritmu ZCCFRAME je problém se stabilitou výkonnostních ukazatelů. Matematická kvalita Q sice byla vylepšena doplněním množiny ZCC řetězců o jejich podřetězce, ale tím vznikly problémy s vhodným nastavením parametrů, jako je počet koster, protože celkový počet zpracovávaných ZCC řetězců byl velmi vysoký. Tento problém byl řešen jejich prořezáváním podle limitů na hodnotu průměrné a maximální ceny cíle v nich obsažených kandidátů nebo podle dalších vlastností. Množina ZCC řetězců tak byla omezena pouze na řetězce, u nichž je pravděpodobné, že skutečně budou součástí cesty s globálně minimální kumulativní cenou. Ani po doplnění uvedených modifikací, nedosahoval algoritmus ZCCFRAME (resp. jeho nejlepší varianta ZCCFRAME$^{\mathrm{prn}}$) očekávaných výsledků.

Další úvahy vedly na návrh jiného algoritmu, který by zvýšil stabilitu ukazatelů výkonu, udržel vysokou míru urychlení a také zachoval myšlenku využití ZCC řetězců. Díky tomu, že se podařilo velikost vstupní množiny ZCC řetězců výrazně snížit pomocí prořezávání (i při minimální délce $LEN^{\min} = 2$), bylo možné uvažovat o algoritmu, který by používal podobný přístup jako základní Viterbiův algoritmus VITBASE, ale místo jednotlivých kandidátů by se pracovalo na úrovni ZCC řetězců. Byl tak vytvořen nový algoritmus značený kódem *ZCCVIT*.

8.2.9.1 Strategie algoritmu ZCCVIT

Hlavní myšlenkou algoritmu je pokusit se k výběru optimální cesty grafem využít v maximální možné míře předvybrané ZCC řetězce (včetně podřetězců) až do minimální možné délky o dvou kandidátech. Proti algoritmu ZCCFRAME nedochází k urychlení celého procesu vytvořením kandidátních koster, v nichž se poté doplňují chybějící úseky, ale postupně se zpracovávají všechny ZCC řetězce a pro každý se hledá nejlepší předchůdce resp. nejlepší možná cesta od počátku grafu. Tím, že se využívají i velmi krátké ZCC řetězce, je vysoce pravděpodobné, že se při hledání nejlepšího předchůdce podaří najít přímo sousedící ZCC řetězec a při aktualizaci hodnoty průběžné kumulativní ceny C^* tak bude potřeba pouze jeden výpočet ceny řetězení C^{c}.

Pokud je nutné najít cestu grafem spojující dva ZCC řetězce vzdálených od sebe více než jednu jednotku, pak se stejně jako u algoritmu ZCCFRAME

použije algoritmus BEFS. Je zřejmé, že s rostoucím výskytem těchto případů dochází k výraznému navýšení výpočetních nároků celého procesu, a proto je součástí strategie algoritmu ZCCVIT upřednostnit přímo navazující ZCC řetězce. Při hledání nejlepšího předchůdce jsou všechny přípustné předchozí ZCC řetězce seřazeny nejprve podle vzdálenosti od zpracovávaného ZCC řetězce (od nejbližšího), poté podle průběžné kumulativní ceny (od nejnižší) a případně ještě podle své délky (od nejdelšího). Pokud dojde k nalezení nejlepšího předchůdce v určité vzdálenosti, pak se cesty ke vzdálenějším ZCC řetězcům již nehledají.

8.2.9.2 Popis algoritmu ZCCVIT

Algoritmus má shodné některé části s algoritmem ZCCFRAME. Pracuje se stejnou množinou ZCC řetězců doplněnou o přípustné podřetězce jako u algoritmu ZCCFRAME a také používá algoritmus BEFS k vyhledání vhodných cest spojujících ZCC řetězce mezi sebou. Pro nalezení optimální cesty celým grafem se však již používá zcela jiného postupu, který odpovídá svým charakterem prohledávání grafu do šířky. Pokud by se množina ZCC řetězců po jejich prořezání uspořádala do stejného (zjednodušeného) grafu popsaného v oddíle 8.2.5.2 na straně 89, pak by strategie algoritmu ZCCVIT byla shodná se základním Viterbiovým algoritmem VITBASE. Rozdíl je v tom, že místo kandidátů reprezentují uzly grafu ZCC řetězce a že hrany jsou tvořeny cestami složenými z jednoho nebo více kandidátů původního grafu. Další odlišností od algoritmu VITBASE je to, že uzly grafu ZCC řetězců nejsou uniformními jednotkami a nelze tak použít všechny optimalizace jako u variant VITOPT nebo VITPRUNE (např. neplatí, že předchozí uzly se nacházejí na stejné pozici). Základní schéma algoritmu je znázorněno na obrázku 8.26.A.

Začátek hledání je shodný s algoritmem ZCCFRAME, tj. v grafu kandidátů se vyhledají všechny ZCC řetězce a jejich množina se doplní o všechny podřetězce až do minimální délky $LEN^{\min}$ (schéma doplnění podřetězců bylo uvedeno v předchozím textu na obrázku 8.20). Z takto vytvořené množiny se odstraní ZCC řetězce, u nichž je průměrná cena cíle kandidátů nižší než hodnota parametru ${}_{zcc}C^{\mathrm{t}}_{\mathrm{avg}}$ nebo v nichž cena cíle C^{t} libovolného kandidáta překročí limit stanovený parametrem ${}_{zcc}C^{\mathrm{t}}_{\mathrm{max}}$. Následně se ZCC řetězce rozdělí do skupin podle indexu počátečního kandidáta a své délky. V rámci skupin se ZCC řetězce seřadí podle součtu cen cíle C^{t} v nich obsažených kandidátů

a v každé skupině se ponechá pouze množství nejlepších kandidátů určené parametrem $GROUPCOUNT^{\max}$.

Dalším krokem je seřazení všech zbylých ZCC řetězců podle pozice počátečního kandidáta a poté se postupně ke každému z nich vyhledá nejlepší předchůdce (viz schéma na obrázku 8.26). To se provádí tak, že se vyhledá množina všech předcházejících ZCC řetězců v maximální vzdálenosti dle parametru $TRANS^{\max}$. Množina se seřadí podle indexu koncových kandidátů řetězců (od nejvyššího – tj. od nejbližšího řetězce) a podle průběžné kumulativní ceny C^* do nich vedoucí nejlepší cesty od počátku grafu. Pak se postupně hledá vždy nejlepší cesta k předcházejícím ZCC řetězcům algoritmem BEFS (viz obrázek 8.26.C).

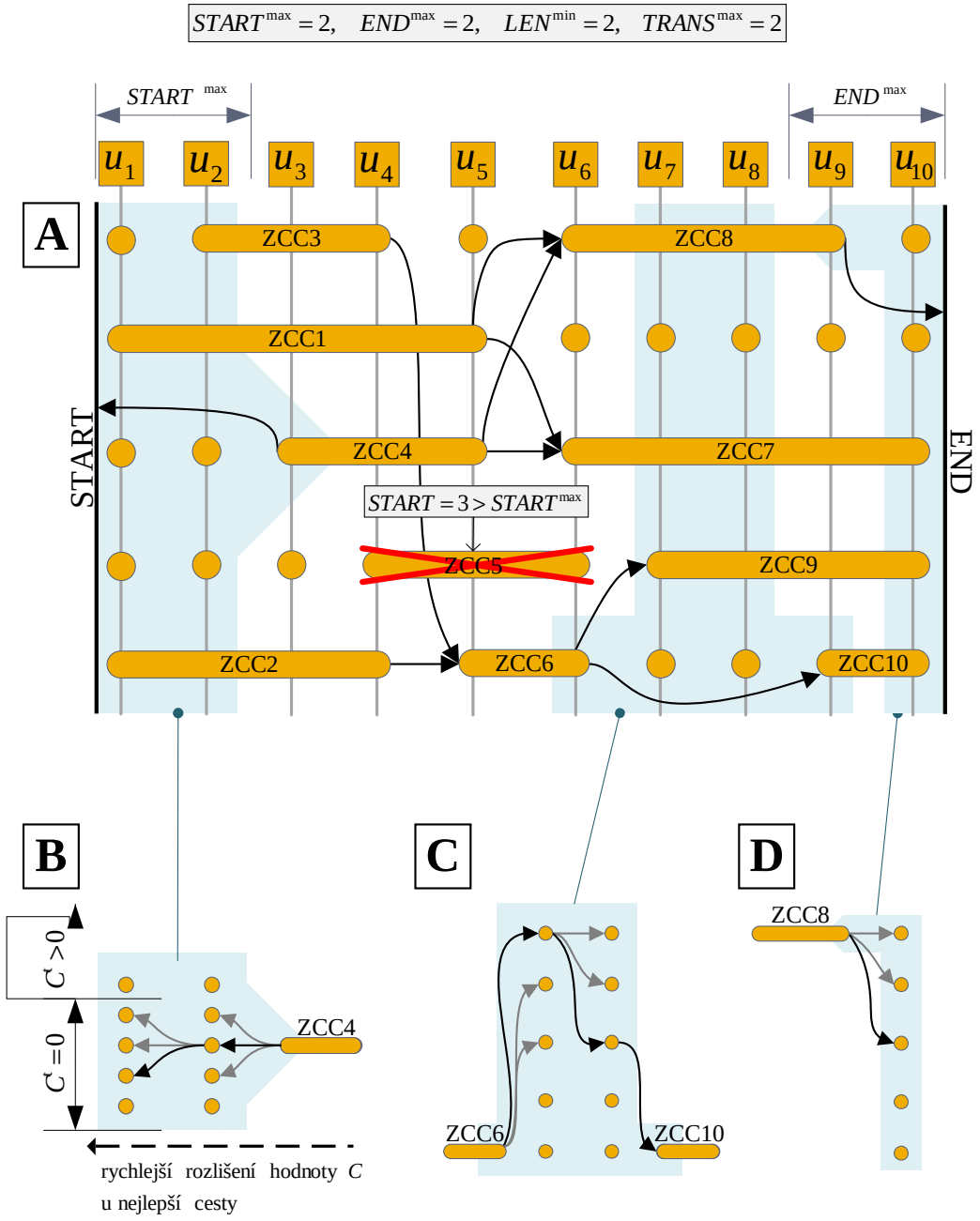
Pokud ZCC řetězci nepředchází žádný jiný řetězec, pak se vyhledá pouze nejlepší cesta od počátku grafu pomocí algoritmu BEFS, přičemž stejně jako u algoritmu ZCCFRAME je příslušný podgraf revertovaný (viz obrázek 8.26.B). Počáteční pozice ZCC řetězce však nesmí přesáhnout hodnotu stanovenou parametrem $END^{\max}$, jinak byl byl řetězec z hledání vyřazen (příkladem je řetězec ZCC5 na obrázku 8.26.A).

Pokud pozice posledního kandidáta v ZCC řetězci dosáhne limitu definovaného parametrem $END^{\max}$, pak se po vyhledání nejlepšího předchůdce navíc nalezne i cesta do konce grafu kandidátů (viz obrázek 8.26.D).

Vítěznou se nakonec stává úplná cesta s nejnižší hodnotou kumulativní ceny C a posloupnost odpovídajících kandidátů se získá zpětným trasováním přes odkazy na nejlepší předchůdce ZCC řetězců v kombinaci s cestami, které je spojují.

SPECIÁLNÍ PŘÍPAD NASTAVENÍ PARAMETRŮ

Jednou z vlastností algoritmu ZCCVIT je to, že pomocí specifického nastavení parametrů lze změnit jeho chování na základní Viterbiův algoritmus VITBASE. Pokud by totiž platilo, že parametr $LEN^{\min} = 1$, pak by výsledky byly shodné s cestami získanými algoritmem VITBASE a matematická míra zhoršení kvality Q by byla nulová. Rychlost procesu výběru jednotek S by ale byla nižší, protože by se nadbytečně zpracovávaly i ZCC řetězce s délkou dvě a více jednotek (dávalo by smysl je vynechat a pak by byla rychlost shodná, tj. $S = 1{,}00$).



Obrázek 8.26: Schéma algoritmu ZCCVIT. V části A jsou znázorněny ZCC řetězce a vyhodnocovaná spojení mezi nimi. Číselné značení ZCC řetězců odpovídá pořadí jejich zpracování. Část B zobrazuje schéma hledání cesty od začátku grafu k řetězci ZCC4, přičemž hledání probíhá v opačném směru, tj. od prvního uzlu řetězce ZCC4 směrem k počátku grafu. Část C znázorňuje schéma hledání cesty mezi dvěma ZCC řetězci uvnitř grafu. V části D je pak zobrazeno hledání od řetězce ZCC8 směrem ke konci grafu – v tomto případě probíhá hledání dopředným směrem.

8.2.9.3 Implementace algoritmu ZCCVIT

Předpis hledání algoritmem ZCCVIT je následující:

1. Ke všem uzlům grafu kandidátů se vypočítá cena cíle C^{t}.
2. V grafu kandidátů se vyhledají všechny ZCC řetězce s minimální délkou $LEN^{\min}$.
3. Množina ZCC řetězců se doplní o všechny podřetězce do minimální délky $LEN^{\min}$.
4. V množině ZCC řetězců se ponechají pouze ZCC řetězce splňující podmínky:
   - Průměrná hodnota ceny cíle C^{t} kandidátů ZCC řetězce je menší nebo rovna hodnotě parametru ${}_{zcc}C^{\mathrm{t}}_{\mathrm{avg}}$, tj. platí:
     $\frac{1}{N}\sum_{i=1}^{N} C^{\mathrm{t}}(u_i) \leq {}_{zcc}C^{\mathrm{t}}_{\mathrm{avg}}$, kde N je počet kandidátů ZCC řetězce a u_i je kandidát s indexem i.
   - Maximální hodnota ceny cíle C^{t} kandidátů ZCC řetězce je menší nebo rovna hodnotě parametru ${}_{zcc}C^{\mathrm{t}}_{\max}$, tj. platí:
     $\forall i = \{1, \ldots, N\} : C^{\mathrm{t}}(u_i) \leq {}_{zcc}C^{\mathrm{t}}_{\max}$, kde N je počet kandidátů ZCC řetězce a u_i je kandidát s indexem i.
5. ZCC řetězce se uspořádají do skupin se shodnou pozicí počátečního kandidáta i shodnou délkou. V každé skupině se ponechá pouze $GROUPCOUNT^{\max}$ ZCC řetězců s nejnižším součtem cen cíle kandidátů C^{t}, z nichž jsou tvořeny. Ze ZCC řetězců, které ve skupinách zůstaly, se utvoří nová množina všech ZCC řetězců k dalšímu zpracování.
6. Množina všech ZCC řetězců se seřadí od nejnižší pozice počátečního kandidáta.
7. Z množiny ZCC řetězců se vybere první dosud nezpracovaný ZCC řetězec s nejnižší pozicí počátečního kandidáta a provede se:
   (a) Pokud ZCC řetězci nepředchází žádný ZCC řetězec ve vzdálenosti $TRANS^{\max}$, pak:

i. Pokud je ZCC řetězec maximálně ve vzdálenosti $START^{\max}$ od počátku grafu, pak se vyhledá cesta od počátku grafu k prvnímu kandidátovi ZCC řetězce. Použije se algoritmus BEFS (prohledávaný podgraf je revertovaný) a výsledná cesta se uloží k ZCC řetězci, stejně tak i průběžná kumulativní cena cesty.

ii. Pokud je ZCC řetězec ve vzdálenosti větší než $START^{\max}$, pak se odstraní a dále se s ním nepracuje.

(b) Pokud je ZCC řetězec vzdálen od konce grafu maximálně $END^{\max}$ pozic, pak se pomocí algoritmu BEFS vyhledá cesta od koncového uzlu až na konec grafu kandidátů. Zpětným trasováním přes nejlepší předchůdce a cest mezi nimi se složí výsledná úplná cesta a uloží se do množiny FINAL včetně informace o její kumulativní ceně.

(c) Předchozí ZCC řetězce se seřadí podle vzdálenosti od zpracovávaného ZCC řetězce (od nejbližšího) a postupně se vyhledávají cesty grafem kandidátů mezi nimi a zpracovávaným ZCC řetězcem. Pokud je nalezena cesta ze ZCC řetězce v určité vzdálenosti, pak se již cesty ze ZCC řetězců ve větší vzdálenosti nehledají. Nejlepším předchůdcem se stává ZCC řetězec, přes něhož vedoucí cesta má nejnižší průběžnou kumulativní cenu. K ZCC řetězci se uloží odkaz na nejlepšího předchůdce, seznam kandidátů, z nichž je cesta k nejlepšímu předchůdci složena, a dojde k aktualizaci průběžné kumulativní ceny.

8. Algoritmus se vrací zpět na bod 7, dokud se nezpracují všechny ZCC řetězce.

9. Vítěznou cestou grafem kandidátů se stává cesta s nejnižší kumulativní cenou z množiny FINAL.

8.2.9.4 Výkonnost algoritmu ZCCVIT

Podobně jako u předchozích technik výběru jednotek byla provedena syntéza testovací množiny promluv s různými kombinacemi hodnot parametrů algoritmu ZCCVIT (rozsah hodnot parametrů viz tabulka B.7 v příloze B) a výsledky jsou shrnuty v grafu na obrázku 8.27. Přestože u žádného z experimentů nebyly vybrány u všech promluv posloupnosti kandidátů jednotek

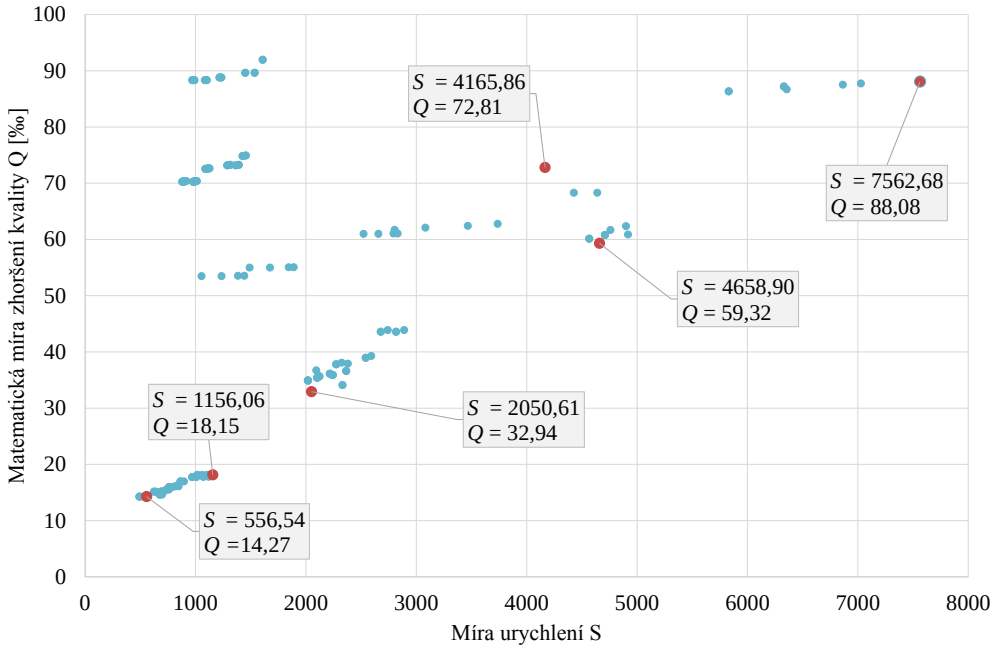
Tabulka 8.10: Souhrnné hodnocení algoritmu ZCCVIT

Vlastnost	Hodnota
Kód algoritmu	ZCCVIT
Míra urychlení (S)	556,54
Matematická míra zhoršení kvality (Q)	14,27 ‰
Míra fragmentace (CD)	28,51 %
Hodnoty parametrů: $LEN^{\min} = 2$, $TRANS^{\max} = 1$, $START^{\max} = 3$, $END^{\max} = 3$, $CHAINDEEP = 100$, $GROUPCOUNT^{\max} = 100$, ${}_{zcc}C^{\mathrm{t}}_{\mathrm{avg}}$=0,3, ${}_{zcc}C^{\mathrm{t}}_{\max}$=0,33	

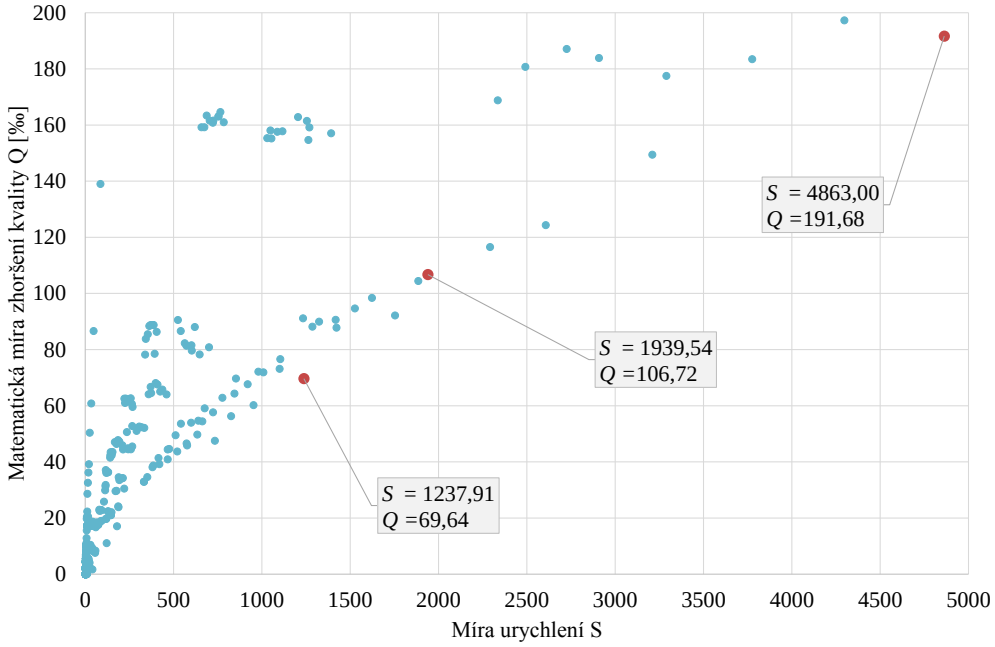
s globálně minimální kumulativní cenou C (viz algoritmus VITBASE), bylo u nejlepšího experimentu dosaženo této shody u 7 z celkových 20 promluv. U ostatních promluv byly odlišné řečové segmenty vybrány ve velmi malém množství případů a tomu také odpovídají velmi nízké hodnoty ukazatele matematické míry zhoršení kvality. Nejnižší hodnota pak činila $Q = 14{,}27$ ‰ při velmi vysoké míře urychlení $S = 556{,}54$ (viz tabulka 8.10) a jak je patrné z grafu výkonnosti, rychlost se postupně zvyšovala až k hodnotám přesahující míru urychlení $S = 1000$, aniž by míra zhoršení matematické kvality překročila hodnotu $Q = 20$ ‰. Hraniční hodnotou v tomto smyslu pak byla konfigurace s výkonem $S = 1156{,}06$ a $Q = 18{,}15$ ‰, přičemž i v tomto případě byla míra shody s referenčním algoritmem velmi vysoká (6 promluv z 20 bylo zcela totožných).

Graf výkonu také vypovídá o velmi dobré stabilitě algoritmu z pohledu kvality, protože i při extrémní míře urychlení $S = 7562{,}68$ byla hodnota míry zhoršení matematické kvality rovna $Q = 88{,}08$ ‰, což je např. přibližně polovina nejnižší hodnoty u nejlépe hodnocené verze algoritmu ZCCFRAME$^{\mathrm{prn}}$ s doplněnými a prořezaným podřetězci. Algoritmem ZCCVIT byly v matematické rovině hodnocení získány i lepší výsledky než u algoritmu VITPRUNE, u něhož roste hodnota ukazatele Q přibližně dvakrát rychleji (viz graf na obrázku 8.28) a např. již při zrychlení $S = 1237{,}91$ byla hodnota $Q = 69{,}64$ ‰ (při srovnatelném urychlení u algoritmu ZCCVIT $S = 1156{,}06$ byla hodnota $Q = 18{,}15$ ‰).

Souhrnně lze konstatovat, že v matematickém měřítku lze algoritmem ZCCVIT dosáhnout významného urychlení a zároveň udržet míru zhoršení kvality Q na velmi nízkých hodnotách. Z tohoto pohledu splnil tento algoritmus očekávání stanovená v této práci a byl tak zařazen do hodnocení v percepční oblasti prostřednictvím poslechových testů. Výsledkům hodnocení a závěrečnému srovnání s ostatními algoritmy je věnován následující oddíl 8.3.



Obrázek 8.27: Matematická míra zhoršení kvality ve vztahu ke zrychlení pro různé parametry algoritmu ZCCVIT.



Obrázek 8.28: Matematická míra zhoršení kvality ve vztahu ke zrychlení pro různé parametry algoritmu VITPRUNE v celém rozsahu. Na rozdíl od grafu na obrázku 8.5 jsou zde zobrazeny úplně všechny výsledky experimentů.

8.3 Souhrnné hodnocení implementovaných algoritmů

V předchozích oddílech této kapitoly byla popsána celá řada algoritmů výběru jednotek v úloze syntézy řeči z textu. Pro každý z algoritmů byly provedeny experimenty s různými kombinacemi nastavení jejich parametrů (pokud se v algoritmu používaly) a nad testovací množinou promluv bylo provedeno určení míry urychlení S a hodnot kvalitativních ukazatelů, tj. míry zhoršení matematické kvality Q a míru fragmentace CD (definice ukazatelů byla uvedena v kapitole 7). Tento způsob hodnocení byl zvolen proto, že srovnání algoritmů a hodnocení kvality generovaných promluv v percepční rovině pomocí poslechových testů je velmi časově i organizačně náročná činnost. Proto byly matematické ukazatele použity k prvotnímu výběru vhodných algoritmů a typicky i jejich optimálních variant nastavení parametrů. Pro snadnější srovnání algoritmů navzájem i vůči referenčnímu algoritmu VITBASE byly všechny experimenty v této fázi práce prováděny nad stejnou databází řečových segmentů vytvořené z řečového korpusu nahraného mužským řečníkem. Až následně byly nejlepší verze algoritmů podrobeny srovnání prostřednictvím poslechových testů, které již byly kvůli vyšší objektivitě realizovány jak pro mužský, tak i pro ženský hlas.

8.3.1 Hodnocení algoritmů matematickými ukazateli

Prvním implementovaným algoritmem s cílem urychlit proces výběru jednotek, byl označen kódem VITOPT a doplnil základní Viterbiův algoritmus VITORIG o optimalizaci při hledání nejlepších předchůdců jednotlivých uzlů grafu kandidátů (viz oddíl 8.1.2). Algoritmus zaručuje nalezení cest grafem s globálně minimální kumulativní cenou C (tj. generované promluvy jsou shodné jako u referenčního algoritmu VITBASE), tomu ovšem odpovídá i nízká míra urychlení $S = 1{,}52$.

Doplněním prořezávání po vyhodnocení kandidátů jednoho celého kroku a související optimalizace vznikla varianta algoritmu značená VITBEAM (viz oddíl 8.1.3). V tomto případě již nebylo zaručeno nalezení posloupnosti s globálně minimální cenou C, nicméně při vhodném konzervativním nastavení parametrů bylo možné dosáhnout stejných výstupů jako u základního Viterbiova algoritmu VITORIG resp. jeho referenční verze VITBASE.

Hodnota ukazatele zrychlení byla v takovém případě velmi nízká a činila $S = 2{,}09$. Nastavením agresivnějšího prořezávání, kdy byly po vyhodnocení každého kroku ponechány pouze desítky kandidátů dané jednotky, bylo možné výběr více urychlit, ale i tak nebyl zisk rychlosti nijak významný a také došlo navýšení hodnot matematického ukazatele kvality Q.

Z algoritmu VITBEAM pak vycházel další algoritmus VITPRUNE, jenž vznikl doplněním další techniky prořezávání, a to podle ceny cíle C^{t}. Konfigurace algoritmu VITPRUNE, která ještě bezpečně zajišťovala nalezení posloupnosti kandidátů s globálně minimální kumulativní cenou C, dávala zrychlení $S = 8{,}35$ (detaily jsou uvedené v tabulce 8.4 v oddíle 8.1.4). Výhodou tohoto konzervativního nastavení byly vyšší hodnoty parametrů prořezávání (v řádu stovek) a otevírá se zde možnost jejich snížení při zachování velmi dobré kvality. Algoritmus je také poměrně stabilní a ukazatele matematické kvality Q i fragmentace CD rostou pozvolna (viz graf na dříve uvedeném obrázku 8.28). Z těchto důvodů byl algoritmus VITPRUNE, na rozdíl od algoritmů VITOPT nebo VITBEAM, zařazen mezi algoritmy, které byly porovnávány poslechovými testy.

Následně byly v práci uvedeny algoritmy využívající ZCC řetězců k urychlení procesu výběru jednotek. V této části byl uveden algoritmus BEFS, jenž sám o sobě není vhodný k prohledávání celého grafu kandidátů, nicméně se ukázal jako velmi vhodný k dohledávání cest mezi ZCC řetězci u dalších algoritmů. První algoritmus založený na ZCC řetězcích byl označen kódem ZCCBEFS a vznikl modifikací algoritmu BEFS, která spočívala v expanzi nejslibnějšího uzlu o celý ZCC řetězec, pokud na daný uzel navazoval. V případě algoritmu ZCCBEFS odpovídal nejlepší výsledek míře zhoršení matematické kvality $Q = 17{,}17$ ‰ při urychlení $S = 9{,}01$. Celkově nebyly výsledky tohoto algoritmu v kontextu této práce uspokojivé a algoritmus byl zatížen sklonem k upřednostnění lokálních minim, takže nebyl více zkoumán a poslechové testy pro něj nebyly provedeny.

Druhý uvedený algoritmus založený na ZCC řetězcích byl nazván ZCCFRAME. Jeho hlavní myšlenkou bylo pokusit se nejprve heuristickými postupy vybrat tzv. kostry složené pouze ze ZCC řetězců a u nich poté dohledat vhodné samostatné kandidáty k doplnění chybějících úseků mezi řetězci. Přestože bylo testováno velké množství konfigurací, byly výsledky z pohledu matematické kvality nevyhovující (hodnoty míry zhoršení matematické kvality Q se pohybovaly v řádu stovek). I přes další analýzu vlastností ZCC

řetězců v cestách s globálně minimální výší kumulativní ceny C se nepodařilo snížit míru zhoršení matematické kvality pod hodnotu $Q < 200$ ‰ u žádné z konfigurací. Problémem u tohoto algoritmu byly velké rozdíly ve výsledcích u jednotlivých promluv. Řešením by pravděpodobně bylo vytvoření pravidel, jak nastavit hodnoty dynamicky v závislosti na počtu a vlastnostech ZCC řetězců u jednotlivých promluv, ovšem za cenu dalšího zvýšení složitosti algoritmu. Pokud se také přihlédne k vyšší paměťové náročnosti, pak se algoritmus ZCCFRAME v žádné z variant nejeví perspektivně, proto nebyl dále testován.

Nejlepších výsledků jak v rovině matematických kvalitativních ukazatelů, tak i rychlosti dosahoval poslední z navržených algoritmů ZCCVIT. S využitím výsledků analýzy ZCC řetězců v posloupnostech vybraných referenčním algoritmem VITBASE byla nalezena konfigurace parametrů dávající velmi vysokou míru urychlení $S = 556{,}54$ při velmi nízké hodnotě matematického zhoršení kvality $Q = 14{,}27$ ‰ (navíc se pro 7 z 20 promluv testovací množiny zcela shodovaly vybrané posloupnosti kandidátů s výběrem pomocí referenčního algoritmu VITBASE). Protože jsou základem algoritmu ZCCVIT ZCC řetězce, byla u výsledných promluv dle předpokladu naměřena nižší hodnota fragmentace CD, která byla se svou hodnotou $CD = 28{,}51$ % dokonce nižší než u referenčního algoritmu VITBASE ($CD = 29{,}39$ %). Algoritmus obecně vykazoval vysokou míru stability a hodnoty matematických ukazatelů byly nejlepší ze všech zkoumaných algoritmů. Protože byly promluvy vygenerované tímto algoritmem na první poslech nerozeznatelné od promluv odpovídajících výstupům algoritmu VITBASE, byl algoritmus ZCCVIT zařazen do závěrečného srovnání pomocí poslechových testů.

8.3.2 Poslechové testy

Z dosud provedených experimentů a hodnocení vyplývá, že nejslibnějších výsledků bylo dosaženo algoritmem ZCCVIT. Míra urychlení u tohoto algoritmu je velmi vysoká i pro konfiguraci s nejnižší hodnotou míry zhoršení matematické kvality Q, která na rozdíl od algoritmu VITPRUNE nebyla nulová. Kromě toho je algoritmus ZCCVIT při dalším zvyšování rychlosti stabilnější vzhledem ke kvalitativním matematickým ukazatelům a proto byly poslechové testy koncipovány tak, aby se ověřilo, jak je řeč vygenerovaná tímto algoritmem hodnocena posluchači při srovnání s referenčním algoritmem

VITBASE a algoritmem VITPRUNE.

První skupina poslechových testů byla realizována za použití databáze řečových segmentů, jež byla použita ve všech předchozích experimentech a hodnoceních. Databáze byla vytvořena segmentací řečového korpusu nahraného mužským hlasem a obsahovala 670 757 segmentů.

Aby se zvýšila objektivita hodnocení, byla uskutečněna i druhá sada testů, při nichž se použila odlišná databáze řečových segmentů. Pro tento účel byla vybrána databáze vytvořená z řečového korpusu nahraného ženským hlasem a čítala 656 406 segmentů.

8.3.2.1 Varianty algoritmů použité v poslechových testech

V oddíle 8.3.1 byly souhrnně hodnoceny jednotlivé algoritmy a byl proveden výběr těch, jež mají na základě matematických měřítek potenciál vyhovět cílům stanoveným v této práci. K hodnocení poslechovými testy s více posluchači byly vybrány algoritmy VITPRUNE a ZCCVIT. Oba algoritmy vykazují dobré výsledky z pohledu urychlení a stability matematických ukazatelů kvality.

U algoritmu VITPRUNE bylo možné při méně agresivním prořezávání dosáhnout výběru posloupností realizací jednotek s globálně minimální kumulativní cenou C, a tedy výsledná kvalita je pak naprosto shodná s výstupem referenčního algoritmu VITBASE, byť za cenu nižší míry urychlení S. Pro testovanou databázi řečových jednotek (mužský hlas) odpovídala bezpečnému výběru matematicky ideálních posloupností konfigurace s urychlením $S = 8{,}35$, jež bude dále značena kódem VITPRUNE$^{\text{cns}}$ [17)].

V případě algoritmu ZCCVIT se u konfigurací parametrů s nejlepším hodnocením kvality se sice podařilo nalézt shodu s referenčním algoritmem VITBASE, ovšem vždy se u několika vět výsledky nepatrně odlišovaly. Nejnižší dosažená matematická míra zhoršení kvality byla rovna hodnotě $Q = 14{,}27$ ‰ při zrychlení $S = 556{,}54$. Tato konfigurace bude označena kódem ZCCVIT$^{\text{opt}}$.

Protože je míra zrychlení u konfigurace ZCCVIT$^{\text{opt}}$ mnohonásobně vyšší, než u varianty modifikovaného Viterbiova algoritmu VITPRUNE$^{\text{cns}}$, byla do testů zařazena konfigurace algoritmu VITPRUNE s odpovídající nejbližší

[17)] Varianta VITPRUNE$^{\text{cns}}$ je dále uváděna pouze pro srovnání a poslechové testy pro ni nebyly prováděny, protože výstup algoritmu je naprosto shodný s výstupem referenčního algoritmu VITBASE.

mírou urychlení. Přesná hodnota zrychlení u této konfigurace byla $S = 563{,}74$ a tato konfigurace byla nazvána VITPRUNE$^{\text{opt}}$.

U varianty VITPRUNE$^{\text{opt}}$ se kvůli vyšší míře prořezávání projevilo zhoršení ukazatele matematické míry zhoršení kvality, a to na hodnotu $Q = 60{,}27$ ‰. Proto byla mezi testované verze algoritmů zařazena i konfigurace algoritmu ZCCVIT s obdobnou mírou zhoršení matematické kvality $Q = 60{,}88$ ‰, která znamenala velmi vysoké urychlení $S = 4917{,}20$ a která bude dále uváděna jako ZCCVIT$^{\text{fst}}$. K ní pak byla opět vyhledána konfigurace algoritmu VITPRUNE s nejbližší hodnotou míry urychlení, a tou byla konfigurace nazvaná VITPRUNE$^{\text{fst}}$ se zrychlením $S = 4863{,}00$. U této varianty je vhodné poznamenat, že míra matematického zhoršení kvality dosáhla již poněkud vyšší hodnoty $Q = 191{,}68$ ‰, přičemž u obdobně rychlé verze ZCCVIT$^{\text{fst}}$ byla tato hodnota pouze $Q = 60{,}88$ ‰ (což je dokonce nižší hodnota než u mnohem pomalejší varianty VITPRUNE$^{\text{opt}}$).

Pro potřeby poslechových testů byly tedy vybrány celkem čtyři konfigurace algoritmů VITPRUNE a ZCCVIT. Jejich výčet, včetně matematických ukazatelů kvality, je uveden v tabulce 8.11. V tabulce jsou pro přehlednost doplněny i hodnoty pro variantu VITPRUNE$^{\text{cns}}$.

8.3.2.2 Poslechové testy pro mužský hlas

V prvním z poslechových testů, kterého se zúčastnilo 15 respondentů, byly porovnávány promluvy testovací množiny vygenerované optimální variantou ZCCVIT$^{\text{opt}}$ a referenčním algoritmem VITBASE s cílem ověřit, zda nedošlo ke zhoršení kvality (popis způsobu realizace poslechových viz oddíl 7.3.2).

Tabulka 8.11: Přehled variant algoritmů použitých při hodnocení a poslechových testech pro mužský hlas

Algoritmus	Míra urychlení (S)	Matematická míra zhoršení kvality (Q) [‰]	Míra fragmentace (CD) [%]
VITBASE	1,00	0,00	29,39
VITPRUNE$^{\text{cns}}$	8,35	0,00	29,39
ZCCVIT$^{\text{opt}}$	556,54	14,27	28,51
VITPRUNE$^{\text{opt}}$	563,74	62,27	38,89
ZCCVIT$^{\text{fst}}$	4917,20	60,88	27,22
VITPRUNE$^{\text{fst}}$	4863,00	191,68	40,35

V naprosté většině případů odpovědí (87,50 %) byly promluvy posouzeny tak, že jejich kvalita je shodná. Počet odpovědí, u nichž posluchači preferovali jednu z promluv, byl velmi nízký, a to pouze 5,42 % pro algoritmus ZCCVIT$^{\text{opt}}$ resp. 7,08 % ve prospěch algoritmu VITBASE. V absolutních číslech se jednalo o 13 resp. 17 odpovědí, kdy byla kvalita u jedné z promluv považována za lepší. Rozdíl je zanedbatelný a s ohledem na celkový počet odpovědí lze říci, že se kvalita syntetické řeči při použití algoritmu ZCCVIT$^{\text{opt}}$ nezhoršila. Počet odpovědí ve prospěch algoritmu VITBASE byl však nepatrně vyšší, a proto byla provedena statistická analýza znaménkovým testem. Ta měla potvrdit či vyvrátit nulovou hypotézu, že kvalita výstupu je u obou algoritmů shodná na hladině významnosti $\alpha = 0{,}05$. Výpočtem z výše uvedených hodnot vychází hodnota $p = 0{,}2923$, a protože platí $p = 0{,}2923 > \alpha = 0{,}05$, nelze nulovou hypotézu na hladině významnosti $\alpha = 0{,}05$ zamítnout, tj. to, že by při daných počtech odpovědí byl výstup algoritmu VITBASE hodnocen lépe, není statisticky významné.

U druhého poslechového testu se stejnou účastí 15 posluchačů se porovnávaly varianty algoritmů ZCCVIT$^{\text{opt}}$ a VITPRUNE$^{\text{opt}}$. V tomto případě byla shodná kvalita hodnocena u nižšího počtu odpovědí (61,90 %) než u prvního testu. Naopak vrostl počet promluv, u nichž byl preferován algoritmus ZCCVIT$^{\text{opt}}$ (22,86 %) nebo algoritmus VITPRUNE$^{\text{opt}}$ (15,24 %). Rozdíly v preferencích jednoho nebo druhého algoritmu jsou opět minimální, avšak algoritmus ZCCVIT$^{\text{opt}}$ byl hodnocen nepatrně lépe, přičemž rozdíl byl statisticky významný (viz tabulka 8.12).

U posledního testu zaměřeného na srovnání algoritmů pro mužský hlas byly promluvy vygenerovány verzemi algoritmů s agresivním prořezáváním ZCC řetězců (ZCCVIT$^{\text{fst}}$) a realizací kandidátů jednotek (VITPRUNE$^{\text{fst}}$). Varianta ZCCVIT$^{\text{fst}}$ založená na ZCC řetězcích byla hodnocena lépe (31,03 % proti 16,90 % u varianty VITPRUNE$^{\text{fst}}$), a to s větším rozdílem než u předchozího testu s optimálními variantami stejných algoritmů. Výsledky tak korespondují i s matematickými ukazateli kvality Q a CD, které měla varianta ZCCVIT$^{\text{fst}}$ lepší než varianta VITPRUNE$^{\text{fst}}$ s obdobným zrychlením. Hodnoty matematických ukazatelů byly dokonce i lepší než optimální konfigurace VITPRUNE$^{\text{opt}}$ (blíže viz dříve uvedená tabulka 8.11).

Shrnutí výsledků poslechových testů u databáze řečových segmentů pro mužský hlas je uvedeno v tabulce 8.12.

Tabulka 8.12: Výsledky srovnávacích poslechových testů pro mužský hlas

Srovnávané varianty		Hodnocení [%]			Test statistické významnosti ($\alpha = 0{,}05$)	
A	B	A = B	A > B	A < B	p	rozdíl stat. významný
ZCCVIT$^{\text{opt}}$	VITBASE	87,50	5,42	7,08	0,2923	ne
ZCCVIT$^{\text{opt}}$	VITPRUNE$^{\text{opt}}$	61,90	22,86	15,24	0,0465	ano
ZCCVIT$^{\text{fst}}$	VITPRUNE$^{\text{fst}}$	52,07	31,03	16,90	0,0003	ano

8.3.2.3 Poslechové testy pro ženský hlas

Druhá sada poslechových testů byla realizována s promluvami vytvořenými nad databází, která byla vytvořena z řečového korpusu nahraného ženským hlasem. Podobně jako u mužského hlasu byly stejným postupem popsaným v oddíle 8.3.2.1 vybrány konfigurace odpovídající variantám algoritmů ZCCVIT$^{\text{opt}}$, VITPRUNE$^{\text{opt}}$, ZCCVIT$^{\text{fst}}$ a VITPRUNE$^{\text{fst}}$. Jejich souhrn včetně matematických ukazatelů kvality je uveden v tabulce 8.13.

Tabulka 8.13: Přehled variant algoritmů použitých při hodnocení a poslechových testech pro ženský hlas)

Algoritmus	Míra urychlení (S)	Matematická míra zhoršení kvality (Q) [‰]	Míra fragmentace (CD) [%]
VITBASE	1,00	0,00	29,36
VITPRUNE$^{\text{cns}}$	7,12	0,00	29,36
ZCCVIT$^{\text{opt}}$	438,30	18,08	28,56
VITPRUNE$^{\text{opt}}$	449,05	71,14	37,02
ZCCVIT$^{\text{fst}}$	4165,86	72,81	26,67
VITPRUNE$^{\text{fst}}$	4135,52	459,21	45,55

Na základě zkušenosti s poslechovými testy u mužského hlasu a také s ohledem na potřebu provést v rámci této práce řadu dalších poslechových testů bylo přistoupeno ke snížení celkového počtu hodnocených promluv na 10 a také také bylo vynecháno ne zcela nezbytné srovnání variant s agresivním prořezáváním ZCCVIT$^{\text{fst}}$ a VITPRUNE$^{\text{fst}}$.

První poslechový test srovnávající variantu algoritmu ZCCVIT$^{\text{opt}}$ s referenčním algoritmem VITBASE přinesl podobné výsledky jako u stejného testu pro mužský hlas. U nepatrně více odpovědí (10,10 %) byly preferovány

promluvy vytvořené algoritmem VITBASE. Algoritmus ZCCVIT^{opt} byl preferován v 9,09 % případů. Shodná kvalita u obou algoritmů tak byla hodnocena v 80,81 % odpovědí posluchačů. Podobně jako u mužského hlasu byla pomocí znaménkového testu ověřena nulová hypotéza, že kvalita testovaných variant je shodná. Počty odpovědí byly v tomto případě 9 ve prospěch varianty ZCCVIT^{opt} a 10 ve prospěch VITBASE. Výpočtem byla určena hodnota $p = 0{,}5$ pro hladinu významnosti $\alpha = 0{,}05$, což znamená, že hypotézu opět nelze zamítnout ($p = 0{,}5 > \alpha = 0{,}05$) a že tedy lepší kvalita u algoritmu VITBASE není statisticky významná.

V druhém testu hodnotili posluchači rozdíly mezi optimálními variantami algoritmů ZCCVIT^{opt} a VITPRUNE^{opt}. Z výsledků tohoto konkrétního testu plyne, že posluchači více preferovali (30,00 % případů) promluvy generované variantou ZCCVIT^{opt} před výstupy získanými variantou VITPRUNE^{opt} (15,29 % případů). I tak byla shodná kvalita hodnocena ve více jak polovině hodnocených promluv (54,71 %). Podobně jako u stejného testu pro mužský hlas lze tak z hlediska kvality označit algoritmus ZCCVIT^{opt} za lepší (rozdíl je statisticky významný).

Detailní výsledky poslechových testů pro ženský hlas jsou zaznamenány v tabulce 8.14.

Tabulka 8.14: Výsledky srovnávacích poslechových testů pro ženský hlas

Srovnávané varianty		Hodnocení [%]			Test statistické významnosti ($\alpha = 0{,}05$)	
A	B	A = B	A > B	A < B	p	rozdíl stat. významný
ZCCVIT^{opt}	VITBASE	80,81	9,09	10,10	0,5000	ne
ZCCVIT^{opt}	VITPRUNE^{opt}	54,71	30,00	15,29	0,0029	ano

8.3.3 Shrnutí závěrů z hodnocení a poslechových testů

V této kapitole byly nejprve hodnoceny implementované algoritmy podle matematických ukazatelů. Největší potenciál k urychlení procesu výběru jednotek při zachování kvality syntetické řeči měly algoritmy ZCCVIT a VITPRUNE. V případě optimální varianty algoritmu ZCCVIT^{opt} bylo dosaženo zrychlení $S = 556{,}54$ ($S = 438{,}30$ pro ženský hlas). Optimální varianta VITPRUNE^{opt}

s obdobnou hodnotou matematické kvality Q pak urychlila výběr jednotek v míře $S = 563{,}74$ ($S = 449{,}05$ pro ženský hlas).

Oba algoritmy ZCCVIT i VITPRUNE tedy umožňují významně urychlit proces výběr jednotek při syntéze řeči z textu, přičemž výsledky poslechových testů ukazují, že se kvalita generovaných promluv nezhoršila ve srovnání s referenčním algoritmem VITBASE.

Algoritmus VITPRUNE lze jednodušeji implementovat a jeho parametry lze velmi snadno měnit například i dynamicky v závislosti na vytížení systému, na němž je syntéza provozována. Nevýhodou proti algoritmu ZCCVIT je vyšší nárůst hodnot ukazatelů zhoršení matematické kvality Q a CD (hodnoty jsou přibližně dvojnásobné), což souvisí u vyšších rychlostí s vysokou mírou prořezání kandidátů jednotek.

Naproti tomu je implementace algoritmu ZCCVIT složitější, ale matematické ukazatele kvality mají stabilnější průběh než u algoritmu VITPRUNE. Velkou výhodou je to, že i při konfiguraci parametrů dávající vysoké zrychlení se pracuje s většinou kandidátů díky tomu, že jsou součástí ZCC řetězců. Kromě toho je algoritmus velmi efektivní v případech, kdy lze požadovanou promluvu pokrýt celým ZCC řetězcem, protože vyhodnocení je okamžité a potřebný výpočetní výkon minimální.

Kapitola 9

Modifikace zaměřené na odstranění artefaktů

Kvalita výstupu systému konkatenační syntézy řeči závisí v první řadě na tom, jak dobře byl namluven a technicky zpracován řečový korpus a jak dobře byla připravena jeho popisná data (hranice řečových segmentů, fonetický popis – transkripce, prozodická anotace apod.). Pokud se jedná o syntézu řeči výběrem jednotek, pak je dalším klíčovým faktorem ovlivňujícím kvalitu i způsob výběru vhodných kandidátů řečových jednotek požadované promluvy. Případné chyby nebo nepřesnosti v některé z uvedených podúloh syntézy řeči způsobují uvnitř promluv vznik rušivých a nepřirozených zvuků, které se nazývají *artefakty* a které zhoršují kvalitu řeči.

Při hodnocení kvality poslechovými testy může pak i jeden jediný výskyt výrazně slyšitelného artefaktu způsobit velmi negativní hodnocení celé promluvy. Navíc platí, že ani výběrem posloupnosti kandidátů s globálně minimální kumulativní cenou C nelze vždy předejít jejich vzniku. V těchto případech se plně projevuje to, že matematické ukazatele kvality nezaručují současně i nejlepší dosažitelnou kvalitu a přirozenost promluvy v měřítku percepčním.

U optimalizovaných algoritmů výběru jednotek, které jsou založeny na ZCC řetězcích, bylo předpokládáno zvýšené riziko vzniku artefaktů v místech napojení delších řetězců. Při experimentech s optimalizací procesu výběru jednotek se skutečně u některých promluv artefakty objevily, ovšem v podobném měřítku se vyskytovaly i u algoritmů zaručujících nalezení cesty grafem kandidátů s globálně minimální kumulativní cenou C (VITBASE, VITOPT). Proto byla opatření proti vzniku artefaktů experimentálně zkoumána u všech

dříve popsaných algoritmů a nikoliv pouze u těch, jež pracují se ZCC řetězci.

9.1 Typy artefaktů

Příčiny vzniku artefaktů, které byly zaznamenány při interních poslechových testech v průběhu experimentů, jsou následující:

Chyba řečníka vzniká při nahrávání řečového korpusu a je způsobena různými „selháními“ (hlasovými indispozicemi) řečníka, jako jsou špatná výslovnost, nevhodná intonace, parazitické zvuky, míchání různých mluvních stylů, různé tempo řeči apod. Pokud nejsou takové chyby odhaleny při anotaci řečového korpusu, pak se při použití nesprávně namluvených jednotek projeví odpovídající chyba i ve výsledné promluvě TTS systému, a to bez ohledu na techniku jejich výběru.

Chyba při segmentaci vzniká nejčastěji při automatické detekci hranic stanovených řečových jednotek. Použitím manuální segmentace se riziko snižuje, nicméně jde o časově velmi náročnou činnost použitelnou pouze u menších řečových korpusů a i osoba provádějící tuto činnost může způsobit chybu. Pokud jsou některé realizace řečových jednotek vytvořené ze segmentů s chybně určenými hranicemi, pak může jejich použití vést ke vzniku velmi nepřirozených projevů ve vygenerované promluvě způsobených chybějícími nebo naopak přebývajícími částmi zvukového signálu.

Výrazně odlišná frekvence F_0 řetězených segmentů způsobuje nejčastěji pozorované artefakty. Pokud se frekvence základního hlasivkového tónu na konci prvního segmentu a na začátku následujícího segmentu významně liší, pak se tato situace projeví jako nepřirozené zakolísání frekvence hlasu a je hodnoceno posluchači velmi negativně. Obdobný problém vzniká i v případě, že se liší tendence průběhu F_0, a to i v případech, kdy se statický rozdíl na hranicích segmentů liší minimálně.

Odlišná délka trvání realizace řečové jednotky je chybou vznikající tehdy, když se do posloupnosti optimální sekvence kandidátů dostane segment, jehož délka je významně delší nebo kratší než by bylo na dané pozici

vhodné. Tento typ chyby se projevuje velmi nepřirozeným vyzněním promluvy, kdy posluchač například uprostřed věty slyší nevhodně dlouhou hlásku, kterou by spíše svými vlastnostmi očekával na konci.

První dva typy uvedených problémů způsobené chybou řečníka nebo nepřesností v segmentaci řečového korpusu nelze na úrovni algoritmu výběru jednotek nijak ovlivnit, ale lze říci, že pokud je přípravná fáze vývoje TTS systému (tj. zejména nahrání řečového korpusu a jeho segmentace) provedena kvalitně, pak je četnost takových chyb minimální.

Větším problémem jsou však artefakty způsobené odlišnou frekvencí F_0 řetězených jednotek nebo nevhodnou délkou trvání, jejichž projevy jsou v promluvách častější a posluchači je hodnotí velmi negativně. Nejčastěji jsou jejich výskyty důsledkem nedokonalosti hodnoticích funkcí ceny řetězení C^{c} a ceny cíle C^{t}. Například v aktuální verzi systému ARTIC se s frekvencemi segmentů sice při výpočtu ceny řetězení C^{c} pracuje, ale jde pouze o statický rozdíl na hranicích řetězených segmentů se slabší váhou ve výsledném hodnocení. Podobně se při výpočtu ceny cíle C^{t} nezohledňuje délka trvání, ale stejně jako u intonace, je modelována implicitně na základě umístění řečového segmentu v původní promluvě v řečovém korpusu. V konečném důsledku se může stát, že nahrazení některého kandidáta optimální sekvence jiným kandidátem s horším hodnocením C^{t} nebo C^{c} může vést k odstranění artefaktu a výrazně tím zlepšit percepční kvalitu promluvy (např. díky lepší návaznosti F_0).

Následující text je zaměřen na techniky, jak vzniku artefaktů předcházet na úrovni výběru jednotek, přičemž některé z nich mohou současně vést k urychlení výběru jednotek (příkladem může být omezení kandidátů podle frekvence základního hlasivkového tónu F_0 popsané v oddíle 6.1.5).

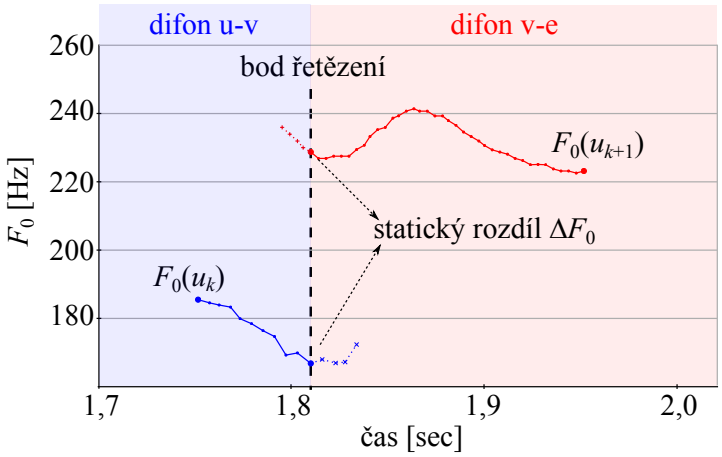
9.2 Omezení řetězení kandidátů s významně odlišnou frekvencí F_0

V kapitole 6 této práce bylo uvedeno množství způsobů jak optimalizovat proces výběru jednotek. V oddíle 6.1.5 je pak popsána technika založená na předvýběru kandidátů pro řetězení pomocí porovnání výšky frekvence základního hlasivkového tónu F_0. Optimalizace uvedená v publikaci Conkie & Syrdal, 2011 je zaměřena na urychlení hledání nejlepšího předchůdce kandidáta

v rámci Viterbiova algoritmu tím, že se kandidáti předchozího kroku omezí pouze na segmenty, jejichž frekvence F_0 se neliší o více než parametrem určený rozsah. V publikaci je uváděno rozmezí $\Delta F_0 = \pm 50\ Hz$, což je relativně vysoká hodnota, ovšem v této disertační práci bylo cílem urychlení a nikoliv zamezení vzniku artefaktů. Optimalizace tedy vycházela z myšlenky omezit počet předchozích kandidátů, kteří by se s vysokou pravděpodobností stejně nestali nejlepším předchůdcem. Na rozdíl od uvedené publikace, kde se používala průměrná hodnota F_0[1)], byly v této práci používány frekvence na začátku a na konci řečových segmentů.

OMEZENÍ ŘETĚZENÍ DLE STATICKÉHO ROZDÍLU F_0

Experimenty s omezením řetězení kandidátů podle F_0 byly prováděny na variantách algoritmů, pro které bylo dosaženo nejlepších výsledků, tj. VITPRUNE$^{\text{opt}}$ a ZCCVIT$^{\text{opt}}$.



Obrázek 9.1: Znázornění statického rozdílu frekvence F_0 v místě řetězení dvou řečových segmentů.

Nejprve byly algoritmy upraveny tak, že se před výpočtem ceny řetězení mezi sousedními kandidáty u_k a u_{k+1} provedl test dle následujícího kritéria:

$$\left|F_{0k}^{\text{e}} - F_{0k+1}^{\text{b}}\right| \leq \Delta F_0{}^{\max}, \tag{9.1}$$

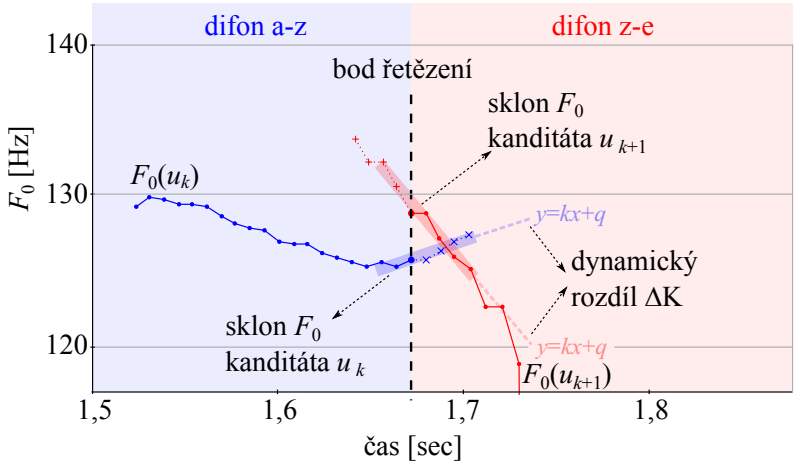
[1)]Lze se domnívat, že autoři použili střední hodnotu frekvence F_0 z toho důvodu, že neměli pro řečové segmenty k dispozici okamžiky hlasivkových pulsů a určení přesné frekvence na hranicích segmentů by tak bylo složité.

tj. zda rozdíl statických hodnot koncové frekvence $F_{0_k}^{\mathrm{e}}$ předcházejícího kandidáta a počáteční frekvence $F_{0_{k+1}}^{\mathrm{b}}$ následujícího kandidáta (viz obrázek 9.1) nepřesahuje limit stanovený nově zavedeným parametrem $\Delta {F_0}^{\max}$. Pokud kritérium nebylo splněno, pak se přeskočilo vyhodnocení ceny řetězení C^{c} a technicky bylo spojení odstraněno z grafu kandidátů.

OMEZENÍ ŘETĚZENÍ DLE TENDENCE PRŮBĚHU F_0

Pro oba algoritmy byla opakovaně prováděna syntéza promluv, u nichž se vyskytovaly artefakty způsobené nesouladem frekvence F_0. Postupně byl snižován limit $\Delta {F_0}^{\max}$ a bylo sledováno, při jaké hodnotě jsou problémová zřetězení ještě slyšitelná. U velmi nízkých hodnot ($\Delta {F_0}^{\max} < 10$ Hz) sice došlo k odstranění největších problémů, nicméně řada artefaktů přetrvala.

Příkladem může být zřetelně slyšitelný artefakt, který vznikl v místě napojení dvou segmentů, jejichž frekvence F_0 v místě spojení byla velmi blízká, a to $\Delta F_0 = 3{,}39$ Hz (viz obrázek 9.2). Při podrobném zkoumání tohoto artefaktu byla nalezena příčina, která spočívala v rozdílné tendenci frekvence příslušných řečových segmentů v původní nahrávce. Přímky, které aproximují průběh frekvence v bodě řetězení, se zcela rozcházely, přičemž první z nich měla výrazně klesající a druhá naopak vzestupnou tendenci. V důsledku toho zde byl slyšitelný artefakt, a protože se přímky protkaly téměř přesně v místě napojení segmentů, nezabránila jeho vzniku ani nízká hodnota parametru $\Delta {F_0}^{\max}$.



Obrázek 9.2: Znázornění rozdílu sklonu frekvence F_0 v místě řetězení dvou řečových segmentů.

Protože se ukázalo, že test na statický rozdíl frekvencí F_0 řetězených řečových segmentů je nedostatečný, byly provedeny další experimenty, při nichž se porovnávaly sklony průběhu frekvence F_0, resp. sklony přímek aproximujících hodnoty F_0 v časových okamžicích hlasivkových pulsů. Bylo provedeno několik experimentů a na základě nich byla nakonec použita aproximace na konci, resp. začátku řečových segmentů vypočítaná z množiny dvou předcházejících a čtyř následujících hlasivkových pulzů z původní nahrávky v řečovém korpusu. Takto zvolený rozsah byl stanoven na základě pozorování vizualizací průběhu frekvence F_0 u různých náhodně vybraných promluv. Použití většího počtu následujících hlasivkových pulzů vychází z úvahy, že průběh frekvence F_0 za bodem řetězení má větší vliv na vznik případného artefaktu. Nabízí se možnost využít i širší okolí bodu řetězení, tím by ale již mohlo docházet k velkému zkreslení aproximace, zejména u velmi krátkých řečových segmentů[2)].

Samotný test napojitelnosti dvou řečových segmentů probíhal tak, že nejprve byla vždy provedena aproximace přímek $y^{\mathrm{e}}_{F_0}(u_k)$ a $y^{\mathrm{b}}_{F_0}(u_{k+1})$ reprezentujících průběh frekvence F_0 v bodě řetězení dvou sousedních kandidátů u_k a u_{k+1}. Pomocí lineární regrese byly tak vypočteny koeficienty odpovídajících dvou přímek:

$$y^{\mathrm{e}}_{F_0}(u_k) = a^{\mathrm{e}}_k x + b^{\mathrm{e}}_k, \tag{9.2}$$

$$y^{\mathrm{b}}_{F_0}(u_{k+1}) = a^{\mathrm{b}}_{k+1} x + b^{\mathrm{b}}_{k+1}, \tag{9.3}$$

kde a^{e}_k a a^{b}_{k+1} reprezentují sklon frekvence F_0 na konci kandidáta u_k a na začátku kandidáta u_{k+1}. Koeficienty b^{e}_k a b^{b}_{k+1} jsou pak aproximovanými hodnotami frekvence F_0 přímo v bodě řetězení.

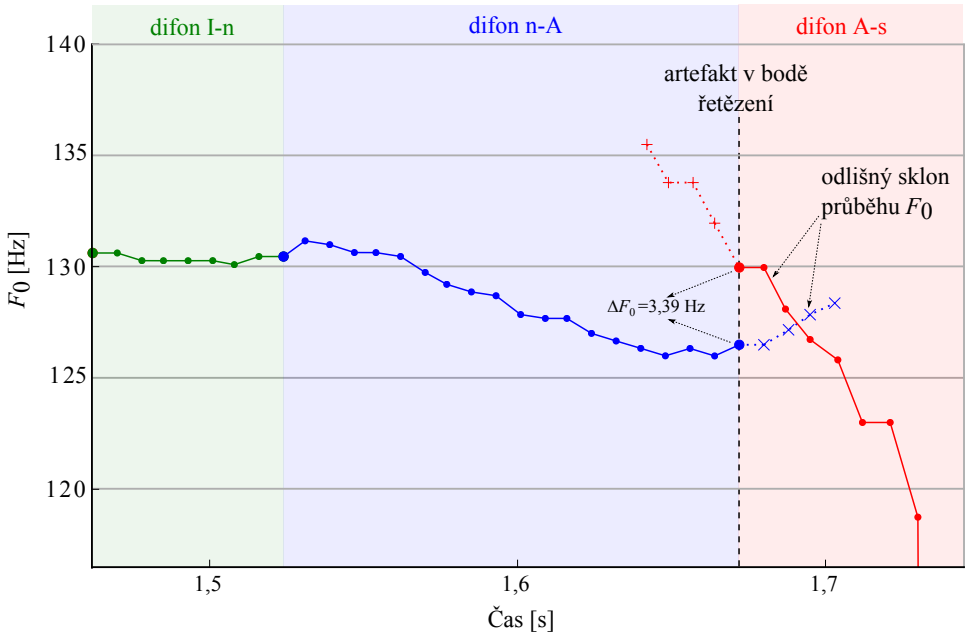
Původní test na statický rozdíl byl v algoritmech ponechán a parametr $\Delta {F_0}^{\max}$ zůstal zachován, protože hodnoty $F_0{}^e_k$ a $F_0{}^b_{k+1}$ jsou přesnější než aproximované hodnoty b^{e}_k and b^{b}_{k+1}. Test však byl rozšířen o nový parametr $\Delta a^{\max}$, kterým se stanovil maximální rozdíl hodnot a^{e}_k a a^{b}_{k+1}, tj. rozdíl sklonu

[2)] Otázka volby počtu hlasivkových pulzů pro aproximaci průběhu frekvence F_0 nebo i obecně způsob posouzení vhodnosti spojení dvou řečových segmentů je úlohou, která jde svým rozsahem nad rámec této práce. Výsledky sice ukázaly zlepšení v kvalitě generované promluvy i možnosti jak artefakty odstranit, nicméně je potřeba na ně nahlížet tak, že jde o prvotní experimenty, na něž bude možné do budoucna navázat.

průběhu frekvence F_0 v místě řetězení dvou řečových segmentů. Při testu napojitelnosti dvou kandidátů je důležitá i kontrola na to, zda se shoduje směr přímek, tj. zda obě přímky mají vzestupnou nebo klesající tendenci, protože pokud se liší, tak i při minimálním rozdílu absolutní hodnoty koeficientů a_k^{e} a a_{k+1}^{b} může dojít ke vzniku artefaktu. Kontrola na soulad směru je prováděna testem na shodu znaménka, protože pokud je hodnota a kladné číslo, pak frekvence v daném úseku stoupá a záporná hodnota odpovídá klesající tendenci. Původní kritérium 9.1 bylo tedy doplněno o následující dvě podmínky:

$$\mathrm{sgn}\left[a_k^{\mathrm{e}}\right] = \mathrm{sgn}[a_{k+1}^{\mathrm{b}}], \tag{9.4}$$

$$\left|a_k^{\mathrm{e}} - a_{k+1}^{\mathrm{b}}\right| \leq \Delta a^{\max}. \tag{9.5}$$



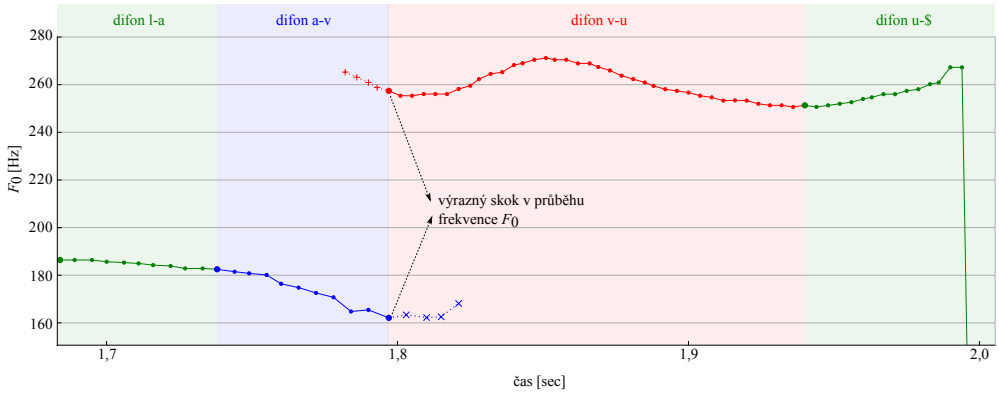
Obrázek 9.3: Příklad slyšitelného artefaktu v syntetizované promluvě. Hodnota statického rozdílu frekvencí F_0 je sice nízká, avšak sklon průběhu F_0 se v bodě řetězení významně liší.

V průběhu dalších experimentů byl postupně snižován limit na rozdíl koeficientů a do doby, než byly odstraněny sledované artefakty. Nakonec byla pro další experimenty stanovena hodnota parametru $\Delta a^{\max} = 20$ (para-

metr určující limit statického rozdílu frekvence F_0 byl nastaven na hodnotu $\Delta {F_0}^{\max} = 30$).

PROBLÉM S INTONACÍ U ZJIŠŤOVACÍCH OTÁZEK

Součástí množiny testovacích promluv bylo i několik zjišťovacích otázek, jejichž intonace má na konci výrazně vzestupnou tendenci. Při experimentech byl u takových promluv pozorován výskyt problému, kdy při použití kontrol na rozdíl a sklon průběhu frekvence F_0 docházelo k výběru řečových segmentů s nevhodnou intonací. Byla tedy provedena detailní analýza a srovnání se správně znějící verzí promluvy bez omezení dle průběhu frekvence F_0. Při ní bylo zjištěno, že v promluvě s intonací odpovídající zjišťovací otázce vzrostla na konci věty v jednom okamžiku frekvence F_0 o 95 Hz (viz obrázek 9.4), aniž by byl slyšet nějaký artefakt. Okamžik téměř skokového zvýšení frekvence odpovídal právě místu promluvy, kdy se zvýšila zcela přirozeně intonace u zjišťovací otázky.



Obrázek 9.4: Příklad průběhu frekvence F_0 na konci zjišťovací otázky. Promluva zněla `Neměla bych si oholit hlavu?` a výrazný skok frekvence F_0 je patrný na přechodu mezi difony `a-v` a `v-u`.

Na základě tohoto poznatku byla kontrola na soulad průběhu frekvence F_0 u zjišťovacích otázek vypnuta na konci promluvy resp. neprováděla se u realizací jednotek zařazených do koncového prozodému promluvy, jenž byl v práci Romportl *et al.*, 2004 značen kódem P2. Takové opatření je sice velmi jednoduché a zvyšuje riziko vzniku jiných nežádoucích artefaktů, nicméně v kontextu této práce účel splnila. Vhodnější implementace takové úpravy by vyžadovala detailnější analýzu a statistické vyhodnocení většího množství otázek je úloha překračující rámec této práce.

9.2.1 Výkonnost algoritmů při omezení F_0

U obou algoritmů VITPRUNE i ZCCVIT došlo zavedením opatření k zamezení vzniku artefaktů v důsledku odlišného průběhu frekvence F_0 k mírnému zhoršení kvality v matematickém měřítku. Z pohledu percepční kvality a tedy přirozenosti promluv, došlo naopak k výraznému zlepšení díky odstranění artefaktů, což bylo doloženo i poslechovými testy.

U algoritmu VITPRUNE je důsledkem předvýběru předchozích kandidátů dle průběhu F_0 snížení počtu možných spojení při hledání nejlepšího předchůdce a tím došlo k urychlení procesu výběru jednotek. V testovací verzi algoritmu došlo k navýšení výpočetní náročnosti, protože sklony F_0 byly počítány v době syntézy. U produkčního systému se však předpokládá doplnění potřebných koeficientů do databáze řečových segmentů, takže výpočty proběhnou pouze v přípravné fázi systému a nebudou nijak zatěžovat samotný výběr jednotek.

V případě algoritmu ZCCVIT se rychlost výběru téměř nezměnila (nebo se snížila), protože vnitřní cyklus hledání nejlepšího předchůdce se opakuje, dokud není skutečně zpracován počet předcházejících ZCC řetězců stanovený parametrem *CHAINDEEP*. Na druhou stranu je algoritmus ZCCVIT robustnější a odolnější při dalším snižování hranice stanovené parametrem $\Delta a^{\max}$. I při velmi nízkých hodnotách nedocházelo k problémům s nalezením cesty grafem kandidátů, což u algoritmu VITPRUNE neplatilo (u jedné z testovacích promluv nebyla cesta grafem nalezena a nebyla ve výsledcích započítána). Při praktickém nasazení by tak bylo nutné navrhnout řešení takových situací (back–off), tedy zřejmě ponechat v grafu kandidátů některé segmenty, i když nesplňují podmínku na soulad průběhu frekvence F_0. Dalším možným řešením je doplnit hodnocení statického rozdílu i odlišnosti tendence průběhu frekvence F_0 do výpočtu ceny řetězení (s vyšší váhou), takže by nevhodná spojení řečových segmentů mohla být významně penalizována, ovšem nikoliv zcela vynechána[3)].

[3)] Tento způsob nebyl testován, protože hlavním zaměřením této práce je urychlení procesu výběru jednotek. Navíc by zahrnutí míry souladu frekvence F_0 do ceny řetězení C^{c} nevedlo ke zrychlení, protože by nedošlo ke snížení počtu přípustných spojení mezi kandidáty.

9.3 Prořezání kandidátů s velmi odlišnou délkou trvání

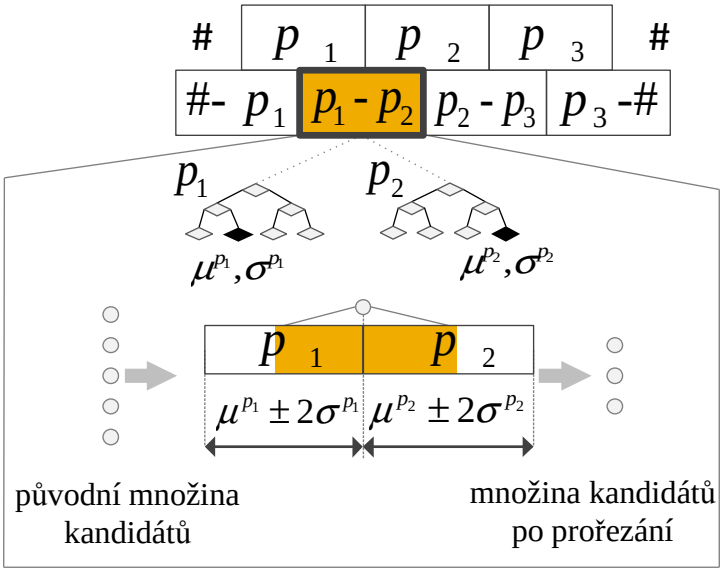
Další artefakty, které mají v případě svého výskytu významný vliv na hodnocení přirozenosti promluv, vznikají v důsledku nedokonalosti hodnoticí funkce ceny cíle C^{t}. Pokud funkce C^{t} nepracuje s žádným modelem trvání jednotek v promluvách, pak může dojít k situacím, kdy se v cestě s globálně minimální cenou použije řečový segment s nevhodnou délkou svého trvání.

Nejjednodušší metodou, jak omezit použití segmentů s nepřirozenou délkou trvání v daném místě promluvy, je vypočítat průměrnou délku trvání kandidátů (případně medián) a na všech pozicích grafu pak prořezat realizace jednotek, jejichž délka se od střední hodnoty významně liší. Při experimentech a poslechových testech se sice podařilo odstranit většinu artefaktů, ale jednoznačně se ukázalo, že výsledná řeč se stává velmi monotónní a „umělá“. Důvodem je to, že střední hodnoty délek kandidátů si jsou na všech pozicích velmi podobné, čímž se vytrácí intonace a dynamika generovaných promluv. Je totiž známo, že délka trvání konkrétní hlásky není vždy stejná a liší se v závislosti na její pozici v promluvě (typické je např. prodloužení hlásky na konci promluvy). Odlišná může být i délka hlásky ve vazbě na fonetický kontext.

Snížit riziko vzniku artefaktů tohoto typu a zároveň zachovat přirozenost promluv lze tak pouze zavedením modelu trvání jednotek, pomocí něj pro každou jednotku v promluvě provést odhad a pak znevýhodnit nebo zcela vyřadit z výběru kandidáty, jejichž délka se od odhadu významně liší. Další experimenty v této práci vycházely z publikace Romportl & Kala, 2007 a byl při nich využíván model trvání založený na klasifikačních a regresních rozhodovacích stromech (zkr. CART[4)]). Pro každý foném (nikoliv difon[5)]) byl vytvořen samostatný rozhodovací strom, který byl natrénován na promluvách původního řečového korpusu. K natrénování modelu byly všechny výskyty fonémů opatřeny celkem 172 lingvistickými příznaky, které zahrnují např. třídu jednotky, znělost resp. neznělost nebo velké množství údajů o okolí a umístění jednotky v promluvě.

[4)] Z angl. *Classification and Regression Tree*), detaily viz např. Breiman *et al.*, 1984

[5)] Rozhodovací stromy byly vztaženy k fonémům. Pokud by se jednalo o difony, pak by stromů bylo obrovské množství a zároveň by vznikl problém s nedostatečnou velikostí trénovací množiny.



Obrázek 9.5: Schéma využití modelu trvání jednotek k prořezání realizací s významně odlišnou délkou ve srovnání s odhadem.

Samotný mechanismus odstranění kandidátů s nevhodnou délkou byl implementován tak, že pro každou jednotku u_k (tj. difón p_1-p_2 sestávající z částí fonémů p_1 a p_2) na pozici k byla nejprve odhadnuta délka trvání $\hat{d}_k$ pomocí regresních stromů pro fonémy p_1 a p_2. Výsledný list rozhodovacího stromu pak určil střední hodnotu $\mu_k^{p_j}$ a směrodatnou odchylku $\sigma_k^{p_j}$ všech realizací jednotek p_j, které do listu náležely.

Následně se z množiny kandidátů jednotek u_k odstranily difóny vytvořené z fonémů p_j, pokud délka trvání $d_k^{p_j}$ nesplňovala následující kritérium:

$$\mu_k^{p_j} - 2\sigma_k^{p_j} < d_k^{p_j} < \mu_k^{p_j} + 2\sigma_k^{p_j}, \quad j = 1, 2. \tag{9.6}$$

Po prořezání tak byly v množině kandidátů ponechány pouze realizace jednotek, jejichž délka se na dané pozici nelišila o více jak dvojnásobek směrodatné odchylky od průměrné hodnoty odhadnuté pomocí modelu trvání. Protože model trvání pracuje s fonémy, musely být odhadnuté hodnoty přepočítány difony (délka difonu je určena součtem polovin délek sousedních fonémů, které difon tvoří – viz schéma na obrázku 9.5).

Po doplnění algoritmu výběru jednotek o prořezání grafu kandidátů s využitím rozhodovacích stromů bylo zřetelně dosaženo odstranění artefaktů

způsobených nevhodnou délkou řečových segmentů, přičemž dynamika výsledných promluv zůstala zachována.

Z hlediska výpočetní náročnosti bylo sice nutné vyhodnotit všechny fonémy v promluvě celkem 172 lingvistických příznaků a musel být navíc proveden i samotný odhad rozhodovacím stromem, nicméně z celkového pohledu jde o zanedbatelný výpočetní čas. Dále pak je nutné uchovávat jako součástí databáze řečových i samotné rozhodovací stromy, ale nárůst velikosti je také zanedbatelný.

9.4 Hodnocení modifikací k odstranění artefaktů

Experimenty ukázaly, že zejména zamezení řetězení řečových segmentů s odlišnou tendencí průběhu frekvence F_0 má zásadní vliv na výslednou percepční kvalitu syntetické řeči. Sice zde pak dochází ke zhoršení matematických ukazatelů kvality, nicméně díky odstranění artefaktů je přirozenost promluv mnohem vyšší a při srovnání s výsledky základního Viterbiova algoritmu byly algoritmy hodnoceny dokonce výrazně lépe. Obdobně jako u poslechových testů popsaných v oddíle 8.3.2 byly k hodnocení použity varianty algoritmů ZCCVIT$^{\text{opt}}$ a VITPRUNE$^{\text{opt}}$ doplněné o uvedené modifikace k odstranění artefaktů (dále budou varianty značeny ZCCVIT$^{\text{art}}$ a VITPRUNE$^{\text{art}}$). U varianty algoritmu ZCCVIT$^{\text{art}}$ bylo dosaženo urychlení s hodnotou $S = 639{,}70$ a v případě algoritmu VITPRUNE$^{\text{art}}$ dokonce $S = 1569{,}08$. Při jejich přímém porovnání poslechovými testy byly výsledky srovnatelné, ovšem z matematického hlediska a z pohledu konstrukce obou algoritmů je zjevné, že algoritmus ZCCVIT je robustnější a dokáže najít optimální cestu grafem i při velmi agresivním nastavení limitů při kontrole na soulad frekvence základního hlasivkového tónu F_0 v místě řetězení dvou segmentů z různých promluv řečového korpusu. Velkou výhodou algoritmu ZCCVIT je to, že i při nastavení parametrů vedoucích k výraznému urychlení se prohledává celý původní graf kandidátů. Naproti tomu rychlosti měřené u varianty algoritmu VITPRUNE$^{\text{art}}$ jsou vyšší, ale při doplnění kontrol dle frekvence F_0 již docházelo k případům, kdy nebyla nalezena žádná cesta grafem a syntéza příslušné promluvy nebyla dokončena. To je způsobeno tím, že vysokých rychlostí u algoritmu VITPRUNE je dosaženo na úkor extrémního prořezání grafu kandidátů podle ceny cíle, a to až

na počty kandidátů v řádu jednotek pro každou pozici promluvy. Pokud se navíc nastavil algoritmus ZCCVIT tak, aby bylo matematické zhoršení kvality shodné se stejně rychlou verzí algoritmu VITPRUNE, pak bylo dosaženo rychlosti $S = 3737{,}59$. Algoritmem VITPRUNE již nebylo možné takové rychlosti ani dosáhnout, protože kontrola na soulad frekvence F_0 způsobila, že se nedokončila syntéza ani z jedné promluv. Algoritmus ZCCVIT lze tak považovat za stabilnější a odolnější vůči opatřením, která vedou k výraznému zlepšení přirozenosti syntetické řeči produkované TTS systémem. Matematické hodnocení výkonu variant algoritmů ZCCVIT$^{\text{art}}$ a VITPRUNE$^{\text{art}}$ obsahující techniky minimalizující vznik artefaktů je uvedeno v tabulce 9.1 pro mužský hlas a 9.2 pro ženský hlas.

Zlepšení kvality generovaných promluv bylo prokázáno i pomocí poslechových testů, jichž se účastnilo celkem 15 posluchačů. Byly porovnány promluvy vygenerované pomocí základního Viterbiova algoritmu, tj. s globálně minimální kumulativní cenou, s promluvami získanými variantami algoritmů VITPRUNE$^{\text{art}}$ a ZCCVIT$^{\text{art}}$ s modifikacemi k odstranění artefaktů. Přestože při matematickém srovnání byly oba algoritmy horší, tak při poslechových testech je posluchači preferovali před promluvami s globálně minimální kumulativní cenou. Detailní výsledky poslechových testů jsou uvedeny v tabulkách 9.3 pro mužský hlas a v tabulce 9.4 pro ženský hlas[6)]. V poslechovém testu pro mužský hlas byla sice varianta VITPRUNE$^{\text{art}}$ hodnocena nepatrně lépe než varianta ZCCVIT$^{\text{art}}$, ale stejné srovnání u ženského hlasu dopadlo s opačným výsledkem a navíc i s větším rozdílem preferencí. Výstupy algoritmů byly tedy při poslechových testech srovnatelné, nicméně stále platí závěry předchozích experimentů, že výkon algoritmu ZCCVIT je stabilnější, protože i při vyšší míře urychlení vyhodnocuje mnohem větší počet kandidátů jednotek dané promluvy. Důležitým závěrem experimentů s modifikacemi k odstranění artefaktů je to, že se jedná účinný nástroj, jak zvýšit kvalitu syntetické řeči a že jde o směr, který si zaslouží další a podrobnější výzkum.

[6)] Do sady poslechových testů pro algoritmy s modifikacemi proti vzniku artefaktů byl u mužského hlasu zařazen navíc test k porovnání varianty VITPRUNE$^{\text{art}}$ s referenčním algoritmem VITBASE. Důvodem je to, že důraz u tohoto poslechového testu je na samotných modifikacích algoritmů. Nicméně u ženského hlasu již nebyla tato kombinace z kapacitních důvodů testována.

Tabulka 9.1: Výkon variant algoritmů s modifikacemi k odstranění artefaktů pro mužský hlas

Algoritmus	Míra urychlení (S)	Matematická míra zhoršení kvality (Q) [‰]	Míra fragmentace (CD) [%]
VITBASE	1,00	0,00	29,39
ZCCVIT$^{\text{art}}$	616,11	205,52	31,09
VITPRUNE$^{\text{art}}$	1610,63	252,17	37,92

Tabulka 9.2: Výkon variant algoritmů s modifikacemi k odstranění artefaktů pro ženský hlas

Algoritmus	Míra urychlení (S)	Matematická míra zhoršení kvality (Q) [‰]	Míra fragmentace (CD) [%]
VITBASE	1,00	0,00	27,20
ZCCVIT$^{\text{art}}$	458,57	171,82	31,20
VITPRUNE$^{\text{art}}$	1156,45	310,52	42,15

Tabulka 9.3: Výsledky srovnávacích poslechových testů algoritmů s modifikacemi proti vzniku artefaktů pro mužský hlas

Srovnávané varianty		Hodnocení [%]			Test statistické významnosti ($\alpha = 0{,}05$)	
A	B	A = B	A > B	A < B	p	rozdíl stat. významný
ZCCVIT$^{\text{art}}$	VITBASE	46,67	38,67	14,67	< 0,0001	ano
VITPRUNE$^{\text{art}}$	VITBASE	51,56	34,22	14,22	< 0,0001	ano
ZCCVIT$^{\text{art}}$	VITPRUNE$^{\text{art}}$	57,78	18,22	24,00	0,1090	ne

Tabulka 9.4: Výsledky srovnávacích poslechových testů algoritmů s modifikacemi proti vzniku artefaktů pro ženský hlas

Srovnávané varianty		Hodnocení [%]			Test statistické významnosti ($\alpha = 0{,}05$)	
A	B	A = B	A > B	A < B	p	rozdíl stat. významný
ZCCVIT$^{\text{art}}$	VITBASE	45,38	30,77	23,85	0,1712	ne
ZCCVIT$^{\text{art}}$	VITPRUNE$^{\text{art}}$	68,82	20,00	11,18	0,0267	ano

Kapitola 10

Hodnocení algoritmů nad redukovanou databází řečových segmentů

Všechny předchozí experimenty byly v souladu s hlavním zaměřením této práce testovány a hodnoceny při použití rozsáhlých databází řečových segmentů o velikosti více jak 600 MB, které čítaly 670 757 segmentů pro mužský hlas a 656 406 segmentů pro hlas ženský. Pokud by se měla syntéza řeči provozovat s minimálními nároky na paměť a úložiště (např. na mobilním telefonu), pak by bylo vhodné použít méně obsáhlé databáze. Experiment k ověření chování a výkonu algoritmů při omezeném počtu dostupných kandidátů řečových jednotek byl prováděn na redukované databázi o velikosti přibližně 100 MB. Redukce původních úplných databází řečových segmentů byla provedena metodou publikovanou v práci Hanzlíček *et al.*, 2013, jež pracuje na základě statistického měření nejčastěji používaných realizací jednotek. Výsledné databáze obsahovaly v případě mužského hlasu 99 175 řečových segmentů a v případě ženského hlasu 93 969 řečových segmentů.

10.1 Výkon algoritmů

Podobně jako v oddíle 8.3 věnovanému hodnocení implementovaných algoritmů byly nad redukovanými databázemi testovány algoritmy ZCCVIT a VITPRUNE.

ALGORITMUS VITBASE

Kromě hodnocení těchto dvou algoritmů jako zástupců optimalizovaných metod výběru jednotek je zajímavá i změna ve výkonu referenčního algoritmu VITBASE. Primárním účelem redukce databáze je snížení paměťových nároků, ovšem v důsledku nižšího počtu dostupných realizací řečových segmentů také dochází logicky i k navýšení rychlosti výběru optimální sekvence kandidátů i u samotného algoritmu VITBASE. Hodnocení výkonu algoritmů (tabulka 10.1 pro mužský hlas a tabulka 10.2 pro ženský hlas) bylo pro úplnost doplněno o míru zrychlení výběru jednotek při srovnání s referenčním algoritmem nad úplnou, tj. neredukovanou, databází řečových segmentů. Proces výběru jednotek byl při použití redukované databáze u algoritmu VITBASE rychlejší 22,78× u mužského a 23,46× ženského hlasu. Dalším zjištěním při srovnání promluv vygenerovaných algoritmem VITBASE nad redukovanou databází je to, že ani u mužského ani u ženského hlasu nebyly žádné dvě promluvy tvořeny naprosto stejnými řečovými segmenty jako při použití úplné databáze.

ALGORITMY ZCCVIT A VITPRUNE

Nejprve byla provedena syntéza testovací množiny promluv pomocí variant algoritmů ZCCVIT$^{\text{opt}}$ a VITPRUNE$^{\text{opt}}$, jež byly popsány v oddíle 8.3.2.1. Míra urychlení výběru jednotek byla nad redukovanou databází nižší, což se dalo očekávat, protože pokud je k dispozici nižší počet kandidátů jednotek, pak i prostor pro zrychlení není takový jako u úplné databáze řečových segmentů. Míra urychlení u varianty ZCCVIT$^{\text{opt}}$ činila pro mužský hlas $S = 189{,}09$ a $S = 229{,}39$ pro ženský hlas. Varianta VITPRUNE$^{\text{opt}}$ pak dávala nižší míry zrychlení, a to $S = 88{,}22$ pro mužský hlas a $S = 44{,}83$ pro ženský hlas. Zrychlení obou variant algoritmů je tedy u redukované databáze nižší, ale stále dosahuje akceptovatelných hodnot, zejména proto, že celková míra urychlení vůči referenčnímu algoritmu VITBASE nad úplnou databází dosahuje hodnot, které odpovídající agresivním verzím algoritmů ZCCVIT$^{\text{fst}}$ a VITPRUNE$^{\text{fst}}$ (viz tabulky 8.11 a 8.13). Hodnota ukazatele matematická míry zhoršení kvality Q byla také vyšší než v případě úplné databáze, nicméně u optimálních variant nebyly také vysoké. Výsledné hodnoty míry fragmentace *CD* jsou také konzistentní s úplnou databází, protože u varianty algoritmu ZCCVIT$^{\text{opt}}$ byla fragmentace nižší než u referenčního algoritmu VITBASE a varianta VITPRUNE$^{\text{opt}}$ měla tuto hodnotu nepatrně vyšší. Úplný výčet výkonnostních ukazatelů pro všechny varianty jsou uvedeny v tabulkách 10.1

(mužský hlas) a 10.2 (ženský hlas).

Tabulka 10.1: Výkon variant algoritmů nad redukovanou databází řečových segmentů pro mužský hlas

Algoritmus	Míra urychlení (S)	Matematická míra zhoršení kvality (Q) [‰]	Míra fragmentace (CD) [%]	Míra urychlení vůči VITBASE nad úplnou databází (S)
VITBASE	1,00	0,00	37,54	22,78
ZCCVIT$^{\text{opt}\,1)}$	189,09	59,78	35,67	4307,47
VITPRUNE$^{\text{opt}\,1)}$	83,22	33,74	40,54	1895,75
ZCCVIT$^{\text{art}\,1)}$	127,82	164,61	38,71	2911,74
VITPRUNE$^{\text{art}\,1)}$	302,80	289,93	49,34	6897,78

Tabulka 10.2: Výkon variant algoritmů nad redukovanou databází řečových segmentů pro ženský hlas

Algoritmus	Míra urychlení (S)	Matematická míra zhoršení kvality (Q) [‰]	Míra fragmentace (CD) [%]	Míra urychlení vůči VITBASE nad úplnou databází (S)
VITBASE	1,00	0,00	36,57	23,46
ZCCVIT$^{\text{opt}\,1)}$	229,39	58,46	34,30	5831,49
VITPRUNE$^{\text{opt}\,1)}$	44,83	79,49	43,82	1051,71
ZCCVIT$^{\text{art}\,1)}$	141,24	207,33	39,30	3313,49
VITPRUNE$^{\text{art}\,1)}$	126,94	228,65	46,69	2978,01

Další část analýzy výkonu algoritmů nad redukovanou databází byla věnována variantám algoritmů s modifikacemi minimalizujícími vznik artefaktů. U varianty algoritmu VITPRUNE$^{\text{art}}$ byl výkon proti variantě VITPRUNE$^{\text{opt}}$ výrazně vyšší, což se dalo předpokládat v důsledku omezení možných dvojic kandidátů, u nichž byl statický rozdíl i sklon průběhu frekvence F_0 v definovaném limitu. Opačné chování bylo pozorováno u algoritmu ZCCVIT$^{\text{art}}$, u něhož byla rychlost u obou hlasů nižší než u optimální varianty ZCCVIT$^{\text{opt}}$. Zpomalení lze vysvětlit tím, že kvůli omezení pomocí frekvence F_0 se také snížil počet přímo navazujících ZCC splňující omezující podmínky spojení krajních kandidátů jednotek. Proto bylo nutné vyhodnotit více spojení mezi

[1)] Parametry algoritmu byly shodné jako při experimentech nad úplnou databází.

vzdálenějšími ZCC řetězci a bylo tak nutné častěji použít algoritmus BEFS k dohledání cest složených z jednotlivých kandidátů. Na druhou stranu vykazovala varianta ZCCVIT$^{\text{art}}$ lepší hodnoty u kvalitativních ukazatelů než variant VITPRUNE$^{\text{art}}$. Detailní hodnoty opět viz tabulky 10.1 (mužský hlas) a 10.2 (ženský hlas).

10.2 Poslechové testy

K vyhodnocení toho, zda nedošlo použitím algoritmů ZCCVIT a VITPRUNE v kombinaci s redukovanou databází řečových segmentů ke zhoršení kvality syntetické řeči, byly použity poslechové testy. Postup jejich realizace byl shodný s předchozími testy. Jednalo se opět o testy typu *CCR* a hodnocení se zúčastnilo celkem 11 posluchačů.

Z výsledků uvedených v tabulkách 10.3 (mužský hlas) a 10.4 (ženský hlas) vychází, že u optimální varianty ZCCVIT$^{\text{opt}}$ se kvalita nezhoršila. Naopak u mužského hlasu byla dokonce preference výstupu algoritmu ZCCVIT$^{\text{opt}}$ nepatrně vyšší. Je také zřejmé, že u většiny odpovědí byla kvalita při srovnání s referenčním algoritmem VITBASE považována za stejně dobrou.

V případě variant s doplněnými modifikacemi proti vzniku artefaktů preferovali posluchači u mužského i ženského hlasu promluvy generované algoritmem ZCCVIT$^{\text{art}}$. Opět se potvrzuje zásadní vliv hladkého průběhu frekvence F_0 i vhodného trvání vybraných řečových segmentů na výslednou kvalitu syntetické řeči.

10.3 Shrnutí výsledků

Na základě experimentů s algoritmy ZCCVIT a VITPRUNE nad redukovanou databází řečových segmentů lze říci, že oba algoritmy se osvědčily i při nižším počtu možných kandidátů jednotek pro generované promluvy. Nepotvrdila se tak původní obava, že při použití algoritmu ZCCVIT dojde ke zhoršení kvality, protože v grafu kandidátů nebude dostatek vhodných ZCC řetězců.

Zajímavou statistikou, jež potvrzuje vhodnost užití algoritmů i pro redukovanou databázi, je počet promluv, které se zcela shodovaly s výběrem pomocí referenčního algoritmu VITBASE. Například u mužského hlasu se jak

u algoritmu ZCCVIT$^{\text{opt}}$ tak i VITPRUNE$^{\text{opt}}$ podařilo nalézt tuto shodnou posloupnost u dvou promluv testovací množiny.

Tabulka 10.3: Výsledky srovnávacích poslechových testů algoritmů nad redukovanou databází řečových segmentů pro mužský hlas

Srovnávané varianty		Hodnocení [%]			Test statistické významnosti ($\alpha = 0{,}05$)	
A	B	A = B	A > B	A < B	p	rozdíl stat. významný
ZCCVIT$^{\text{opt}}$	VITBASE	77,92	12,99	9,09	0,3145	ne
ZCCVIT$^{\text{art}}$	VITBASE	67,53	16,88	15,58	0,5000	ne

Tabulka 10.4: Výsledky srovnávacích poslechových testů algoritmů nad redukovanou databází řečových segmentů pro mužský hlas

Srovnávané varianty		Hodnocení [%]			Test statistické významnosti ($\alpha = 0{,}05$)	
A	B	A = B	A > B	A < B	p	rozdíl stat. významný
ZCCVIT$^{\text{opt}}$	VITBASE	73,08	12,50	14,42	0,2858	ne
ZCCVIT$^{\text{art}}$	VITBASE	61,54	20,19	18,27	0 4373	ne

Kapitola 11

Závěr

Tato disertační práce se zabývá optimalizací vnitřních procesů výběru jednotek u TTS systémů na bázi syntézy řeči výběrem jednotek, přičemž hlavním zaměřením je urychlení vyhledání optimální sekvence řečových segmentů. Hlavním problémem této úlohy je nutnost vyhledat cestu s minimální cenou napříč velmi rozsáhlým grafem tvořeným všemi dostupnými realizacemi řečových jednotek, což vede na obrovské množství kombinací a nutnosti vypočítat i odpovídající množství ohodnocení hran grafu čítající až desítky miliónů operací.

V rámci práce bylo provedeno velké množství experimentů s algoritmy, jež lze obecně rozdělit do dvou skupin.

První skupinu tvoří algoritmy založené na běžně používaném Viterbiovu algoritmu a jeho modifikacích. Nejlepších výsledků v této skupině bylo dosaženo algoritmem VITPRUNE, který doplňuje původní Viterbiův algoritmus o vhodné prořezávání grafu kandidátů společně s dalšími optimalizacemi. Tím bylo u testovaného mužského hlasu dosaženo urychlení $S = 8{,}35$ (pro ženský hlas byla hodnota $S = 7{,}12$) při nastavení parametrů, jež s minimálním rizikem vybere cestu s globálně minimální kumulativní cenou.

Do druhé skupiny pak patří algoritmy, které vycházejí z původní myšlenky využít k optimalizaci výběru jednotek ZCC řetězce, tj. úseky tvořené kandidáty s nulovou cenou jejich spojení. Postupně byla vyzkoušena řada algoritmů, které kombinovaly různé techniky, a nejpříznivější poměr mezi mírou urychlení a kvalitou výsledné promluvy byl nalezen u algoritmu značeného kódem ZCCVIT. V případě mužského hlasu byla dosažena míra urychlení $S = 556{,}54$ a u ženského hlasu pak $S = 438{,}30$. Jeho konfigurace s nejlepším hodnocením

matematické kvality dokázala vybrat shodné cesty grafem kandidátů jako u referenčního algoritmu VITBASE v 7 z 20 promluv a zbývající promluvy se od optimální sekvence lišily minimálně. Přestože tedy žádná kombinace hodnot parametrů nevedla k výběru cest s globálně minimální kumulativní cenou pro úplně všechny promluvy testovací množiny, byly rozdíly kvality v matematickém měřítku nepatrné a poslechovými testy bylo ukázáno, že kvalita je zcela srovnatelná s výstupem původně používaného algoritmu VITBASE.

Další část experimentů byla věnována opatřením, která by snížila riziko vzniku artefaktů, tj. nepřirozených jevů ve vygenerovaných promluvách. Technikám vedoucím k zamezení vzniku artefaktů bylo v rámci práce poskytnuto méně prostoru než optimalizacím zaměřeným na urychlení výběru jednotek, protože se jedná o rozsáhlé téma, které přesahuje rámec kladených cílů. Nicméně se u prvotních experimentů zřetelně ukazuje, že i za pomoci poměrně jednoduchých opatření lze dosáhnout výrazného zlepšení kvality výstupu syntetizéru.

11.1 Souhrn z pohledu cílů práce

Hlavní cíle práce byly vydefinovány v kapitole č. 5 a jejich naplnění lze shrnout následujícím způsobem:

Analýza stávajících možností a technik optimalizace výběru jednotek. Popisu a rozboru dostupných publikací k tématu optimalizace výběru jednotek v úloze syntézy řeči z textu je věnována celá kapitola 6. Je zde uvedeno množství optimalizací nejčastěji používaného Viterbiova algoritmu i další techniky, jež se zabývají např. předvýběrem kandidátů jednotek pomocí shlukování nebo využitím různých typů mezipamětí snižujících výpočetní zátěž v procesu výběru jednotek.

Experimentální ověření optimalizací základního Viterbiova algoritmu. V rámci práce byla implementována řada algoritmů založených na Viterbiovu algoritmu a jeho modifikacích, z nichž část pocházela z existujících publikací, ale byly použity i některé originální postupy. Popis experimentů je uveden v oddíle 8.1 a výkonnost algoritmů včetně jejich hodnocení jsou zahrnuty v oddíle 8.3.

Analýza výskytu řetězců s nulovou cenou řetězení v promluvách vygenerovaných TTS systémem. Před návrhem algoritmů založených na řetězcích s nulovou cenou řetězení byla provedena detailní analýza výskytu a vlastností těchto řetězců. Výsledky úvodní analýzy jsou shrnuty v oddílech 8.2.1 a 8.2.2, které zahrnují i množství statistik a grafů. V průběhu experimentů s algoritmy využívajícími ZCC řetězce vznikla ještě doplňující analýza vlastností ZCC řetězců ve vítězných posloupnostech s globálně minimální kumulativní cenou C získanou referenčním Viterbiovým algoritmem (viz oddíl 8.2.6).

Návrh a experimentální ověření algoritmů založených na řetězcích s nulovou cenou řetězení. Součástí práce byl návrh několika algoritmů, jež byly založeny na původní myšlence využití obecných ZCC řetězců. Jedná se o klíčovou část této práce, při níž bylo navrženo několik algoritmů. Některé z nich se i přes prvotní předpoklady k dosažení vysoké míry urychlení nakonec neosvědčily, nicméně jejich implementace vedla k získání mnoha poznatků o výskytu ZCC řetězců v promluvách a jejich vlastnostech. Na tomto základě byl nakonec vytvořen algoritmus, jenž umožňuje významně snížit výpočetní náročnost procesu výběru, aniž by došlo ke zhoršení kvality výsledné syntetické řeči. Výsledky byly potvrzeny nejen hodnocením pomocí matematických ukazatelů, ale i množstvím poslechových testů.

Analýza výskytu artefaktů ve vygenerovaných promluvách. Návrh a experimentální ověření technik vedoucích k odstranění artefaktů. Protože optimalizace rychlosti algoritmu výběru jednotek velmi úzce souvisí s kvalitou, byla na základě pozorování systematických problémů způsobujících vznik artefaktů (tj. nepřirozených jevů) v syntetické řeči realizována i analýza možného původu těchto problémů. Na základě ní pak byly do algoritmů výběru jednotek doplněny i modifikace minimalizující vznik popsaných typů artefaktů. Prvním opatřením bylo vytvoření modelu trvání pomocí CART stromů a jeho využití k omezení výběru jednotek s nevhodnou délkou trvání. Druhá část opatření se vztahovala k průběhu frekvence F_0 v místech řetězení segmentů, které na sebe přímo nenavazují v původním řečovém korpusu.

Obě modifikace byly zahrnuty do nejslibnějších verzí algoritmů, jejich výkon byl zhodnocen matematickými ukazateli a také byly algoritmy

ověřeny pomocí poslechových testů. Analýza typů artefaktů, postupy k zamezení jejich vzniku a hodnocení modifikovaných algoritmů jsou součástí kapitoly 9.

Ověření výkonnosti algoritmů pro další hlas řečníka (ženský) a pro redukované databáze řečových segmentů. Verze algoritmů s nejlepším hodnocením byly ověřeny i nad druhou databází řečových segmentů vytvořenou segmentací řečového korpusu nahraného ženským hlasem. Hodnocení výsledků pro ženský hlas je uvedeno společně s hodnocením pro mužský hlas v oddíle 8.3, a to včetně výsledků poslechových testů. Stejně tak byl pro ženský hlas ověřen a v oddíle 9 popsán i výkon algoritmů s modifikacemi zabraňujícími vznik artefaktů.

Chování implementovaných algoritmů bylo také testováno na redukované databázi řečových segmentů, která obsahovala přibližně 6× méně realizací řečových jednotek. Popis experimentů i výsledky pro mužský i ženský hlas jsou součástí kapitoly 10.

Srovnání a vyhodnocení všech vytvořených algoritmů a jejich variant. K hodnocení algoritmů výběru jednotek bylo definováno několik ukazatelů, jež umožnily v matematické rovině porovnat míry zrychlení a kvalitu generovaných promluv (definice ukazatelů jsou uvedeny v kapitole 7). Hodnocení pomocí těchto ukazatelů je vždy uvedeno průběžně u popisu konkrétních algoritmů. Pomocí matematických měřítek pak byly vybrány algoritmy s nejlepšími výsledky a ty byly následně podrobeny poslechovým testům. Souhrnné hodnocení algoritmů a jejich vzájemné srovnání je pak uvedeno v oddíle 8.3.

11.2 Závěrečné hodnocení výsledků

Výsledky této disertační práce ukázaly, že vhodnými technikami lze u velmi rozsáhlých databází řečových segmentů výrazně snížit výpočetní náročnost procesu výběru jednotek v úloze syntézy řeči z textu. Nejlepších výsledků bylo dosaženo použitím algoritmů VITPRUNE a ZCCVIT. Algoritmus VITPRUNE je optimalizovaná verze základního Viterbiova algoritmu a jeho výhodou je relativně jednoduchá implementace. Základním kamenem algoritmu ZCCVIT je pak využití řetězců s nulovou cenou řetězení v množině kandidátů jednotek

dané promluvy. Jeho výhodou jsou nižší a stabilnější hodnoty matematických ukazatelů kvality, a také větší robustnost při použití modifikací minimalizujících vznik artefaktů v produkované řeči.

Pokud by se v systému použila velmi omezená databáze řečových segmentů (o velikosti jednotek MB), pak by mohl již nastat problém s nalezením dostatečného počtu ZCC řetězců a v tom případě se jeví jako vhodnější použít algoritmus VITPRUNE. V situacích, kdy je prioritou kvalita generované řeči a je tak nutné používat velmi rozsáhlou databázi řečových segmentů (nebo středně velkou databázi o objemu desítek až stovek MB), pak lze doporučit užití algoritmu ZCCVIT, jenž má větší potenciál dosáhnout vysoké míry urychlení procesu výběru jednotek bez výraznějšího dopadu na kvalitu řeči.

11.3 Návrhy na další práci

Hlavním těžištěm této práce bylo snížení výpočetní náročnosti procesu výběru jednotek a v této oblasti se podařilo za daných podmínek nalézt vhodné algoritmy, jež lze doporučit k nasazení v produkčních verzích TTS systémů. Po vyhodnocení všech provedených experimentů bylo možné i tak nalézt další možné cesty ke zlepšení. Navíc byla část experimentů, jako například modifikace k odstranění artefaktů, řešena jako doplňkové téma, které svou složitostí přesahuje rámec předkládané práce. Návrhy na možnou další práci jsou následující:

- Jednou z možností jak vylepšit výkon algoritmu ZCCVIT by bylo upravit prohledávání z úplného hledání do šířky na prohledávání expanzí nejslibnější cesty. Jednalo by se tedy o obdobu algoritmu BEFS, která by mohla vést k mírnému urychlení celého procesu výběru jednotek.

- Během experimentů se ukázalo, že výsledná kvalita výrazně klesá s výskytem byť jediného artefaktu ve vygenerované promluvě. Při poslechových testech byly varianty algoritmů s opatřeními proti jejich vzniku hodnoceny výrazně lépe než verze bez nich. Dokonce i promluvy vygenerované pomocí základního Viterbiova algoritmu dávající posloupnosti s globálně minimální kumulativní cenou C byly preferovány méně, a to i přes vysoké rozdíly hodnot matematických ukazatelů kvality. Další výzkum by bylo vhodné zaměřit právě na tuto oblast a zejména na zamezení řetězení

řečových segmentů s odlišným průběhem frekvence F_0. Protože se tedy ukázalo, že jde o klíčový problém, nabízí se provést hlubší analýzu a z různých pohledů prozkoumat vztah mezi průběhem frekvence F_0 a vznikem slyšitelných artefaktů. Na jejím základě by pak bylo možné navrhnout účinnější opatření zlepšující kvalitu syntetické řeči.

- Další návrh úzce souvisí s předchozím bodem, nicméně je zaměřen na samotné algoritmy výběru jednotek. Do systému ARTIC by bylo vhodné doplnit jako součást výpočtu ceny cíle i sklon průběhu frekvence F_0 s vysokou váhou vůči ostatním zkoumaným charakteristikám. Místo striktního prořezání nevhodných kombinací by pak měla tato cena u nevhodných dvojic kandidátů vyšší hodnotu, což by sice nemuselo znamenat úplné odstranění artefaktů, ale také by nemohlo dojít k problémům jako například u algoritmu VITPRUNE s vysokou mírou prořezávání, kdy nebyla syntéza u některých promluv vůbec dokončena v důsledku nedostatku vhodných kandidátů.

Příloha A

Množina testovacích promluv

Množina testovacích promluv odpovídá náhodnému výběru textů z novinových článků tak, aby byly zastoupeny i promluvy, u nichž na konci stoupá intonace (zjišťovací otázky, souvětí složené z více klauzí). Jednotlivé klauze v souvětí jsou uváděné samostatně kvůli zpracovávaným statistikám, ale při poslechových testech byla souvětí přehrávána jako celek, aby měl posluchač k dispozici celý kontext a nebyla pro něj promluva matoucí. Pokračující části souvětí jsou uvozeny znakem ↳.

1. Cena města Plzně je oceněním nejen pro mě.
2. S hlasováním o nedůvěře vládě by měla opozice šetřit.
3. Věznice podle něj dnes plní funkci sociálních ústavů.
4. Neměla bych si oholit hlavu?
5. Výpadek japonské jaderné elektrárny Fukušima možná způsobila krysa,
6. ↳ která se dostala k rozvaděči.
7. Já jsem si tuhle otázku kladla.
8. Obchodníci tomu musejí přizpůsobit své strategie komunikace.
9. Kdy si republika připomíná sto padesáté osmé výročí narození zakladatele novodobého státu.
10. Hájek tak pokračuje v kritice Havla a jeho příznivců.

11. Podepsala se na něm lehká viróza,
12. ↳ se kterou bojuje několik dnů.
13. Pozval jsem si pana předsedu Dvořáka na schůzku.
14. Hokejová extraliga zažila ve čtvrtek rekordně dlouhý zápas.
15. Nebojíte se o post jedničky?
16. Bylo to neuvěřitelně náročné.
17. Lidé ke mně chodí utrácet své peníze a doufají,
18. ↳ že tím vydělají mnohem víc.
19. V takovémto souboji si ale vítězství to mužstvo vybojuje.
20. Chcete jim to snad zakázat?

Příloha B

Hodnoty parametrů v experimentech

Tabulka B.1: Hodnoty parametrů v experimentech pro algoritmus VITBEAM

Parametr	Rozsah hodnot
BW	$1, 5, 10, \ldots, 50, 100, 200, 300, \ldots, 1000$

Tabulka B.2: Hodnoty parametrů v experimentech pro algoritmus VITPRUNE

Parametr	Rozsah hodnot
BW	$1, 5, 10, \ldots, 100, 150, 200, \ldots, 1000$
TW	$1, 5, 10, \ldots, 100, 150, 200, \ldots, 1000$
TWR	$0, 5, 10, 20, 30, 40, 50$

Tabulka B.3: Hodnoty parametrů v experimentech pro algoritmus BEFS

Parametr	Rozsah hodnot
$DEEP$	$10, 15, 20, \ldots, 50, 60, 70, \ldots, 2000$
α	0, 0,01, 0,02, 0,05, 0,10, 0,15, 0,20, …, 0,5

Tabulka B.4: Hodnoty parametrů v experimentech pro algoritmus ZCCBEFS

Parametr	Rozsah hodnot
$DEEP$	$10, 15, 20, \ldots, 50, 60, 70, \ldots, 2000$
α	0, 0,01, 0,02, 0,05, 0,10, 0,15, 0,20, …, 0,5

Tabulka B.5: Hodnoty parametrů v experimentech pro algoritmy ZCCFRAME a ZCCFRAME$^{\text{sub}}$

Parametr	Rozsah hodnot
$LEN^{\min}$	2, 3, 4, 5
$TRANS^{\max}$	1, 2, 3, 4, 5
$START^{\max}$	0, 1, 2, 3
$END^{\max}$	0, 1, 2, 3
$CHAINDEEP$	$5, 10, 20, 30, \ldots, 100$
$FRAMECOUNT^{\max}$	1000, 2000, 3000, 4000, 5000
$CANDCOUNT^{\max}$	1000, 2000, 3000, 4000, 5000

Tabulka B.6: Hodnoty parametrů v experimentech pro algoritmus ZCCFRAME$^{\text{prn}}$

Parametr	Rozsah hodnot
$LEN^{\min}$	2, 3, 4, 5
$TRANS^{\max}$	1, 2, 3, 4, 5
$START^{\max}$	0, 1, 2, 3
$END^{\max}$	0, 1, 2, 3
$CHAINDEEP$	$5, 10, 20, 30, \ldots, 100$
$FRAMECOUNT^{\max}$	1000, 2000, 3000, 4000, 5000
$CANDCOUNT^{\max}$	1000, 2000, 3000, 4000, 5000
${}_{zcc}C^{\text{t}}_{\text{avg}}$	0,3
${}_{zcc}C^{\text{t}}_{\max}$	0,33
$GROUPCOUNT^{\max}$	$10, 20, 30, \ldots, 100, 200, 300$

Tabulka B.7: Hodnoty parametrů v experimentech pro algoritmus ZCCVIT

Parametr	Rozsah hodnot
$LEN^{\min}$	2, 3, 4, 5
$TRANS^{\max}$	1, 2, 3, 4, 5
$START^{\max}$	0, 1, 2, 3
$END^{\max}$	0, 1, 2, 3
$CHAINDEEP$	$5, 10, 20, 30, \ldots, 100$
${}_{zcc}C^{\text{t}}_{\text{avg}}$	0,3
${}_{zcc}C^{\text{t}}_{\max}$	0,33
$GROUPCOUNT^{\max}$	$10, 20, 30, \ldots, 100, 200, 300$

Příloha C

Syntetizované promluvy

C.1 Promluvy použité v poslechových testech

Na přiloženém CD jsou ve složce `promluvy_testy` k dispozici syntetizované promluvy ve formátu `wav`. Pro každý test je vytvořena podsložka a jejich výčet včetně popisu je uveden v souboru `promluvy_testy/index.html`.

C.2 Ukázky artefaktů

Na přiloženém CD jsou ve složce `promluvy_artefakty` k dispozici příklady následujících artefaktů v syntetizovaných promluvách:

- příklad artefaktu v důsledku chybného trvání,
- příklad artefaktu vzniklého v důsledku nesouladu sklonu průběhu frekvence F_0.

Příklady jsou ve formátu `wav`. V souboru `promluvy_artefakty/index.html` je pak uložen výčet artefaktů včetně jejich popisu.

C.3 Ostatní ukázky

Na přiloženém CD je ve složce `promluvy_ostatni` uchován příklad promluvy – doplňující otázky, v níž dochází ke konci k výraznému skoku frekvence F_0, ovšem tento rozdíl frekvencí v místě řetězení se neprojevuje jako artefakt. Stejně u ostatních složek, je zde k dispozici soubor `promluvy_ostatni/index.html` s bližším popisem a promluva je ve formátu `wav`.

Shrnutí (česky)

Tato disertační práce se zabývá optimalizací procesu výběru jednotek v konkatenační syntéze řeči, přičemž hlavním zaměřením je urychlení vyhledání optimální sekvence řečových segmentů. Klíčovým problémem této úlohy je nutnost vyhledat cestu s minimální cenou napříč velmi rozsáhlým grafem tvořeným možnými realizacemi řečových jednotek, což vede na obrovské množství kombinací a nutnosti vypočítat i odpovídající množství ohodnocení hran grafu čítající až desítky miliónů operací.

Součástí práce je řada analýz složení promluv vygenerovaných pomocí běžně užívaného Viterbiova algoritmu, jehož nevýhodou jsou velmi vysoké výpočetní nároky. V rámci práce bylo navrženo a testováno množství algoritmů, které lze rozdělit do dvou skupin. První skupinu tvoří algoritmy, jež doplňují Viterbiův algoritmus o optimalizační techniky snižující nezbytný počet vyhodnocení cen cíle. Ve druhé skupině jsou algoritmy založené na původní myšlence využití souvislých řetězců s nulovou cenou řetězení, tj. úseků původní nahrávky z původního řečového korpusu nahraného lidským řečníkem (řetězce jsou značeny zkratkou ZCC – z angl. zero concatenation cost).

Výsledkem experimentů jsou dva srovnatelné algoritmy, které umožňují zvýšit velmi významně rychlost procesu výběru jednotek (přibližně 500×) při zachování kvality generované řeči. Kvalita výstupu pro oba algoritmy byla ověřena i pomocí poslechových testů.

Doplňkovým tématem práce byla i analýza vzniku nežádoucích artefaktů způsobených buď výběrem řečového segmentu s nevhodnou délkou na dané pozici promluvy nebo zřetězením dvou kandidátů s odlišným průběhem frekvence základního hlasivkového tónu. V rámci práce byla navržena opatření jak vzniku artefaktů předcházet, čímž bylo dosaženo ještě vyšší kvality syntetické řeči ve srovnání s původním Viterbiovým algoritmem.

Résumé (English)

This thesis addresses the optimization of the unit selection process in a concatenation synthesis and it mainly focuses on speeding-up the search for the optimal speech segments sequence. The key problem of this task is the need to find the minimal cost path through the graph consisted of all available unit candidates. This leads to a huge amount of acceptable combinations, and therefore the need to compute an adequate number of graph edges costs counting up to tens of millions of operations.

The work incorporates the analysis of speech utterances synthesized using common the Viterbi algorithm, which has a major drawback of being computationally demanding. Within the work a number of algorithms were proposed and tested. These algorithms can be divided into two specific groups. The first group is made up of modifications of the original Viterbi scheme, which introduces optimization techniques to decrease the necessary amount of concatenation cost evaluations. Algorithms in the second group are based on the novel idea of using continuous chains of speech segments, which correspond to larger chunks from the original speech corpora. These chains do not require compute concatenation cost between containing speech segments, as it always has a zero value (therefore, they are denoted as ZCC – zero concatenation cost).

The research resulted in two comparable algorithms, both of which significantly increased the speed of the unit selection process (approx. 500×), while the quality of the produced synthetic speech was maintained. The quality of the TTS system output was also evaluated and verified by listening tests.

The additional topic of this work was the analysis of unwanted artifacts. These unwanted artifacts are caused by either selecting the speech segment which has an inappropriate length, or by concatenating two candidates of which the fundamental frequency has a different tendency. During the work,

methods to prevent the causing of unwanted artifacts were designed and thus the quality of speech was improved in comparison with the original Viterbi algorithm.

The additional topic to this work was the analysis of unwanted artfeacts, which are caused by selecting the speech segment with an inappropriate length or by concatenating two candidates which fundamental frequency has a different tendency. During the work a methods to prevent the causing of artefacts were designed and thus the quality of the speech was increased in comparison with the original Viterbi algorithm.

Resümee (Deutsch)

Diese Disertationsarbeit befasst sich mit der Optimierung des Prozesses der Auswahl der Einheiten in der konkatenierten (zusamengefügten) Sprachsynthese, wobei die Hauptrichtung die Beschleunigung des Suchens der optimalen Sequenz der Sprachsegmente ist. Das Schlüsselproblem dieser Aufgabe ist die Notwendigkeit, den Weg mit dem minimalen Wert quer durch einen sehr umfangreichen Graph herauszufinden, der durch die möglichen Realisierungen der Spracheinheiten gebildet wird. Dies führt zur riesigen Menge der Kombinationen und zur Notwendigkeit die entsprechende Menge der Auswertung der Kanten des Graphes auszurechnen, die bis zehn Millionen von Operationen zählen.

Der Bestandteil der Arbeit ist eine Reihe von den Analysen der Struktur der Ansprachen, die mit Hilfe von dem üblich benutzten Viterbis Algorithmus generiert sind, dessen Nachteil sehr hohe Berechnungsansprüche sind. Im Rahmen der Arbeit wurde eine Menge von Algorithmen entworfen und getestet, die in zwei Gruppen geteilt werden können. Die erste Gruppe bilden die Algorithmen, die den Viterbis Algorhitmus um solche optimalisierenden Techniken ergänzen, die die nötige Anzahl der Auswertung der Zielwerte vermindern. In der zweiten Gruppe sind die Algorithmen, die auf dem ursprünglichen Gedanke der Nutzung der zusammenhängenden (kohärenten) Ketten mit der Nullwertkette gegründet sind, das ist die Nutzung der Abschnitte der ursprünglichen Tonaufnahme aus dem ursprünglichen Sprachkorpus, die vom menschlichen Redner (die Ketten sind bezeichnet mit der Abkürzung ZCC – aus dem Englischen: zero conatenation cost) aufgenommenen wurden.

Das Resultat der Experimente sind zwei vegleichbare Algorithmen, die sehr bedeutsam die Geschwindigkeit des Auswahlprozesses der Einheiten beim Erhalten der Qualität der generierten Sprache zu erhöhen ermöglichen (ungefähr 500 mal). Das Ergänzungsthema der Arbeit war sie Analyse der

Entstehung der unerwünschten Artefakte, die entweder durch die Auswahl des Sprachsegments mit der unpassenden Länge auf der gegebenen Position der Ansprache oder durch die Verbindung der Kettung von zwei Kandidaten mit dem unterschiedlichen Verlauf der Frequenz des elementaren Stimmtones verursacht werden. Im Rahmen der Arbeit wurden die Maßnahmen vorgeschlagen, wie der Entstehung der Artefakte vorzubeugen, wodurch noch höhere Qualität der synthetischen Sprache imVergleich mit dem originalen Viterbis Algorithmus erreicht wurde.

Literatura

Allen, J., Hunnicutt, M. S. & Klatt, D. (1987). *From Text to Speech: The MITalk System*. Cambridge: Cambridge University Press.

Bellegarda, J. R. (2008). Unit-Centric Feature Mapping for Inventory Pruning in Unit Selection Text-to-Speech Synthesis. In: *Proceedings of Audio, Speech, and Language Processing, IEEE Transactions*. pp. 74–82, pp. 74–82.

Benesty, J., Sondhi, M. M. & Huang, Y. (eds.) (2008). *Springer Handbook of Speech Processing*. Dordrecht: Springer-Verlag.

Beutnagel, M., Mohri, M. & Riley, M. (1999). Rapid Unit Selection from a Large Speech Corpus for Concatenative Speech Synthesis. In: *Proceedings of 6th European Conference on Speech Communication and Technology (EUROSPEECH 1999)*, pp. 607–610. Budapest, Hungary.

Bjørkan, I., Svendsen, T. & Farner, S. (2005). Comparing spectral distance measures for join cost optimization in concatenative speech synthesis. In: *INTERSPEECH*, pp. 2577–2580. ISCA.

Black, A. W. (2002). Perfect Synthesis For All Of The People All Of The Time. In: *Proceedings of IEEE 2002 Workshop on Speech Synthesis*. Santa Monica, USA.

Black, A. W. & Taylor, P. (1997). Automatically Clustering Similar Units For Unit Selection In Speech Synthesis. In: *Proceedings of 5th European Conference on Speech Communication and Technology (EUROSPEECH 1997)*, pp. 601–604. Rhodes, Greece.

Blouin, C., Rosec, O., Bagshaw, P. & D'Alessandro, C. (2002). Concatenation cost calculation and optimisation for unit selection in TTS.

In: *Proceedings of IEEE 2002 Workshop on Speech Synthesis*, pp. 231 – 234. Santa Monica, USA.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984). *Classification and Regression Trees*. Pacific Groove, CA., USA: Wadsworth & Brooks.

Čepko, J., Talafová, R. & Vrabec, J. (2008). Indexing join costs for faster unit selection synthesis. In: *Proceedings of 15th International Conference on Systems, Signals and Image Processing (IWSSIP 2008)*, pp. 503–506. Bratislava, Slovak Republic.

Coker, C. H. (1968). Speech synthesis with a parametric articulatory model. In: *Proceedings of Speech Symposium*, pp. A–4. Kyoto, Japan.

Conkie, A., Beutnagel, M., Syrdal, A. K. & Brown, P. (2000). Preselection of candidate units in a unit selection-based text-to-speech synthesis system. In: *Proceedings of 6th International Conference on Spoken Language Processing (ICSLP 2000)*, vol. 3, pp. 314–317. Beijing, China.

Conkie, A. & Syrdal, A. K. (2011). Using F0 to constrain the unit selection Viterbi network. In: *Proceedings of 36th International Conference on Accoustics, Speech, and Signal Processing (ICASSP 2011)*, pp. 5376–5379. Prague, Czech Republic.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. vol. 1. pp. 269–271, pp. 269–271.

Dutoit, T. (2008). Statistical Parametric Synrthesis. In: Benesty *et al.*, 2008, chap. 21.4, pp. 450–453.

Erdem, C., Beck, F., Hirschfeld, D., Hoege, H. & Hoffmann, R. (2002). Robust unit selection based on syllable prosody parameters. In: *Proceedings of IEEE 2002 Workshop on Speech Synthesis*, pp. 159–162. Santa Monica, USA.

Grůber, M. (2008). Syntéza expresivní řeči. Odborná práce ke státní doktroské zkoušce.

Grůber, M. (2012). Syntéza expresivní reči s využitím dialogových aktů k popisu expresivity. Disertační práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd.

Grůber, M. & Tihelka, D. (2010). Expressive Speech Synthesis for Czech Limited Domain Dialogue System - Basic Experiments. In: *Proceedings of 10th International Conference on Signal Processing (ICSP 2010)*, vol. 1, pp. 561–564. Beijing, China: Institute of Electrical and Electronics Engineers, Inc.

Guo, Q., Wang, B. & Katae, N. (2008). Speech Database Compacted for an Embedded Mandarin TTS System. In: *Proceedings of 6th International Symposium on Chinese Spoken Language Processing (ISCSLP 2008)*, pp. 1–4. Kunming, China.

Hamza, W. & Donovan, R. (2002). Data-driven segment preselection in the IBM trainable speech synthesis system. In: *Proceedings of 7th International Conference on Spoken Language Processing (ICSLP 2002)*, pp. 2609–2612. Denver, Colorado, USA.

Hansakunbuntheung, C., Rugchatjaroen, A. & Wutiwiwatchai, C. (2005). Space reduction of speech corpus based on quality perception for unit selection speech synthesis. In: *Proceedings of Sixth Symposium on Natural Language Processing (SNLP 2006)*, pp. 127–132.

Hanzlíček, Z., Matoušek, J. & Tihelka, D. (2013). Experiments on Reducing Footprint of Unit Selection TTS System. In: *Proceedings of 16th International Conference on Text, Speech and Dialogue (TSD 2013)*, pp. 249–256. Pilsen, Czech Republic.

Hill, D., Manzara, L. & Taube-Schock, C.-R. (1995). Real-time articulatory speech-synthesis-by-rules. In: *Proceedings of 14th Annual International Voice Technologies Conf (AVIOS 1995)*, pp. 27–44. San Jose.

Horák, P. (2002). Automatic Speech Segmentation Based on Alignment with a Text-to-Speech System. In: *Improvements in Speech Synthesis* (Keller E., Bailly G., Monaghan A., Terken J., Huckwale M., ed.), pp. 328–338. Berlin: John Wiley & Sons, Ltd.

Hunt, A. & Black, A. W. (1996). Unit Selection in Concatenative Speech Synhesis System Using A Large Speech Database. In: *Proceedings of 1996 IEEE International Conference on Accoustics, Speech, and Signal Processing (ICASSP 1996)*, pp. 373–376. Atlanta, USA.

Isogai, M. & Mizuno, H. (2010). Speech Database Reduction Method for Corpus-Based TTS System. In: *Proceedings of International Conference on Spoken Language Processing (INTERSPEECH 2010)*, pp. 158–161. Makuhari, Japan.

ITU-T Recomendation P.800 (1996). Methods for objective and subjective assessment of quality.

Jinyoung, K. & Joohun, L. (2007). Modified LBG Clustering Algorithms for Small Unit Inventory in Corpus-based TTS System. http://www.fit.hcmus.edu.vn/elib/handle/123456789/166722.

Kala, J. & Matoušek, J. (2007). Duration Modelling in Czech TTS System using Classification and Regression Trees. In: *Proceedings of 17th Czech-German Workshop on Speech Processing*, pp. 154–159. Prague, Czech Republic.

Kala, J. & Matoušek, J. (2014a). Quality Improvements of Zero-Concatenation-Cost Chain Based Unit Selection. In: *Proceedings of 16th International Conference on Speech and Computer (SPECOM2014)*. Novi Sad, Serbia (accepted).

Kala, J. & Matoušek, J. (2014b). Very Fast Unit Selection using Viterbi Search with Zero-Concatenation-Cost Chains. In: *Proceedings of 39th International Conference on Accoustics, Speech, and Signal Processing (ICASSP 2014)*, pp. 2588–2592. Florence, Italy.

Kala, J., Matoušek, J. & Tihelka, D. (2009). Reduction of Computing Cost in Unit Selection TTS System with Large Corpus. In: *Proceedings of 19th Czech-German Workshop on Speech Processing*, pp. 85–91. Prague, Czech Republic.

Kirkpatrick, B., O'Brien, D. & Scaife, R. (2006). Feature extraction for spectral continuity measures in concatenative speech synthesis. In: *INTERSPEECH*. ISCA.

Kominek, J. & Black, A. W. (2004). Impact of Durational Outlier Removal from Unit Selection Catalogs. In: *Proceedings of 5th ISCA Speech Synthesis Workshop (SSW5)*, pp. 155–60. Pittsburgh, USA.

Krul, A., Damnati, G., Yvon, F., Boidin, C. & Moudenc, T. (2007). Adaptive database reduction for domain specific speech synthesis. In: *Proceedings of 6th ISCA Speech Synthesis Workshop (SSW6)*, pp. 217–222. Bonn, Germany.

Kumar, R. & Kishore, S. P. (2004). Automatic pruning of unit selection speech databases for synthesis without loss of naturalness. In: *Proceedings of 8th International Conference on Spoken Language Processing (INTERSPEECH 2004)*, pp. 1377–1380. Jeju Island, Korea.

Lakkavalli, V. R., Arulmozhi, P. & Ramakrishnan, A. G. (2010). Continuity metric for unit selection based text-to-speech synthesis. In: *Proceedings of International Conference on Signal Processing and Communications (SPCOM 2010)*, pp. 1 – 5. Bangalore, India.

Lee, C. H., Jung, S. K. & Kang, H. G. (2007). Applying a Speaker-Dependent Speech Compression Technique to Concatenative TTS Synthesizers. In: *Proceedings of 2008 IEEE International Conference on Accoustics, Speech, and Signal Processing (ICASSP-IEEE 2008)*, vol. 15, pp. 632–640.

Legát, M. (2012). Concatenation Cost in Unit Selection Speech Synthesis. Ph.D. dissertation. University of West Bohemia in Pilsen, Faculty of Applied Sciences.

Ling, Z. H., Hu, Y., Shuang, Z. W. & Wang, R. H. (2002). Decision tree based unit pre-selection in Mandarin Chinese synthesis. In: *Proceedings of International Symposium on Chinese Spoken Language Processing (ISCSLP 2002)*.

Mahalanobis, P. C. (1936). On the generalised distance in statistics. In: *Proceedings National Institute of Science*, vol. 2, pp. 49 – 55. India.

Markel, J. D. & Gray, A. H. (1976). *Linear Prediction of Speech*. New York, USA: Springer-Verlag.

Matoušek, J. & Romportl, J. (2008). Automatic Pitch-Synchronous Phonetic Segmentation. In: *INTERSPEECH 2008, proceedings of 9th Annual Conference of International Speech Communication Association*, pp. 1626–1629. Brisbane, Australia: ISCA.

Matoušek, J., Tihelka, D. & Hanzlíček, Z. (2009). Reducing Footprint of Unit Selection TTS System by Excluding Utterances from Source Speech Corpus. In: *Proceedings of 19th Czech-German Workshop on Speech Processing*, pp. 92–98. Prague, Czech Republic.

Matoušek, J., Tihelka, D. & Psutka, J. (2003). Automatic segmentation for Czech concatenative speech synthesis using statistical approach with boundary-specific correction. In: *Eurospeech 2003 - Interspeech, proceedings of the 8th European Conference on Speech Communication and Technology*, pp. 301–304. Geneva, Switzerland: ISCA.

Matoušek, J., Tihelka, D. & Romportl, J. (2006). Current state of Czech text-to-speech system ARTIC. In: *Proceedings of 9th International Conference on Text, Speech and Dialogue (TSD 2006)*, vol. 4188 of *Lecture Notes in Artificial Intelligence*, pp. 439–446. Berlin, Heidelberg: Springer.

Matoušek, J. (2008). Počítačová syntéza řeči. Habilitační práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd.

Nishizawa, N. & Kawai, H. (2007). A Preselection Method Based on Cost Degradation from the Optimal Sequence for Concatenative Speech Synthesis. In: *Proceedings of 8th Annual Conference of the International Speech Communication Association (INTERSPEECH 2007)*, pp. 2869–2872. Antwerp, Belgium.

Nishizawa, N. & Kawai, H. (2008). Unit database pruning based on the cost degradation criterion for concatenative speech synthesis. In: *Proceedings of 2008 IEEE International Conference on Accoustics, Speech, and Signal Processing (ICASSP-IEEE 2008)*, pp. 3969–3972. Las Vegas, USA.

Palková, Z. (1994). *Fonetika a fonologie češtiny.* Praha: Karolinum.

Psutka, J., Müller, L., Matoušek, J. & Radová, V. (2006). *Mluvíme s počítačem česky.* Prague, Czech Republic: Academia.

Romportl, J. (2006). Structural Data-Driven Prosody Model for TTS Synthesis. In: *Proceedings of the Speech Prosody 2006 Conference*, pp. 549–552. Dresden, Germany: TUDpress.

Romportl, J. & Kala, J. (2007). Prosody Modelling in Czech Text-to-Speech Synthesis. In: *Proceedings of the 6th ISCA Workshop on Speech*

Synthesis, pp. 200–205. Bonn: Rheinische Friedrich-Wilhelms-Universität Bonn.

Romportl, J., Matoušek, J. & Tihelka, D. (2004). Advanced Prosody Modelling. In: *Proceedings of the 7th International Conference on Text, Speech and Dialogue (TSD 2004)*, vol. 3206 of *Lecture Notes in Artificial Intelligence*, pp. 441–447. Berlin, Heidelberg: Springer.

Rubin, P. E., Baer, T. & Mermelstein, P. (1981). An articulatory synthesizer for perceptual research. In: *Journal of the Acoustical Society of America*, vol. 70, p. 321–328.

Sakai, S., Kawahara, T. & Nakamura, S. (2008). Admissible stopping in viterbi beam search for unit selection in concatenative speech synthesis. In: *Proceedings of 2008 IEEE International Conference on Accoustics, Speech, and Signal Processing (ICASSP-IEEE 2008)*, p. 4613–4616. Las Vegas, USA.

Schröder, M. (2007). Introduction to Unit Selection synthesis. presentation for seminar - Unit selection and HMM-based speech synthesis.

Šiška, Z. (2001). *Fonetika a fonologie*. Olomouc.

Spanias, A., Painter, T. & Atti, V. (2007). *Audio Signal Processing And Coding*. Hoboken, New Jersey: John Wiley & Sons, Inc.

Tarjan, R. E. & Yao, A. C.-C. (1979). Storing a Sparse Table. In: *Communications of the ACM*, vol. 22. pp. 606–611, pp. 606–611.

Taylor, P. (2009). *Text-to-speech synthesis*. New York, USA: Cambridge University Press.

Taylor, P. & Black, A. W. (1999). Speech synthesis by phonological structure matching. In: *Proceedings of 6th European Conference on Speech Communication and Technology (EUROSPEECH 1999)*. Budapest, Hungary.

Tihelka, D. (2007). Corpus-based Approach to Unit Selection Speech Unit Inventory Reduction in ARTIC TTS. In: *Proceedings of 17th Czech-German Workshop on Speech Processing*, pp. 160–167. Prague, Czech Republic.

Tihelka, D., Kala, J. & Matoušek, J. (2010). Enhancements of Viterbi Search for Fast Unit Selection Synthesis. In: *Proceedings of International*

Conference on Spoken Language Processing (INTERSPEECH 2010), pp. 174–177. Makuhari, Japan.

Tihelka, D. & Matoušek, J. (2004). The design of Czech language formal listening tests for the evaluation of TTS systems. In: *Proceedings of 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pp. 2099–2102. Lisbon, Portugal: European Language Resources Association.

Tihelka, D., Matoušek, J. & Kala, J. (2007). Quality Deterioration Factors in Unit Selection Speech Synthesis. In: *Proceedings of 10th International Conference on Text, Speech and Dialogue (TSD 2007)*, vol. 4629 of *Lecture Notes in Artificial Intelligence*, pp. 508–515. Pilsen, Czech Republic: Springer-Verlag Berlin Hiedelberg 2007.

Tihelka, D. & Romportl, J. (2008). Statistical Evaluation of Reliability of Large Scale Listening Tests. In: *Proceedings of 9th International Conference on Signal Processing (ICSP 2008)*, pp. 631–636. Beijing, China.

Tokuda, K., Kobayashi, T. & Imai, S. (1995). Speech parameter generation from HMM using dynamic features. In: *Proceedings of 1995 IEEE International Conference on Accoustics, Speech, and Signal Processing (ICASSP 1995)*, pp. 660–663. Detroit, Michigan, USA.

Tsiakoulis, P., Chalamandaris, A., Karabetsos, S. & Raptis, S. (2008). A statistical method for database reduction for embedded unit selection speech synthesis. In: *Proceedings of 2008 IEEE International Conference on Accoustics, Speech, and Signal Processing (ICASSP-IEEE 2008)*, pp. 4601–4604. Las Vegas, USA.

Vepa, J. & King, S. (2003). Kalman-filter based join cost for unit-selection speech synthesis. In: *INTERSPEECH*. ISCA.

Vepa, J., King, S. & Taylor, P. (2002). New objective distance measures for spectral discontinuities in concatenative speech synthesis. In: *Proceedings of IEEE 2002 Workshop on Speech Synthesis*, pp. 223 – 226. Santa Monica, USA.

Yoshimura, T., Tokuda, K., Masuko, T., Kobayashi, T. & Kitamura, T. (1999). Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis. In: *Proceedings of 6th European Conference on*

Speech Communication and Technology (EUROSPEECH 1999), pp. 2347–2350. Budapest, Hungary.

Zen, H., Nose, T., Yamagishi, J., Sako, S., Masuko, T., Black, A. W. & Tokuda, K. (2007). The HMM-based speech synthesis system (HTS) version 2.0. In: *Sixth ISCA Workshop on Speech Synthesis (SSW6)*, pp. 294–299. ISCA.

Zen, H., Tokuda, K. & Black, A. W. (2009). Statistical parametric speech synthesis. vol. 51. Elsevier Science Publishers BV, pp. 1039–1064, pp. 1039–1064.

Zhang, W., Gu, L. & Gao, Y. (2010). Recent improvements of Probability Based Prosody Models for Unit Selection in concatenative Text-to-Speech. In: *Proceedings of 2010 International Conference on Accoustics, Speech, and Signal Processing (ICASSP 2010)*, pp. 3777 – 3780. Taipei, Taiwan.

Seznam publikovaných prací

Kala, J. & Matoušek, J. (2007). Duration Modelling in Czech TTS System using Classification and Regression Trees. In: *Proceedings of 17th Czech-German Workshop on Speech Processing*, pp. 154–159. Prague, Czech Republic.

Kala, J. & Matoušek, J. (2014a). Quality Improvements of Zero-Concatenation-Cost Chain Based Unit Selection. In: *Proceedings of 16th International Conference on Speech and Computer (SPECOM2014)*. Novi Sad, Serbia (accepted).

Kala, J. & Matoušek, J. (2014b). Very Fast Unit Selection using Viterbi Search with Zero-Concatenation-Cost Chains. In: *Proceedings of 39th International Conference on Accoustics, Speech, and Signal Processing (ICASSP 2014)*, pp. 2588–2592. Florence, Italy.

Kala, J., Matoušek, J. & Tihelka, D. (2009). Reduction of Computing Cost in Unit Selection TTS System with Large Corpus. In: *Proceedings of 19th Czech-German Workshop on Speech Processing*, pp. 85–91. Prague, Czech Republic.

Romportl, J. & Kala, J. (2007). Prosody Modelling in Czech Text-to-Speech Synthesis. In: *Proceedings of 6th Speech Synthesis Workshop*, pp. 200–205. Bonn, Germany.

Tihelka, D., Kala, J. & Matoušek, J. (2010a). Enhancements of Viterbi Search for Fast Unit Selection Synthesis. In: *Proceedings of International Conference on Spoken Language Processing (INTERSPEECH'2010)*, pp. 174–177. Makuhari, Japan.

Tihelka, D., Matoušek, J. & Kala, J. (2007). Quality Deterioration Factors in Unit Selection Speech Synthesis. In: *Proceedings of 10th*

International Conference on Text, Speech and Dialogue (TSD'2007), vol. 4629 of *Lecture Notes in Artificial Intelligence*, pp. 508–515. Pilsen, Czech Republic: Springer-Verlag Berlin Hiedelberg 2007.

Tihelka, D., Matoušek, J. & Kala, J. (2010b). ARTIC speaks on Android. In: *Proceedings of 20th Czech-German Workshop on Speech Processing*, vol. 2010, pp. 147–151. Prague, Czech Republic: Institute of Photonics and Electronics AS CR.