Efficient Decoding of High-order Hidden Markov Models

Herman A. Engelbrecht

Dissertation presented for the degree of Doctor of Philosophy at the University of Stellenbosch

Promotor: Prof J.A. du Preez

September 2007

Pectora roborant cultus recti

Declaration

I, the undersigned, hereby declare that the work contained in this dissertation is my own original work, except where stated otherwise.

SIGNATURE DATE

Abstract

Most speech recognition and language identification engines are based on hidden Markov models (HMMs). Higher-order HMMs are known to be more powerful than first-order HMMs, but have not been widely used because of their complexity and computational demands. The main objective of this dissertation was to develop a more time-efficient method of decoding high-order HMMs than the standard Viterbi decoding algorithm currently in use.

We proposed, implemented and evaluated two decoders based on the Forward-Backward Search (FBS) paradigm, which incorporate information obtained from low-order HMMs. The first decoder is based on time-synchronous Viterbi-beam decoding where we wish to base our state pruning on the complete observation sequence. The second decoder is based on time-asynchronous A* search. The choice of heuristic is critical to the A* search algorithms and a novel, task-independent heuristic function is presented. The experimental results show that both these proposed decoders result in more time-efficient decoding of the fully-connected, high-order HMMs that were investigated.

Three significant facts have been uncovered. The first is that conventional forward Viterbi-beam decoding of high-order HMMs is not as computationally expensive as is commonly thought.

The second (and somewhat surprising) fact is that backward decoding of conventional, high-order left-context HMMs is significantly more expensive than the conventional forward decoding. By developing the right-context HMM, we showed that the backward decoding of a mathematically equivalent right-context HMM is as expensive as the forward decoding of the left-context HMM.

The third fact is that the use of information obtained from low-order HMMs significantly reduces the computational expense of decoding high-order HMMs. The comparison of the two new decoders indicate that the FBS-Viterbi-beam decoder is more time-efficient than the A* decoder. The FBS-Viterbi-beam decoder is not only simpler to implement, it also requires less memory than the A* decoder.

We suspect that the broader research community regards the Viterbi-beam algorithm as the most efficient method of decoding HMMs. We hope that the research presented in this dissertation will result in renewed investigation into decoding algorithms that are applicable to high-order HMMs.

Synopsis

Verskuilde Markov-modelle (VMM's) vorm die basis van die meeste spraakherkenning- en taalidentifikasie-stelsels. Dit is bekend dat hoër-orde-VMM's kragtiger is as hul eerste-orde ekwivalente, maar eersgenoemde word oor die algemeen vermy weens hul kompleksiteit en verwerkingsvereistes. Die hoofdoel van hierdie proefskrif was om 'n meer tyd-effektiewe metode te ontwikkel as die Viterbi-dekoderingsalgoritme waarmee VMM's tans algemeen ontsyfer word.

In hierdie verhandeling word twee dekodeerders voorgestel, beide gebaseer op die Vorentoe-Agtertoe-Soektogbeginsel (VAS), en die voorgestelde tegnieke word ook prakties geïmplementeer. Die VAS-beginsel inkorporeer inligting vanuit lae-orde VMM's in die soektog. Die eerste dekodeerder is gebaseer op tydsinkrone Viterbi-bundeldekodering, waarin ons verlang om ons toestandsnoeiïng te grond op die volledig waargenome sekwensie. Die tweede dekodeerder berus op 'n tyd-asinkrone A*-soektog. A*-soekalgoritmes is besonder sensitief vir die keuse van 'n heuristiek, en ons stel vervolgens ook 'n nuwe, taak-onafhanklike heuristiekfunksie voor. Eksperimentele resultate dui aan dat beide dekodeerders die volverbinde, hoër-orde VMM's wat in die ondersoek gebruik is, vinniger kan dekodeerder.

Drie noemenswaardige bevindings is gemaak. Die eerste is dat die gebruiklike voorwaartse Viterbi-bundeldekodering van hoër-orde VMM's nie so berekeningsintensief is as wat algemeen aanvaar word nie.

Die tweede (en ietwat verrassende) bevinding is dat truwaartse dekodering van konvensionele hoër-orde, linkerkonteks-VMM's aansienlik duurder is as die gebruiklike voorwaartse dekodering. Deur die regterkonteks-VMM te ontwikkel, toon ons aan dat die truwaartse dekodering van 'n wiskundig ekwivalente regterkonteks-VMM dieselfde verwerkingskoste het as die voorwaartse dekodering van die linkerkonteks-VMM.

Die derde bevinding is dat die gebruik van inligting wat uit laer-orde-VMM's ontgin word, die berekeningskoste van hoër-orde-VMM-dekodering aansienlik kan verlaag. Die vergelyking van die twee dekodeerders dui aan dat die VAS-Viterbi-bundeldekodeerder nie net eenvoudiger is om te implementeer nie, maar ook minder geheuespasie vereis as die A*-dekodeerder.

Ons vermoed dat die breër navorsingsgemeenskap die Viterbi-bundelalgoritme as die mees effektiewe VMM-dekoderingstegniek beskou. Ons hoop dat die navorsing vervat in hierdie verhandeling sal lei tot hernieude ondersoek van dekoderingsalgoritmes toepaslik tot hoër-orde-VMM's.

Acknowledgements

I would like to express my sincere gratitude to the following people:

- First of all to my wife, Marian, who supported, encouraged me (and had to live with me) during the rain and shine of this journey.
- My promotor, Prof. Johan du Preez, for his unwavering belief that the idea must work and that we must just find the way.
- Dr. Ludwig Schwardt, for the stimulating discussions regarding high-order hidden Markov models, which allowed me to properly define the right-context HMM.
- My good friend, Dr. Gert-Jan van Rooyen, who after seven years and many attempted explanations, still do not understand hidden Markov models (even after sharing an office for two year).
- My parents, for their love, support and encouraging me to continue.
- Japie and Schalk, my two favourite brothers in the world.
- Neels Fourie and Armscor for their financial support.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

CMS	Cepstral Mean Subtraction
DC	Diagonal covariance
EM	Expectation maximisation
FBS	Forward-Backward Search
FIT	Fast Incremental Training
GMM	Gaussian mixture model
HMM	Hidden Markov model
LC-HMM	Left-context hidden Markov model
LDA	Linear discriminant analysis
LVCSR	Large-vocabulary continuous speech recognition
MFCC	Mel-frequency cepstral coefficients
MLE	Maximum likelihood estimation
ORED	Order Reducing Algorithm
pdf	Probability Density Function
RC-HMM	Right-context hidden Markov model
VQ	Vector quantisation

Symbols

$\mathbf{X}_1^T$	Output observation sequence of length T.
$\mathbf{x}_t$	Output observation at time t.
N	The number of emitting states of a hidden Markov model Φ.
π_i	The initial state probability of state i.
$s_t = i$	Denotes the occurrence of state i at time t. Technically the state variable s_t takes on the value of i at time t.
s_t^*	The state at time t, that forms part of the optimal state sequence.
$a_{i_1 i_2 \cdots i_R}$	A R^{th}-order state transition probability.
$\overrightarrow{a}_{i_1 i_2 \cdots i_R}$	A left-context, R^{th}-order state transition probability conditioned

	on the preceding states.
$\overleftarrow{a}_{i_1 i_2 \cdots i_R}$	A right-context, R^{th}-order state transition probability conditioned on the following states.
$b_i(\mathbf{x}_t)$	The observation output probability density function associated with state i. Also denotes the likelihood of the state i generating the observation $\mathbf{x}_t$ at time t.
$\mathbf{S}_n^m$	The state sequence starting at time n and ending at time m.
$\mathbf{S}^*$	The optimal decoded state sequence of the model Φ that have generated the observations $\mathbf{X}_1^T$.
R	The Markov order of a HMM. Denotes how many states directly influence the state transition probabilities.
$P(\cdot)$	Probability
Φ	A hidden Markov model.
$\Phi_{\mathrm{R,lc}}$	An R^{th}-order, left-context hidden Markov model.
$\Phi_{\mathrm{R,rc}}$	An R^{th}-order, right-context hidden Markov model.
$\hat{\Phi}_{\mathrm{R-K,lc}}$	An $(R-K)$-order, left-context, pseudo HMM derived from the HMM $\Phi_{\mathrm{R,lc}}$.
$\hat{\Phi}_{\mathrm{R-K,rc}}$	An $(R-K)$-order, right-context, pseudo HMM derived from the HMM $\Phi_{\mathrm{R,lc}}$.
$\hat{\overrightarrow{a}}_{i_1 i_2 \cdots i_R}$	A left-context, R^{th}-order state transition pseudo probability conditioned on the preceding states.
$\hat{\overleftarrow{a}}_{i_1 i_2 \cdots i_R}$	A right-context, R^{th}-order state transition pseudo probability conditioned on the following states.
$\overrightarrow{\gamma}_t(i_1, \ldots, i_R)$	Left-context, state sequence probability observation, given the complete observation sequence $\mathbf{X}_1^T$.
$\hat{\overrightarrow{\gamma}}_t(i_1, \ldots, i_R)$	Approximate left-context, state sequence probability observation, given the complete observation sequence $\mathbf{X}_1^T$.
$\overrightarrow{\alpha}_t(i_1, \ldots, i_R)$	Left-context forward probability at time t.
$\overrightarrow{\delta}_t(i_1, \ldots, i_R)$	Left-context, best-path forward probability at time t.
$\overrightarrow{\delta}'_t(i_1, \ldots, i_R)$	Left-context, best-path forward probability at time t excluding the likelihood of the observation $\mathbf{x}_t$.
$\overrightarrow{\Psi}^{\delta}_t(i_1, \ldots, i_R)$	Left-context back pointer a time t.
$\overrightarrow{\beta}_t(i_1, \ldots, i_R)$	Left-context backward probability at time t.
$\overrightarrow{\beta}'_t(i_1, \ldots, i_R)$	Left-context backward probability at time t including the likelihood of the observation $\mathbf{x}_t$.
$\overrightarrow{\epsilon}_t(i_1, \ldots, i_R)$	Left-context, best-path backward probability at time t.
$\overrightarrow{\epsilon}'_t(i_1, \ldots, i_R)$	Left-context, best-path backward probability at time t including the likelihood of the observation $\mathbf{x}_t$.

$\overrightarrow{\Psi}^{\epsilon}_t(i_1, \ldots, i_R)$ Left-context "forward" pointer at time t.

$\hat{\overleftarrow{\alpha}}_t(i_1, \ldots, i_R)$ Right-context forward probability at time t.

$\hat{\overleftarrow{\delta}}_t(i_1, \ldots, i_R)$ Right-context, best-path forward probability at time t.

$\hat{\overleftarrow{\Psi}}^{\delta}_t(i_1, \ldots, i_R)$ Right-context back pointer a time t.

$\hat{\overleftarrow{\beta}}_t(i_1, \ldots, i_R)$ Right-context backward probability at time t.

$\hat{\overleftarrow{\epsilon}}_t(i_1, \ldots, i_R)$ Right-context, best-path backward probability at time t.

$\hat{\overleftarrow{\Psi}}^{\epsilon}_t(i_1, \ldots, i_R)$ Right-context "forward" pointer at time t.

$\hat{\overrightarrow{\alpha}}_t(i_1, \ldots, i_R)$ Left-context forward probability at time t, determined using the derived low-order HMM.

$\hat{\overrightarrow{\delta}}_t(i_1, \ldots, i_R)$ Left-context, best-path forward probability at time t, determined using the derived low-order HMM.

$\hat{\overrightarrow{\delta}}'_t(i_1, \ldots, i_R)$ Left-context, best-path forward probability at time t excluding the likelihood of the observation $\mathbf{x}_t$, determined using the derived, low-order HMM.

$\hat{\overrightarrow{\Psi}}^{\delta}_t(i_1, \ldots, i_R)$ Left-context back pointer a time t, determined using the derived, low-order HMM.

$\hat{\overrightarrow{\beta}}_t(i_1, \ldots, i_R)$ Left-context backward probability at time t, determined using the derived, low-order HMM.

$\hat{\overrightarrow{\beta}}'_t(i_1, \ldots, i_R)$ Left-context backward probability at time t including the likelihood of the observation $\mathbf{x}_t$, determined using the derived, low-order HMM.

$\hat{\overrightarrow{\epsilon}}_t(i_1, \ldots, i_R)$ Left-context, best-path backward probability at time t, determined using the derived, low-order HMM.

$\hat{\overrightarrow{\epsilon}}'_t(i_1, \ldots, i_R)$ Left-context, best-path backward probability at time t including the likelihood of the observation $\mathbf{x}_t$, determined using the derived, low-order HMM.

$\hat{\overrightarrow{\Psi}}^{\epsilon}_t(i_1, \ldots, i_R)$ Left-context back pointer a time t, determined using the derived, low-order HMM.

$\hat{\overleftarrow{\alpha}}_t(i_1, \ldots, i_R)$ Right-context forward probability at time t, determined using the derived, low-order HMM.

$\hat{\overleftarrow{\delta}}_t(i_1, \ldots, i_R)$ Right-context, best-path forward probability at time t, determined using the derived, low-order HMM.

$\hat{\overleftarrow{\Psi}}^{\delta}_t(i_1, \ldots, i_R)$ Right-context back pointer a time t, determined using the derived, low-order HMM.

$\hat{\overleftarrow{\beta}}_t(i_1, \ldots, i_R)$ Right-context backward probability at time t, determined using the derived, low-order HMM.

$\hat{\overleftarrow{\epsilon}}_t(i_1, \ldots, i_R)$ Right-context, best-path backward probability at time t, determined

	using the derived, low-order HMM.
$\hat{\overleftarrow{\Psi}}^{\epsilon}_t(i_1, \ldots, i_R)$	Right-context "forward" pointer at time t, determined using the derived, low-order HMM.
B	The pruning beam-width used during the search.
B_h	The pruning beam-width used in the heuristic search.
K	The number of order that is dropped when deriving the low-order HMM Φ_{R-K} from the R^{th}-order HMM Φ_R.
n	A node in the search graph G
$n_t(i_1, \ldots, i_R)$	A node in the search graph G which is uniquely specified by the time index t and the state sequence $(s_{t-R+1} = i_1, \ldots, s_t = i_R)$
n_s	The root node of the search graph G.
n_g	The goal node of the search graph G.
G	A search graph
$g(n)$	The cost function of the node n. The cost of the best path from the root node n_g to the node n.
$h(n)$	The heuristic function of the node n which is an estimate of the best path from the node n to the goal node n_g.
$h^*(n)$	The cost of the best-path from the node n to the goal node f_n.
	The evaluation function of the node n. An estimate of the cost of the best-path from the root node n_s to the goal node n_g which also goes through the node n.
$p(n)$	The parent node of the node n.
$M(n)$	The set of successors nodes to the node n.
v	A node in the search graph G.
v_{OPEN}	A node on the OPEN list.
v_{CLOSED}	A node on the CLOSED list.
v_M	The set of successors nodes to the node v.
$c\,[n_a, n_b]$	The cost of making a transition from node n_a to node n_b.
C_{tot}	The total decoding cost.
C_{s}	The search cost.
C_{h}	The heuristic cost.
C_{c}	The heuristic conversion cost.

Chapter 1

Introduction

1.1 Motivation

Most speech recognition and language identification engines are based on hidden Markov models (HMMs). HMMs concurrently model two stochastic processes, the underlying temporal structure and the locally stationary character of the process being modelled. Since efficient estimation and decoding algorithms exist for first-order HMMs, they are almost universally used in modern automatic speech recognisers. However, first-order HMMs have limitations which prevent them from properly modelling real-world stochastic processes [19]. The limitations arise from the first-order Markov assumption and the output-independence assumption. High-order HMMs are known to be more powerful, because of their better ability to model the temporal structure of the stochastic process by generalising the first-order Markov assumption [23, 24]. It has been shown that the use of high-order HMMs reduces the language identification error by a factor of three [11]. However, high-order HMMs have not been widely used because of their complexity and computational demands. In the past, HMMs have been decoded in a time-synchronous fashion using the Viterbi or pruned Viterbi-beam algorithms, which are breadth-first search algorithms. Breadth-first search algorithms are guaranteed to find the best solution, but might waste time by examining fruitless paths. In this dissertation we address the need for efficient algorithms for decoding high-order HMMs.

1.2 Research Objectives

The main objective of this dissertation is to develop a more time-efficient method of decoding high-order HMMs than the standard Viterbi decoding algorithm currently in use. We will specifically investigate using low-order HMMs to reduce the search space the decoder has to explore in order to find the optimal state sequence.

1.3 Prior work on decoding of HMMs

HMMs have been used in the field of continuous speech recognition since 1975 [5, 17]. Recently, HMMs have been used in a variety of fields including handwriting recognition [26, 27], pattern recognition in molecular biology [22, 13] and robotics [2]. Most of the research regarding decoding has been performed in the field of speech recognition. Since speech decoding can be viewed as the decoding of hierarchical, high-order HMMs[1], we will review some of the decoding strategies used for speech recognition as they might be applicable to the decoding of high-order HMMs.

1.3.1 Speech decoding strategies

According to Nguyen et al. [31], the most commonly used search algorithms are time-synchronous Viterbi-beam search and best-first stack search (a variant of A* search). However, the majority of decoding strategies are implemented using first-order HMMs. It is also important to realise that most speech decoding strategies do not use static HMM-based networks. The speech decoders implement the language models as N-grams and performs the decoding by dynamically creating (and destroying) the search graph. Although the N-gram is a special degenerate case of the HMM, we suspect that they are used because there exist efficient parameter estimation techniques for estimating N-gram probabilities from large text corpora. When performing large-vocabulary continuous speech recognition (LVCSR), the static HMM-based networks become too large for the available storage space (memory) [34] and thus various techniques have been developed for searching through a dynamically created search graph. Since the static and dynamic network are equivalent with respect to finding the optimal state sequence, we will only investigate the decoding of high-order HMMs using static networks.

The speech decoding search algorithms can be divided into the following categories [31]:

Fast Match

Fast match is a method for the rapid computation of a list of candidates that constrain successive search phases. Fast match is typically used in conjunction with a more accurate and computationally expensive search algorithm. The purpose of a fast match algorithm is to reduce the computational expense of performing the more complex search. In a sense, fast match can be regarded as an additional pruning threshold to meet. A fast match is admissible if the recognition errors that appear in a system using the fast match

[1]The language model represents the top-level, high-order HMM and the word or phone models represent lower-level, first-order HMMs.

followed by a detailed match are those that would appear if only the detailed match was performed [16].

Time-synchronous Viterbi search

The Viterbi search algorithm was first developed in 1967 [14, 45, 46]. Time-synchronous search algorithms explore all areas of the search space that occur at a specific time frame before moving onto the next time frame. All the states of the HMM are updated in lock-step frame-by-frame as the speech is processed. The computation required for this method is proportional to the number of states in the model and the number of frames in the input. The Viterbi search is admissible and the Viterbi-beam search is inadmissible, although it has been found that for suitably wide beams, the Viterbi-beam algorithm rarely does not find the optimal state sequence [28]. Little benefit is gained from using a fast match algorithm as the search considers starting all possible words at each time frame. Thus, it would be necessary to run the fast match algorithm at each time frame, which would be too expensive.

Best-first stack search

The stack decoder was first developed by IBM [4] and has been successfully used in LVCSR systems [3, 18, 36, 37]. The true best-first search algorithm keeps a sorted stack of the highest scoring hypotheses (or partial sequence through the HMM). At each iteration, the hypothesis with the highest score is advanced by all possible next words, which results in more hypotheses on the stack. The best-first search has the advantage that it can theoretically minimise the number of hypotheses considered if there is a good (heuristic) function to predict which theory to follow next. The heuristic function determines whether the best-first stack search is admissible. The search can also take very good advantage of a fast match algorithm at the point where it advances the best hypothesis. The main disadvantage is that there is no guarantee as to when the algorithm will finish. In addition, it is very hard to compare theories of different length.

Pseudo time-synchronous stack search

This search is a compromise between time-synchronous search and best-first search. In this search, the shortest hypothesis (the one that ends earliest in the signal) is updated first. All active hypotheses are within a short time delay of the end of the speech signal. To keep the algorithm from requiring exponential time, a beam-type pruning is applied to all hypotheses that end at the same time. Since the method advances one hypothesis at a time, it can also take advantage of a fast match algorithm.

N-Best paradigm

This paradigm was developed in 1989 as a way to integrate speech recognition with natural language processing [41, 42]. It is a type of fast match at the sentence level, which reduces the search space to a short list of likely whole-sentence hypotheses. The idea is to use simple and fast knowledge sources to quickly determine a short list of likely sentences. These likely sentences are then re-scored using more complex and detailed knowledge sources. As with fast match algorithms, there is the possibility of pruning away the correct sentence during the N-Best list generation, which causes the search to be inadmissible.

Forward-Backward Search Paradigm

The algorithm is a general paradigm developed in 1986 [1] in which inexpensive approximate time-synchronous search in the forward direction is used to speed up a more complex search in the backwards direction. A disadvantage of most of the other decoding strategies is that the pruning is only based on the partial observation sequence seen thus far. It can happen that a state, occurring at an early time frame, might seem promising based on the observations seen thus far. The state might be part of a path that is not very promising if the rest of the observations are included. The forward-backward search incorporates the complete observation at each time frame when the search space is pruned. The true power of the algorithm is revealed when different models are used in the forward and backward directions. In the forward direction approximate acoustic models can be used while in the backward direction more detailed HMMs with more complex language models are used.

1.3.2 HMM types

Over the years a number of different types of HMMs have been developed. Bengio [7] provides an excellent review of HMMs, the different types of HMMs and extensions to HMMs and related models. The different types of HMMs usually only differ with respect to the definition of the state output observations probability density functions (pdfs), the definition of the state transition probabilities and the topology. The majority of speech decoders use discrete-valued hidden states, and continuous or discrete output observation pdfs conditioned on a single state.

Since both the high-order and low-order HMM share the same set of pdfs, we are not overly concerned with HMM variants based on different pdfs. In this research we limit our investigation to HMMs that use mixtures of diagonal-covariance Gaussian densities as state output pdfs. Furthermore, we have only investigated topologies that initially start as fully-connected in the first-order. By allowing state transition probabilities to be dependent on previous states, and not only the current state, we obtain high-order HMMs, which we are primarily concerned with in this research.

1.3.3 High-order HMM algorithms

There are two approaches to using high-order HMMs in the literature. The first approach is to extend the existing first-order algorithms to customised algorithms which is applicable to specific orders and types of HMMs. The disadvantage of this custom approach is that new algorithms need to be created for each different order of HMM. The second approach is to reduce the high-order HMMs to first-order equivalent HMMs and then to use the first-order algorithms. The advantage is that we only need to use the existing (and well-understood) algorithms which are applicable to first-order HMMs (such as Viterbi, Baum-Welch, Forward-Backward, etc.). In this research we favour the second approach, since we have found that using first-order equivalent HMMs results in a deeper insight into the behaviour of the high-order HMMs. However, both approaches are equally valid and result in equivalent high-order behaviour.

It is surprising that the literature contains only a few high-order HMM algorithms. The algorithms can be summarised as follows:

- ORder REDucing (ORED) algorithm [11]: This algorithm reduces arbitrary order HMMs to their first-order equivalent HMMs, thereby enabling the use of all the efficient algorithms that has been developed for first-order HMMs.

- Fast Incremental Training (FIT) of high-order HMMs [12]: This algorithm is used to efficiently estimate the parameters of high-order HMMs.

- Time-synchronous Viterbi-beam: This is the standard first-order decoding algorithm which is used to find the optimal state sequence which has generated a given sequence of observations. He [15] was the first to extend the first-order Viterbi algorithm to the second order.

- Time-synchronous Baum-Welch re-estimation: Krioule, Mari and Haton then extended the Baum-Welch re-estimation algorithm by deriving an algorithm specific to second-order discrete HMMs [21].

We find it interesting that the only high-order decoding algorithm that seems to be used is the time-synchronous Viterbi-beam algorithm. As previously mentioned, the speech decoding problem can be viewed in terms of hierarchical, high-order HMMs. Therefore, it follows that the algorithms that have been developed for speech recognition might be applicable to the problem of decoding high-order HMMS. We will now consider the applicability of the decoding strategies discussed in Section 1.3.1 to the task of decoding high-order HMMs.

Fast-match

Fast match is used to rapidly compile a short list to constrain successive search phases. When fast match is used in speech decoding, the search space is already divided into higher-level categories such as phones and words. The problem with decoding high-order HMMs is that the decoder must find the optimal *state* sequence. In order to apply fast-match to high-order HMMs, it would be necessary to divide the states of the high-order HMMs into some form of sub-categories. Another possibility would be to use less complex pdfs. However, applying fast match to the states of the high-order HMM will lead to the same problems as are found when using fast match with time-synchronous Viterbi search. Since the fast-match would need to be calculated at every time frame, we do not believe fast match can be easily adapted to the decoding of general high-order HMMs.

Time-synchronous search

The time-synchronous Viterbi and pruned Viterbi-beam search are already used for decoding high-order HMMs. Since the number of possible transitions of an N-emitting state, R^{th}-order HMM increases exponentially with the order of the HMM as $O(N^{R+1})$, this causes the computational expense of the Viterbi algorithm to be $O(TN^R)$, for a T-length observation sequence. The Viterbi-beam algorithm improves the computational efficiency of finding the optimal state sequence, but it is difficult to predict the savings in expense. The disadvantage of using the Viterbi search is the exponential increase in the expense of decoding high-order HMMs.

Best-first stack search

The best-first stack search is essentially the A* search algorithm with the heuristic function set to zero. The main disadvantage is that there is no guarantee as to when the algorithm will finish. It should be possible to develop a heuristic function based on low-order HMMs. The challenge is to develop a heuristic function that is admissible for high-order HMMs. The heuristic function used for speech recognition is word-dependent and what is required is a state-dependent heuristic function.

Pseudo time-synchronous stack search

This search is closely related to the best-first stack search, except that beam-pruning is applied to states at the same time-frame. It would also be possible to incorporate low-order HMMs into this search, but the same challenge remains of developing an admissible heuristic function.

N-Best paradigm

It should be possible to use a low-order HMM to generate an N-best list of state sequences. The N-best list of state sequences could then be re-scored using the more complex high-order HMM. The biggest problem with this approach is that N needs to be fairly large to obtain state sequences which truly differ in the state identities. Typically the top N entries in the N-best list is the same series of states, the entries simply differ in the exact time frames that each state occupies. Informal experiments have shown that at the point where N becomes large enough for truly different state series, the N-best list computation becomes larger than the expense of using the Viterbi-beam search.

Forward-Backward Search Paradigm

The Forward-Backward search seems to be the most promising method of using low-order HMMs to guide the search of high-order HMMs. A backward search could be performed on the low-order HMMs to compute the probability of the partial path from a specific state to the final state. When the forward search is performed using the more complex high-order HMMs, the low-order backward probabilities can then be combined with the high-order forward probabilities so that the complete observation sequence can be used for pruning at each time frame. The backward search can be performed using the time-synchronous Viterbi-beam algorithm, while the more detailed forward search could be performed using either the Viterbi-beam search algorithm or the A*-based search algorithms.

1.4 Research Overview

This section provides a high-level overview of the work done in this research. Two decoding approaches based on the adaption of the Forward-Backward search paradigm to the decoding of high-order HMMs are investigated. The first approach is based on the time-synchronous, breadth-first, forward-backward search algorithm, where we wish to base our state pruning on the complete observation sequence, instead of only basing the state pruning on the partial observation sequence, as the Viterbi-beam algorithm does.

The second approach is based on the time-asynchronous, best-first, A* search algorithm. In this approach we will still use state pruning based on complete observations, but the order in which partial paths are examined differs from the previous approach as a heuristic function is used to guide the search so that only the most likely paths are considered. The choice of heuristic is critical to the A* search algorithms and a novel, task-independent heuristic function will be presented.

The following subsections outline the research presented in this dissertation:

1.4.1 High-order HMMs

In chapter 2 we give an overview of hidden Markov model theory by defining the general left-context HMMs and the notation used to manipulate HMMs. This is followed by a discussion on the different types of search algorithms and specifically the standard Viterbi-beam decoder. Since the decoding behaviour of high-order HMMs are not well-known, we end the chapter by performing an experiment to measure the computational expense of the Viterbi-beam search algorithm, by decoding fully-connected, high-order HMMs. These results will be used as the base-line to which the proposed decoders will be compared.

1.4.2 Forward-Backward search of high-order HMMs

In chapters 3 and 4 we discuss how the Forward-Backward search algorithm can be extended or adapted to the task of decoding high-order HMMs. Austin et al. [1, 41] used a simplified algorithm in their forward search and a complicated algorithm in their backward search. We suspect that their heuristic was calculated during the forward search as they were interested in developing a real-time speech decoder. In their later work [31, 32, 29] they used simplified *models* in the forward search and more complex models in the backward search. In this research the heuristic function is first determined during a backward search, using less complex, low-order HMMS. This is then combined with the more complex high-order HMMs during a forward search.

The heuristic function is obtained by backward decoding derived, low-order HMMs. We present two types of decoders, the first uses a time-synchronous Viterbi-beam decoder during the more complex forward search and the second decoder uses the time-asynchronous A* decoder during the forward search. Since the heuristic function for the A* decoder needs to be an accurate prediction of the actual scores that will result during the forward pass, we present a novel, task-independent heuristic for the A* decoder. The admissibility of our heuristic is proven in Appendix B.

Since the information used to guide the decoders (the heuristic function) is obtained by backward decoding low-order HMMs, we continue the chapter by discussing the backward Viterbi-beam decoding algorithm. There is an implicit assumption in the Forward-Backward search paradigm that forward and backward search are computationally equivalent. We test this assumption be measuring the computational expense of backward decoding high-order, left-context HMMs. We are surprised to discover that pruned backward decoding is significantly more expensive than pruned forward decoding, when the same HMM and observations are used. This can cause serious problems for the Forward-Backward search, since the search algorithm depends on the simplified backward search being computationally less expensive than the forward search. We believe that backward decoding is not fundamentally more expensive than forward decoding, but that this dis-

crepancy is caused by the time-asynchronicity of observations and states processed during backward Viterbi-beam decoding. This time-asynchronicity is a result of the definition of left-context transition probabilities and is therefore fundamental to high-order, left-context HMMs. The solution to this problem is the development of the right-context HMM.

In chapter 4 we define a new type of HMM, of which the observations and states will be synchronised in the backward direction. The difference between left-context and right-context HMMs are that the transition probabilities are conditioned on the subsequent states, and not the preceding states. We measure the computational expense of backward decoding the right-context HMM and show that it is computationally equivalent to the forward decoding of left-context HMMs, which allows the Forward-Backward search to be applied to decoding high-order HMMs. In the rest of the chapter we continue to show how the heuristic function, for left-context HMMs, can be determined using equivalent right-context HMMs.

1.4.3 Implementation and Evaluation of decoders

In chapters 5 and 6 we discuss the practical implementation and evaluation of our proposed decoders. Chapter 5 discuss some of the practical issues that needs to be addressed when implementing the new decoders. The issues include efficient memory management as well the choice of efficient data structures used during the decoding.

In Chapter 6 we present the experiments used to evaluate our proposed decoders. We use the CallFriend speech corpus to investigate the influence of different types of pdfs as well as the size of the high-order HMMs, on the computational expense of the decoders. We show that both the proposed decoders are computationally less expensive than the standard Viterbi-beam decoder, with the Forward-Backward search based Viterbi-beam decoder (FBS-Viterbi-beam decoder) being the least expensive. Lastly, we analyse both decoders in order to determine the ratio of the decoding time being spent on computing the heuristic and on performing the search. This analysis also shows that the new decoders are more consistent than the standard Viterbi-beam decoder.

Having shown that when the Forward-Backward search paradigm is adapted to the task of decoding high-order HMMs, it results in search algorithms that are more computationally efficient than time-synchronous Viterbi-beam search, we conclude in Chapter 7 by mentioning some of the outstanding issues and discussing further topics of research.

1.5 Contributions

The contributions of this dissertation can be summarised as follows:

- We show that the forward and backward Viterbi-beam decoding of high-order, left-context HMMs are not computationally equivalent.
- We propose a new definition for the state transition probabilities of an HMM. This leads to a new type of HMM we have termed the right-context HMM.
- We prove that the right-context HMM is mathematically equivalent to the conventionally defined, left-context HMM.
- We show that performing backward Viterbi-beam decoding on the right-context HMM is as efficient as performing forward Viterbi-beam decoding on the left-context HMM.
- We propose two decoders based on the Forward-Backward search paradigm. These decoders incorporate information obtained from decoding low-order derived HMMs. The first decoder is a time-synchronous decoder based on the Viterbi-beam algorithm while the second decoder is a time-asynchronous decoder based on the A* search decoder.
- We propose a novel, task-independent heuristic function for the A* decoder and have proven that the heuristic is admissible. When A* decoders are used in conjunction with HMMs it is usually to perform the task of continuous speech recognition. Heuristic functions have been defined which are specific to the task of continuous speech recognition. Our proposed heuristic function is not specific to the task to which the HMMs are applied and is therefore task-independent.
- We show that both decoders based on the Forward-Backward search paradigm are computationally more efficient in finding the optimal state sequence than the standard time-synchronous Viterbi-beam decoder. When the two new decoders are compared, we find that the time-synchronous, FBS-Viterbi-beam decoder is computationally more efficient than the time-asynchronous A* decoder.
- By analysing the behaviour of the decoders we also show that the new decoders, specifically the FBS-Viterbi-beam decoder, are computationally more consistent than the Viterbi-beam decoder. This also shows that it is better to prune the search space based on complete observations, rather than using partial observations.

The last two contributions show that we have met our stated research objective of developing a more time-efficient search algorithm than the currently used Viterbi-beam algorithm.

Chapter 2

Hidden Markov Models

Hidden Markov models are mathematical models which are used to model stochastic processes. The purpose of this chapter is to define the mathematical notation used to describe hidden Markov models (HMMs). We will shortly discuss the commonly used first-order HMM, but the focus will be on high-order HMMs as well as their first-order equivalent HMMs.

2.1 Conventional HMMs

A conventional HMM consists of a finite set of states that is traversed according to a set of transition probabilities. The transition probabilities conventionally describe the conditional probability of the HMM occupying a specific state, given a *history* of the states that were *previously* occupied. The transition probabilities are usually assumed to be homogeneous, i.e. the same for all time frames. Each state has an associated output probability distribution, which defines the conditional probability that the HMM emits an observation (or feature vector), given that the model is occupying a specific state. An HMM concurrently models two stochastic processes: the temporal structure and the locally stationary character of the system being modelled. The temporal structure is modelled by the transition probabilities and the locally stationary character is modelled by the output conditional pdf. Since only the sequence of output observations is known, the state sequence is said to be hidden (hence the name *hidden* Markov model). The HMM can be viewed as a doubly-embedded stochastic process with the underlying stochastic process (the state sequence) not directly observable.

The two main components of an HMM is the set of probability distribution functions (pdfs) with its corresponding state transition probabilities and the topology (the structure dictating which states are coupled). HMMs can also be viewed as describing probable trajectories in the observation space. The observation space is described by the pdfs associated with the HMM. The trajectories are described by the topology of the HMM

and the probabilities of the trajectories are influenced by the transition probabilities.

We now introduce the notation and conventions required when presenting the algorithms used with HMMs.

2.1.1 Definition and Notation

$\mathbf{X}_1^T = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ denotes the output observation sequence of length T that we want to match to the HMM Φ. The HMM consists of N emitting states, each with an associated conditional pdf. We add additional initial and terminating non-emitting (or null) states so that the HMM consists of a total $N+2$ states. The initial state will always be indexed as state 0 and the terminating state will always be indexed as state $N+1$. The use of extra initial state probabilities (π_i) is commonly seen in the literature, but the additional initial and terminating states make the use of these extra variables unnecessary as the parameters are included in the state transition probabilities ($\pi_i = a_{0i}$). $s_t = i$ denotes the occurrence of state i at time t. The output pdf for state i is denoted by $b_i(\mathbf{x}_t) = f(\mathbf{x}_t|s_t = i)$. The sequence $\mathbf{S}_n^m = \{s_n, s_{n+1}, \ldots, s_m\}$ denotes the occurrence of a sequence of states from time n to time m.

The states are coupled with state transition probabilities indicated by the symbol a with subscripts to index the states involved. In a conventional HMM the state transition probabilities for a R^{th}-order HMM is defined as the conditional probability of making a transition at time t to state i given the sequence of the R preceding states. Thus the conditional probability of making a transition is dependent on the identity of the *preceding* states. As the state transition is only influenced by the preceding states, we will refer to conventionally defined state transition probabilities as *left-context* state transition probabilities. Conventionally defined HMMs, which utilises left-context transition probabilities, will also be referred to as left-context[1] HMMs.

When processing HMMs in pattern recognition applications, there are three principle issues that need that need to be addressed. Firstly, we need to be able to estimate new model parameters from training observations. This issue is called the Learning problem. Secondly, we need to compute the probability of the model given a set of observations. This issue is commonly referred to as the Evaluation problem. Lastly, we need to be able to determine the hidden state sequence that most probably produced a set of observations. This issue is called the Decoding problem. These three principle issues can be formally stated as:

1. **The evaluation problem**: given a model Φ and a sequence of observations $\mathbf{X}_1^T$,

[1]This is not to be confused with left-context biphone models, which are context-dependent phonemes modelled with HMMs.

what is the probability that the model generates the observations i.e. $P(\mathbf{X}_1^T|\Phi)$?

2. **The learning problem**: given a model Φ and a set of observations, how should the model parameter $\tilde{\Phi}$ be adjusted to maximise the joint likelihood $\prod_{\mathbf{X}} P(\mathbf{X}|\Phi)$?

3. **The decoding problem**: given a model Φ and a sequence of observations $\mathbf{X}_1^T$, what state sequence $\mathbf{S}_1^T$ has the highest likelihood of producing the observations?

This dissertation is primarily concerned with solutions to the decoding problem. We are specifically interested in solutions that are more time-efficient than the currently available, standard solutions. The newly developed solutions to the decoding problem form the core of this dissertation and will be discussed in detail in Chapter 3. The standard solution to the decoding problem, namely the Viterbi algorithm, will be discussed later in this chapter. Before we continue our discussion of HMMs, it is necessary to state the assumptions that are required in order to make HMM computations tractable.

2.1.2 HMM Assumptions

Two simplifying assumptions regarding HMMs are made in order to make calculations regarding the three principle issues tractable. The two assumptions are called the *Observation Independence* assumption and the *Markov Order* assumption.

2.1.2.1 Observation Independence Assumption

The first assumption is mathematically expressed as:

$$f\left(\mathbf{x}_t|\mathbf{X}_1^{t-1}, \mathbf{S}_1^t, \Phi\right) = f\left(\mathbf{x}_t|s_t, \Phi\right) \tag{2.1}$$

This means that the likelihood of the t^{th} observation is only dependent on the current state s_t and is not affected by other states or observations. This assumption is not affected by the order R of the HMM.

2.1.2.2 Markov Order Assumption

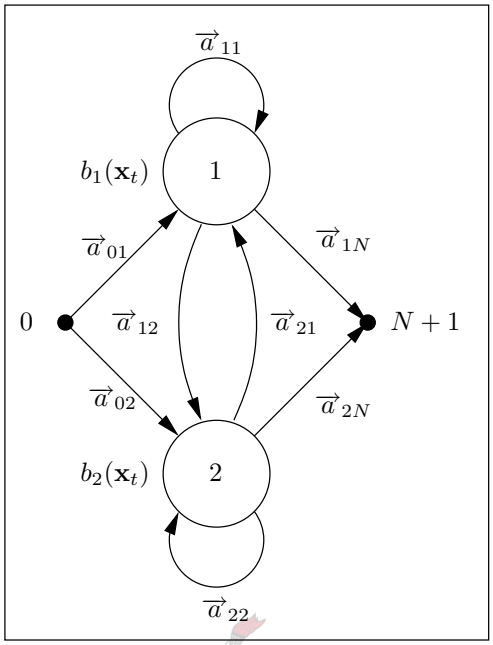
The Markov order assumption is mathematically expressed as:

$$P(s_t|\mathbf{S}_1^{t-1}, \mathbf{X}_1^{T-1}, \Phi) = P(s_t|\mathbf{S}_{t-R}^{t-1}, \Phi) \tag{2.2}$$

This means that the probability of occurrence of the next state is only affected by the identity of the immediately preceding R states. Other states or observations do not affect this probability of occurrence. This assumption is influenced by the order of the HMM.

Having presented the notations necessary to describe and manipulate HMMs we are ready to discuss the most commonly used HMM: the first-order HMM.

Figure 2.1: *A two-emitting state, fully connected, first-order HMM.*



2.1.3 First-order HMMs

By fixing the order of the HMM to $R = 1$ we obtain the first-order HMM. The implication of this for the Markov order assumption, is that the conditional probability of making a transition is only dependent on the identity of the *preceding* state. The state transition probabilities for a first-order HMM are defined as the conditional probability (denoted by $\overrightarrow{a}_{ij}$) of making a transition at time t to state j given the preceding state $s_{t-1} = i$:

$$\begin{aligned} \overrightarrow{a}_{ij} &\triangleq P(s_t = j | s_{t-1} = i) \text{ with} \\ \sum_{j=0}^{N+1} \overrightarrow{a}_{ij} &= 1,\ i \in \{0 \cdots, N+1\} \end{aligned} \tag{2.3}$$

Figure 2.1 illustrates a typical two-emitting state, first-order HMM. We note in the figure that any two states of a first-order HMM is only coupled by a single transition probability, as the transition probability is only dependent on a single preceding state. Note that a link from state i to state j is distinct from the link from state j to state i.

A conventional, N emitting-state, first-order HMM Φ_1 is defined by the parameter set

$$\Phi_{1,\text{lc}} = \{\overrightarrow{a}_{ij}, b_i(\mathbf{x}), i, j \in \{0, \cdots, N+1\}\} \tag{2.4}$$

where the subscript $_{\text{lc}}$ denotes that it is a left-context HMM.

For the processing of first-order HMMs standard algorithms have been developed that efficiently solve the three principle issues. The *Baum-Welch* algorithm is the solution for the Learning problem and the *Forward* algorithm is the solution for the Evaluation problem. The Viterbi algorithm is the solution to the Decoding problem. All three these

algorithms are based on dynamic programming techniques [6], while the Baum-Welch algorithm is an application of the Expectation Maximisation algorithm to HMMs [25].

2.1.3.1 Limitations of first-order HMMs

Currently, first-order HMMs have two main limitations. The first limitation is a result of the first-order Markov assumption which states that being in a state only depends on the identity of the previous state.

The second limitation is that HMMs are well defined only for processes that are a function of a single independent variable, such as time or one-dimensional position.

The first limitation is not a fundamental one by any means. In principle, it is possible to define high-order HMMs in which the dependence is extended to previous states (and outputs). It is the general consensus that such high-order extensions complicate HMMs and quickly results in intractable computation as the number of transitions can grow exponentially with the order of the HMM.

2.1.4 High-order HMMs

By allowing the order of the HMM to be $R \geq 2$ we obtain high-order HMMs. The implication of this for the Markov order assumption is that the conditional probability of making a transition is only dependent on the identity of the R *preceding* states. This is formally stated as follows: The state transition probabilities for a R^{th}-order HMM is defined as the conditional probability (denoted by $\overrightarrow{a}_{i_1 i_2 \ldots i_{R+1}}$) of making a transition at time t to state i_{R+1} given that the identity of the preceding states are $\{s_{t-1} = i_R, \ldots, s_{t-R+1} = i_2, s_{t-R} = i_1\}$:

$$\begin{aligned} \overrightarrow{a}_{i_1 i_2 \ldots i_{R+1}} &\triangleq P(s_t = i_{R+1} | s_{t-1} = i_R, \ldots, s_{t-R+1} = i_2, s_{t-R} = i_1) \text{ with} \\ \sum_{i_{R+1}=0}^{N+1} \overrightarrow{a}_{i_1 i_2 \ldots i_{R+1}} &= 1, \; i_1, i_2, \ldots, i_R \in \{0, \cdots, N+1\} \end{aligned} \tag{2.5}$$

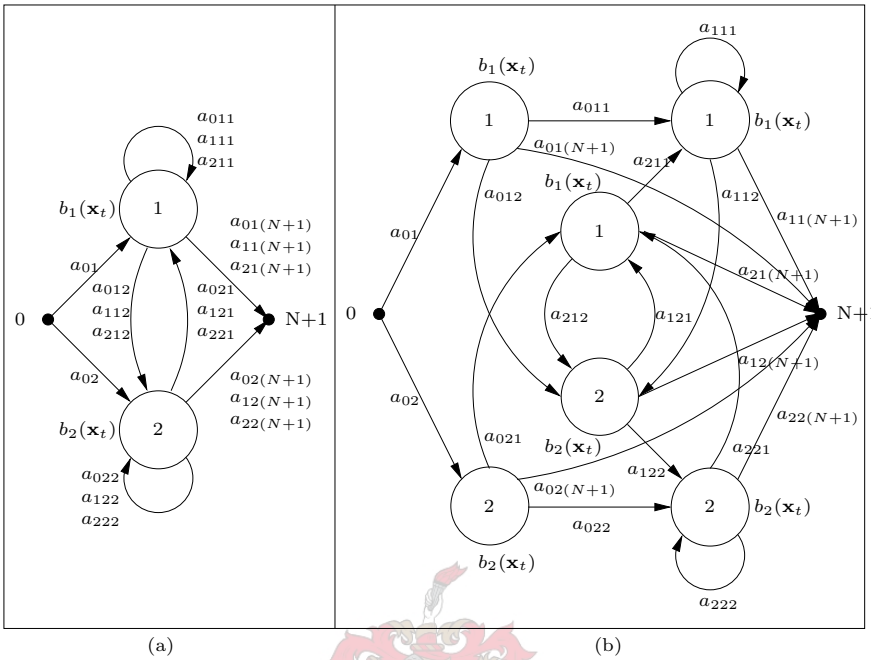
Fig. 2.2(a) illustrates a two emitting-state, second-order HMM with its initial and terminating null states. Note that linked states are coupled with multiple transition probabilities because the transition probabilities are dependent on the identity of two preceding states.

A conventional, N emitting-state, R^{th}-order HMM Φ_{R} is defined by the parameter set

$$\Phi_{\mathrm{R,lc}} = \left\{ \overrightarrow{a}_{i_1 i_2 \ldots i_{R+1}}, b_{i_1}(\mathbf{x}), i_1, i_2, \ldots, i_{R+1} \in \{0, \cdots, N+1\} \right\} \tag{2.6}$$

For the processing of high-order HMMs it is possible to expand the standard first-order algorithms to the higher-orders as has been done by Mari et al. in [24].

Figure 2.2: *(a) A second-order, two-emitting state, left-context HMM. (b) First-order equivalent of (a).*



2.1.4.1 Limitations of high-order HMMs

Three limitations exist to the use of high-order HMMs in practical pattern recognition systems. The first limitation is the computational expense of using high-order HMMs. This computational expense is as a result of the exponential increase in the number of transition probabilities, as the order of the HMM increases. This also requires an increase in data to properly estimate the increased number of parameters.

The second limitation is the computational concerns when training high-order HMMs. Unfortunately, due to the large number of parameters involved in high-order HMMs, directly training such HMMs (or their first-order equivalents) can be a very computationally expensive task. Du Preez [12] also showed that directly training the high-order HMMs results in poorly estimated transition probabilities. He went on to show that it is more efficient to train high-order HMMs by starting at the first order and incrementally increasing the order of the HMM. He called this method Fast Incremental Training (FIT) and it avoids training redundant high-order probabilities by noting which lower-order transition probabilities are zero. This considerably reduces the memory and processor requirements during training. In addition, the resultant models have far fewer parameters and generalise better on previously unseen data. The high-order HMMs used in this research will therefore be trained using the FIT method of training.

The third limitation is the need to expand the standard first-order algorithms to each order. If therefore becomes necessary to build a separate version of each algorithm for

each order of HMM. This limitation can be negated by the use of first-order equivalent HMMs.

2.1.4.2 First-order equivalent HMMs

In this research we use mathematically equivalent, first-order representations of high-order HMMs. This allows us to use the standard, first-order algorithms on any order HMM, instead of having to implement the decoding algorithms for each HMM order. For example, a second-order HMM can be reduced to its first-order equivalent HMM by mapping the states of the second-order HMM, to a first-order HMM, using a twofold product of the original state space [15]. Du Preez [11] presented an algorithm to perform this mapping and called it the Order REDucing (ORED) algorithm.

In Fig. 2.2(b) we show the first-order equivalent of the HMM shown in Fig. 2.2(a). Note that any two states are coupled only with a single transition probability and that multiple states share the same conditional output pdf.

We share Du Preez's viewpoint of high-order HMMs in that

> *High-order transition probabilities are simply an elegant mathematical way of specifying the topology of the model.*

Thus, when we present the theory of high-order HMMs, we use the mathematically elegant high-order transition probabilities. However, when using high-order HMMs, we reduce the high-order HMMs to their first-order equivalents and use the standard first-order algorithms for processing. We thereby avoid the laborious task of implementing the standard (and newly proposed) decoding algorithms for each new order of HMM that we wish to process. We have now established the notation required to manipulate high-order HMMs.

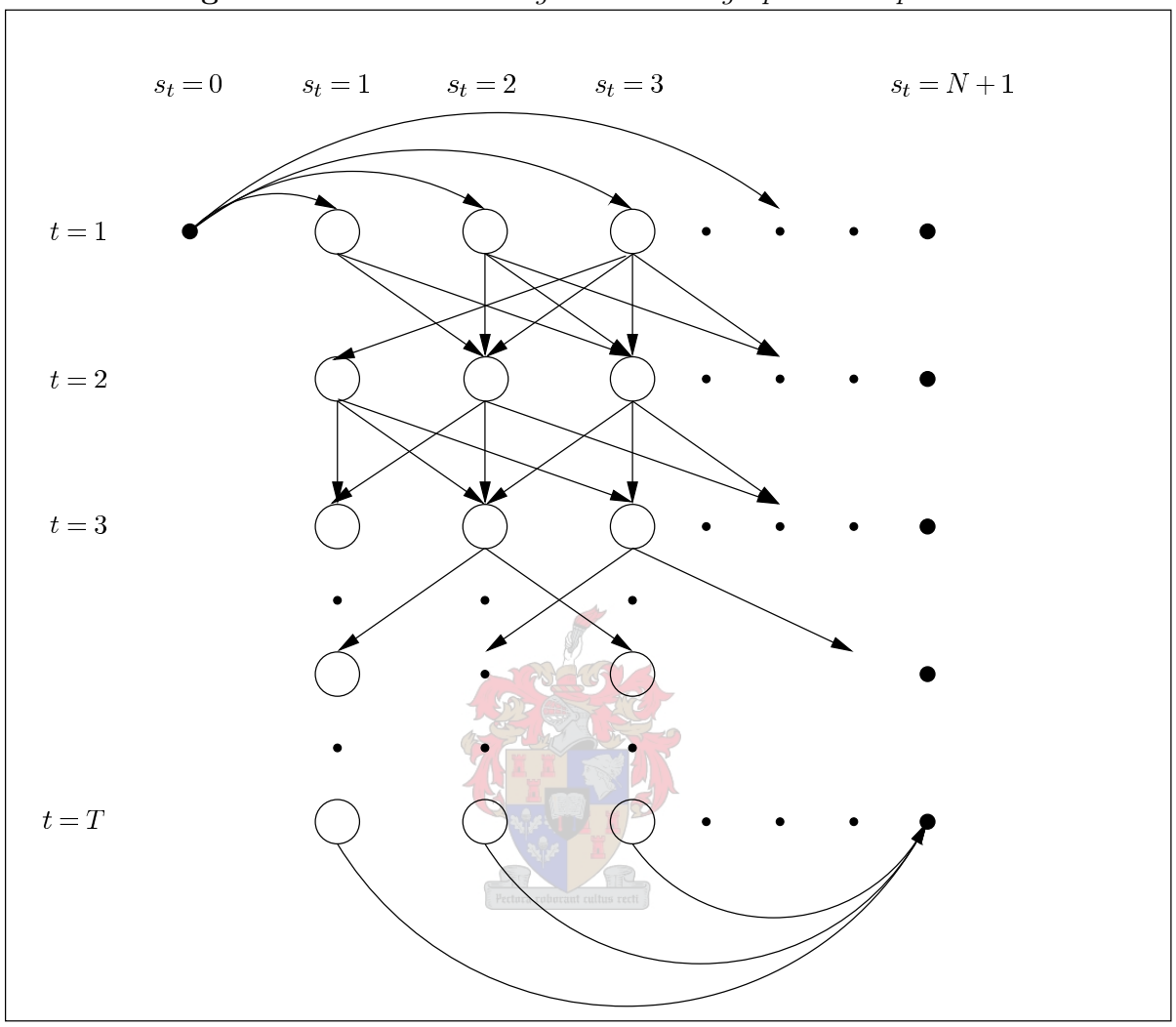
2.2 Decoding of left-context HMMs

In this section we will discuss the standard algorithms used to decode conventional, left-context HMMs.

2.2.1 Types of decoding

We view the decoding of HMMs as a graph search problem as illustrated in Fig. 2.3. The search graph has a root node at time $t = 1$ with state index 0. The goal node occurs at time $t = T$ with state index $N + 1$. Emitting states transition from a node at time t to a node at time $t + 1$ while non-emitting states transition to other states without causing a change in the time index. The purpose of the decoding algorithm is to find the path

Figure 2.3: *HMM decoding viewed as a graph search problem.*



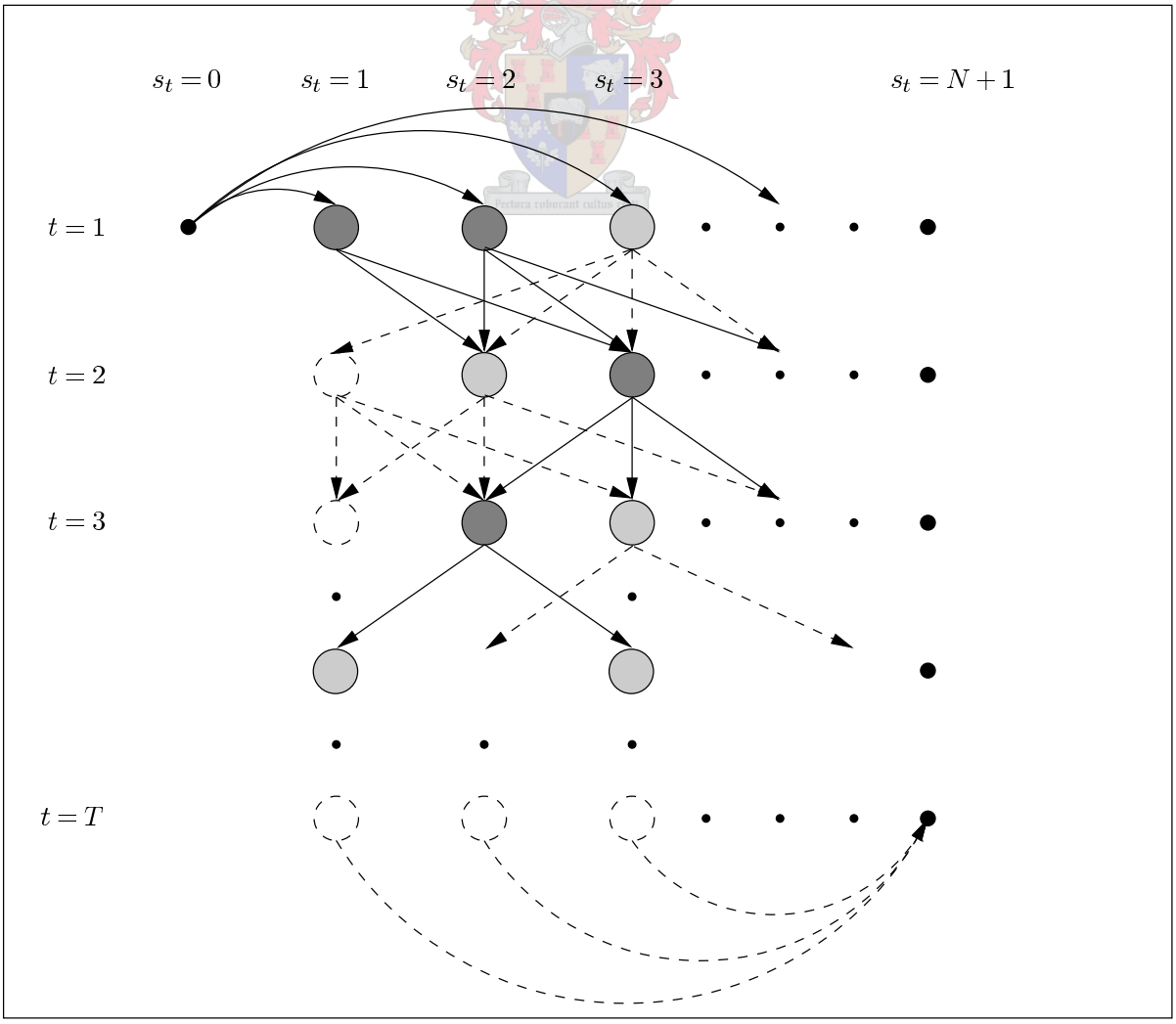
through the graph from the initial node to the goal node, with the maximum likelihood of generating the observation sequence $\mathbf{X}_1^T$. Thus decoding can be seen as the process of starting at the root node and "growing" the search graph until the search algorithm terminates, preferably with the optimal path from the root to the goal node.

When conducting a decoding search through the graph, the search algorithms can be categorised according to the strategy used to traverse (or grow) the search graph. The nodes (the circles) represent HMM states occurring at specific time indexes. The arrows between nodes represent the "cost" of transitioning from one node to another. This "cost" includes both the observation-independent state transition probability, $\hat{a}_{i_1 i_2 \cdots i_{R+1}}$ and the observation-dependent output observation likelihood $b_{i_{R+1}}(\mathbf{x}_t)$. The two search categories of primary interest in this dissertation are time-synchronous search and time-asynchronous search.

2.2.1.1 Time-asynchronous decoding

Time-asynchronous decoding grows the search graph by expanding a node occurring at any time-index. Search algorithms that traverse the search graph in time-asynchronous manner can grow the search graph in such a manner that not all states at a certain time index exist in the graph. There are a number of time-asynchronous search algorithms, for example depth-first search. We are interested in search algorithms that can be guided with information obtained from low-order HMMs, therefore we are interested in the class of *informed* searches. Informed searches traverse the search graph illustrated in Fig. 2.3 by first expanding the initial root node $s_1 = 0$. The algorithms subsequently use some form of heuristic, specific to the problem, to determine which node, at any time-index, is the "best" node to expand next. The purpose is to expend the least amount of effort in finding a path from the root node to the goal node, by expanding the "best" nodes first. Fig. 2.4 illustrates how informed searches traverse the search graph. The dark gray

Figure 2.4: *Time-asynchronous decoding of an HMM, when the decoding is viewed as a graph search problem.*



circles represent nodes that have already been expanded. The light gray circles represent leaf nodes of the search graph and have therefore not been expanded yet. The dashed circles represent nodes that the search algorithm have not yet reached and the dashed lines between circles represent parts of the search graph that the search algorithm has not yet reached. When the search graph is grown, the "best" node to expand is always selected from the set of leaf nodes.

The A* search algorithm is well suited for incorporating low-order HMM information. We will discuss the A* search algorithm in more detail in Chapter 3.

2.2.1.2 Time-synchronous decoding

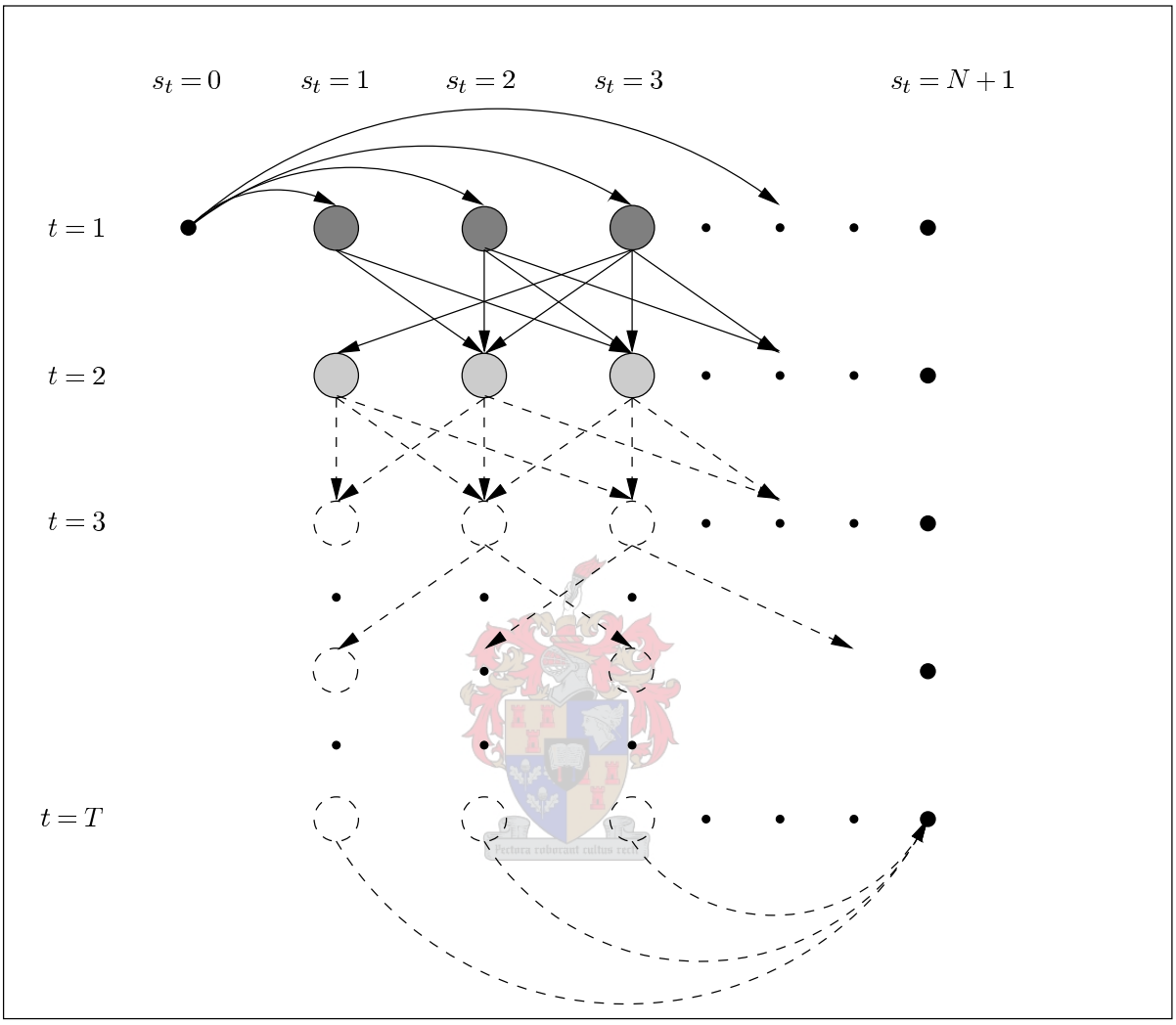
Time-synchronous decoding grows the graph by first expanding all nodes occurring at a specific time-index. Thus, search algorithms that grow the search graph in a time-synchronous manner consider all nodes at a specific time index before considering nodes that occur at the next time index. Time-synchronous searches traverse the search graph illustrated in Fig. 2.3 by first expanding the initial root node $s_1 = 0$. The algorithms next expand all nodes occurring at time $t = 1$ before increasing the time index t by one. At any time in the search the set of leaf-nodes from which the search selects nodes to expand, is solely determined by the time index under consideration. Time-synchronous search algorithms are guaranteed to find an optimal path from the root node to the goal node. A typical time-synchronous search is illustrated in Fig. 2.5. The dark gray circles represent nodes that have already been expanded. The light gray circles represent leaf nodes of the search graph and therefore have not yet been expanded. The dashed circles represent nodes that the search algorithm have not yet reached and the dashed lines between circles represent parts of the search graph that the search algorithm has not yet reached.

Although the Viterbi decoder is a special case of dynamic programming, it can also be viewed as a time-synchronous graph search algorithm. In the next section we will discuss the Viterbi decoder in more detail.

2.2.2 Time-synchronous Viterbi decoding

The Viterbi decoding algorithm can be regarded as a special case of dynamic programming applied to HMMs. The purpose of the decoding algorithm is to determine the state sequence $\mathbf{S}_1^T$ that has the highest likelihood of producing the observation sequence $\mathbf{X}_1^T$ given the HMM Φ_{lc}. When decoding algorithms process the observation sequence in increasing order of time ($t = 1, 2, \ldots, T$) we refer to such algorithms as *forward* decoders. When decoding algorithms process the observation sequence in decreasing order of time ($t = T, T-1, \ldots, 2, 1$) we refer to such algorithms as *backward* decoders. The Viterbi-

Figure 2.5: *Time-synchronous Viterbi decoding of an HMM, when the decoding is viewed as a graph search problem.*



beam decoder is conventionally defined to be a forward decoder.

We start by reviewing the standard algorithm as applied to first-order HMMs and then show how it is generalised to high-order HMMs. Viterbi-beam decoders differ from Viterbi decoders in that they prune the search graph to limit the number of evaluated state sequences. This will be discussed in more detail in section 3.3.3.

The first-order, best-path forward probability is defined to be

$$\overrightarrow{\delta}_t(i) = \max_{\mathbf{S}_1^{t-1}} \left[P(\mathbf{X}_1^t, \mathbf{S}_1^{t-1}, s_t = i | \Phi_{\text{lc}}) \right] \tag{2.7}$$

where $\overrightarrow{\delta}_t(i)$ is the probability of the most likely state sequence $\mathbf{S}_1^t$, which has generated the partial observation sequence $\mathbf{X}_1^t$ and ends at time t in state $s_t = i$, given the HMM Φ_{lc}. The back pointer $\overrightarrow{\Psi}_t^{\delta}(i)$ of state i at time t points to the state most likely to precede state i at time $t-1$. The following induction procedure is used to calculate $\overrightarrow{\delta}_t(i)$:

1. Initialisation: $(t = 1)$

$$\left.\begin{array}{l} \overrightarrow{\delta}_1(i) = \overrightarrow{a}_{0i} b_i(\mathbf{x}_1) \\ \overrightarrow{\Psi}^{\delta}_1(i) = 0 \end{array}\right\} i = 1, \ldots, N \tag{2.8}$$

2. Induction: $(t = 2, 3, \ldots, T)$

$$\left.\begin{array}{l} \overrightarrow{\delta}_t(j) = \left[\max\limits_i \overrightarrow{\delta}_{t-1}(i)\overrightarrow{a}_{ij}\right] b_j(\mathbf{x}_t) \\ \overrightarrow{\Psi}^{\delta}_t(j) = \arg\max\limits_i \left[\overrightarrow{\delta}_{t-1}(i)\overrightarrow{a}_{ij}\right] \end{array}\right\} j = 1, \ldots, N \tag{2.9}$$

3. Termination:

$$\begin{array}{l} \overrightarrow{\delta}_T = \max\limits_i \left[\overrightarrow{\delta}_T(i)\overrightarrow{a}_{i(N+1)}\right] \\ s^*_T = \arg\max\limits_i \left[\overrightarrow{\delta}_T(i)\overrightarrow{a}_{i(N+1)}\right] \end{array} \tag{2.10}$$

4. Backtracking:

$$\begin{array}{l} s^*_t = \overrightarrow{\Psi}^{\delta}_{t+1}(s^*_{t+1}),\ t = T-1, T-2, \ldots, 2, 1 \\ \mathbf{S}^* = (s^*_1, s^*_2, \ldots, s^*_T) \end{array} \tag{2.11}$$

2.2.2.1 Generalisation to high-order HMMs

We define the R^{th}-order, best-path forward probability as

$$\begin{aligned} & \overrightarrow{\delta}_t(i_2, i_3, \ldots, i_{R+1}) \\ = & \max_{\mathbf{S}_1^{t-R}} \left[P(\mathbf{X}_1^t, \mathbf{S}_1^{t-R}, s_{t-R+1} = i_2, s_{t-R+2} = i_3, \ldots, s_t = i_{R+1}|\Phi_{\text{lc}})\right] \end{aligned} \tag{2.12}$$

$\overrightarrow{\delta}_t(i_2, i_3, \ldots, i_{R+1})$ is the probability of the most likely state sequence $\mathbf{S}_1^t$, which has generated the partial observation sequence $\mathbf{X}_1^t$ and ends in the state sequence $\mathbf{S}_{t-R+1}^t = \{s_{t-R+1} = i_2, s_{t-R+2} = i_3, \ldots, s_{t-1} = i_R, s_t = i_{R+1}\}$ at time t, given the HMM Φ_{lc} (if we let $R = 2$ this definition of the best-path forward probability is equivalent to the definition of the second-order, best-path forward probability $\overrightarrow{\delta}_t(i_1, i_2)$ found in [24]). We define $\overrightarrow{\Psi}^{\delta}_{t-R+1}(i_2, i_3, \ldots, i_{R+1})$ to be the back pointer at time t of the state sequence $\{s_{t-R+1} = i_2, s_{t-R+2} = i_3, \ldots, s_{t-1} = i_R, s_t = i_{R+1}\}$ and denotes the most likely state to precede the state sequence at time $t - R$. The following induction procedure is used to calculate $\overrightarrow{\delta}_t(i_2, i_3, \ldots, i_{R+1})$:

1. Initialisation: $(t = 2, 3, \ldots, R;\ i_1, i_2, \ldots, i_R = 1, 2, \cdots, N)$

$$\begin{array}{l} \overrightarrow{\delta}_1(i_1) = \overrightarrow{a}_{0i_1} b_{i_1}(\mathbf{x}_1) \\ \overrightarrow{\delta}_t(i_1, i_2, \ldots, i_t) = \overrightarrow{\delta}_{t-1}(i_1, \ldots, i_{t-1})\overrightarrow{a}_{0i_1 \cdots i_t} b_{i_t}(\mathbf{x}_t) \\ \overrightarrow{\Psi}^{\delta}_1(i_1, i_2, \ldots, i_R) = 0 \end{array} \tag{2.13}$$

2. Induction: $(t = R+1, \ldots, T-1, T;\ i_2, i_3, \ldots, i_{R+1} = 1, \cdots, N)$

$$\begin{aligned}\overrightarrow{\delta}_t(i_2, i_3, \ldots, i_{R+1}) &= \left[\max_{i_1} \overrightarrow{\delta}_{t-1}(i_1, \ldots, i_R)\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}}\right] b_{i_{R+1}}(\mathbf{x}_t) \\ \overrightarrow{\Psi}^{\delta}_{t-R+1}(i_2, i_3, \ldots, i_{R+1}) &= \arg\max_{i_1}\left[\overrightarrow{\delta}_{t-1}(i_1, \ldots, i_R)\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}}\right]\end{aligned} \tag{2.14}$$

3. Termination:

$$\begin{aligned}\overrightarrow{\delta}_T &= \max_{i_1, i_2, \ldots, i_R}\left[\overrightarrow{\delta}_T(i_1, i_2, \ldots, i_R)\overrightarrow{a}_{i_1 i_2 \cdots i_R (N+1)}\right] \\ (s^*_{T-R+1}, \cdots, s^*_{T-1}, s^*_T) &= \arg\max_{i_1, i_2, \ldots, i_R}\left[\overrightarrow{\delta}_T(i_1, \ldots, i_R)\overrightarrow{a}_{i_1 i_2 \cdots i_R (N+1)}\right]\end{aligned} \tag{2.15}$$

4. Backtracking:

$$\begin{aligned}s^*_t &= \overrightarrow{\Psi}^{\delta}_{t+1}(s^*_{t+1}, s^*_{t+2}, \cdots, s^*_{t+R}),\ t = T-R, \ldots, 2, 1 \\ \mathbf{S}^* &= (s^*_1, s^*_2, \ldots, s^*_T)\end{aligned} \tag{2.16}$$

2.2.3 Pruning the decoding search space

The Viterbi-beam decoder uses pruning to limit the number of evaluated state sequences. Instead of retaining all candidates at every time frame, a threshold is used to keep only a subset of promising candidates.

2.2.3.1 Pruned decoding of first-order HMMs

In the case of time-synchronous, beam-pruning search algorithms of first-order HMMs, the subset of promising candidates is determined by calculating the probability of each state at the previous time instance, given the observations seen thus far. Less probable states at the previous time instance are omitted from the rest of the search at time t. For pruned searches in the forward direction, the decoder needs to calculate the probabilities at time $t-1$ i.e.

$$P(s_{t-1} = i|\mathbf{X}_1^{t-1}, \Phi_{\text{lc}}) = \frac{P(\mathbf{X}_1^{t-1}, s_{t-1} = i|\Phi_{\text{lc}})}{P(\mathbf{X}_1^{t-1}|\Phi_{\text{lc}})} \tag{2.17}$$

The probability $P(\mathbf{X}_1^{t-1}, s_{t-1} = i|\Phi_{\text{lc}})$ is called the forward probability by Rabiner and Juang [39] and is denoted by $\overrightarrow{\alpha}_{t-1}(i)$ and is used in the *Forward* algorithm when solving the evaluation problem. Using the forward probability, we can rewrite Eq. 2.17 as follows:

$$P(s_{t-1} = i|\mathbf{X}_1^{t-1}, \Phi_{\text{lc}}) = \frac{\overrightarrow{\alpha}_{t-1}(i)}{P(\mathbf{X}_1^{t-1}|\Phi_{\text{lc}})} \tag{2.18}$$

In order to keep the most probable states at time $t-1$ the decoder does not need to directly calculate $P(s_{t-1} = i|\mathbf{X}_1^{t-1}, \Phi_{\text{lc}})$, but instead can use the forward probability $\overrightarrow{\alpha}_{t-1}(i)$, as we are only interested in the ranking of the states and $P(s_{t-1} = i|\mathbf{X}_1^{t-1}, \Phi_{\text{lc}}) \sim \overrightarrow{\alpha}_{t-1}(i)$. When

using forward Viterbi-beam decoders to decode the HMM, the decoder does not compute the forward probability $\overrightarrow{\alpha}_{t-1}(i)$ as part of the search. However, according to Rabiner and Juang [39] the dynamic range of the forward probability $\overrightarrow{\alpha}_{t-1}(i)$ is usually very large and the best-path forward probability $\overrightarrow{\delta}_{t-1}(i)$ is usually the only significant term in the summation for the forward probability. Therefore, we assume that $\overrightarrow{\alpha}_{t-1}(i) \approx \overrightarrow{\delta}_{t-1}(i)$ and use the *best-path* forward probability $\overrightarrow{\delta}_{t-1}(i)$ instead of the forward probability.

When using beam-pruned forward decoding, the most probable states can be ranked by only using the best-path forward probability $\overrightarrow{\delta}_{t-1}(i)$, which is calculated as part of the search.

The pruning is implemented as a logarithmic beam-width B. All states, whose best-path forward probability does not fall within a certain threshold, are omitted from the rest of the search. For forward decoding the threshold is calculated as

$$e^{-B} \max_k \overrightarrow{\delta}_{t-1}(k), \ B \geq 0 \tag{2.19}$$

2.2.3.2 Pruned decoding of high-order HMMs

In the case of time-synchronous, beam-pruning search algorithms of R^{th}-order HMMs, the subset of promising candidates is determined by calculating the joint probability of sequences of states, ending at the previous time instance, given the observations seen thus far. Less probable sequences of states, ending at the previous time instance, are omitted from the rest of the search at time t. For pruned searches in the forward direction the decoder needs to calculate

$$\begin{aligned} P(s_{t-R} = i_1, \ldots, s_{t-1} = i_R | \mathbf{X}_1^{t-1}, \Phi_{\text{lc}}) &= \frac{P(s_{t-R} = i_1, \ldots, s_{t-1} = i_R, \mathbf{X}_1^{t-1} | \Phi_{\text{lc}})}{P(\mathbf{X}_1^{t-1} | \Phi_{\text{lc}})} \\ &= \frac{\overrightarrow{\alpha}_{t-1}(i_1, \ldots, i_R)}{P(\mathbf{X}_1^{t-1} | \Phi_{\text{lc}})} \end{aligned} \tag{2.20}$$

In order to keep the most probable sequence of states at time $t-1$ the decoder does not need to directly calculate $P(s_{t-R} = i_1, \ldots, s_{t-1} = i_R | \mathbf{X}_1^{t-1}, \Phi_{\text{lc}})$ but can instead use the forward probability $\overrightarrow{\alpha}_{t-1}(i_1, \ldots, i_R)$ as we are only interested in the ranking of the state sequences ending at time $t-1$, and $P(s_{t-R} = i_1, \ldots, s_{t-1} = i_R | \mathbf{X}_1^{t-1}, \Phi_{\text{lc}}) \sim \overrightarrow{\alpha}_{t-1}(i_1, \ldots, i_R)$. When using forward Viterbi-beam decoders to decode the HMM, the decoder does not compute the forward probability $\overrightarrow{\alpha}_{t-1}(i_1, \ldots, i_R)$, therefore the *best-path* forward probability $\overrightarrow{\delta}_{t-1}(i_1, \ldots, i_R)$ is used instead, since according to Rabiner and Juang [39] we can assume that $\overrightarrow{\alpha}_{t-1}(i_1, \ldots, i_R) \approx \overrightarrow{\delta}_{t-1}(i_1, \ldots, i_R)$. The forward Viterbi-beam pruning of high-order, left-context HMMs is very similar to the decoding of first-order HMMs, except that the high-order, best-path forward probability is used when pruning.

It is important to note that the beam-pruning used to limit the search space of the Viterbi-beam decoder is based on *partial* observation sequences. It can be argued that state pruning based on the complete observation sequence should result in a smaller

search space and thus a computationally less expensive decoder. This idea will be further explored in Chapter 3.

2.2.4 Evaluating forward Viterbi-beam decoding of HMMs

2.2.4.1 Corpus

We evaluate the computational expense of decoding left-context HMMs using the CallFriend [8] speech corpus. Left-context, high-order HMMs were trained with the 40 recordings which form the German training set of CallFriend. These HMMs were then evaluated on both the 40 recordings which form the English development set of CallFriend and the 40 recordings which form the German development set. The evaluation performed on the German development set represents the scenario where the training and evalution conditions are matched. The evaluation performed on the English development set represents the scenario where there is a mismatch between the training and evaluation conditions. Due to memory constraints whole recordings were not used during training and decoding. The recordings used for training were divided into 60s segments. The recordings used for evaluation were divided into 20s segments for a total of 1410 segments that were decoded using an HMM of each order.

2.2.4.2 Training of high-order HMMs

Instead of directly training R-th order HMMs on the data, we use the method of Fast Incremental Training (FIT) [12]. This has the advantage of also training all the HMMs with order lower than R. The density functions of an N emitting-state, first-order HMM are initialised by first dividing the training data into N regions utilising vector quantisation. The means of the N regions are then used to initialise diagonal covariance Gaussian output density functions. The transition probabilities are initialised to be equal. The FIT algorithm trains the R-th order HMM by starting with a fully-connected, first-order HMM and increasing the order incrementally. Between each increase in order the HMM parameters are trained using the Viterbi-re-estimation[2] algorithm. State transition probabilities which drop below a threshold of 10^{-5} are removed from the HMM. The initial, fully-connected, first-order HMM has a total of 40 emitting states and 1680 state transition probabilities.

[2]The Viterbi-re-estimation algorithm is similar to the Baum-Welch re-estimation algorithm except that it only considers the most likely state sequence instead of considering the summation of all state sequences. This re-estimation is suboptimal, but has the advantage of being significantly faster.

2.2.4.3 Measuring decoder performance

We define the search cost C_s as the number of transition probabilities evaluated during decoding, and us it as an implementation independent measure of the performance of the decoder. The number of pdfs are independent of the order of the HMM. Over the 1410 segments decoded we compute the average number of evaluated transition probabilities. We compute the normalised search cost $C_{s,n}$ by normalising the number of evaluated transition probabilities with the maximum number of transition probabilities that the standard Viterbi decoder would evaluate. This maximum number of evaluated transition probabilities is equal to the total number of transition probabilities of the HMM, multiplied by the length of the feature vector T, thus the normalised search cost can be computed as

$$C_{s,n} = \frac{\text{number of evaluated transitions}}{(\text{number of transitions of HMM}) \times T} \tag{2.21}$$

As the number of transition probabilities increases exponentially with the order of the HMM, we expect the number of transitions evaluated by standard Viterbi-decoding to also increase exponentially. We measure the search cost for forward Viterbi-beam decoding.

As the decoders are not guaranteed to find an optimal state sequence, the following three results occur when a segment is decoded:

(a) the decoder does not find any state sequence,

(b) the decoder finds a state sequence, but it is not the optimal state sequence, and

(c) the decoder finds the optimal state sequence.

The segment is regarded as being correctly decoded only when the decoding results in (c).

Result (a) occurs (the decoder does not find any state sequence) because the width of the beam was too narrow, resulting in a too small search space in which no single state sequence survives the beam-pruning. A decoder can recover from this error by simply decoding the segment again, this time with a wider beam-width, so that a larger search space is examined. The number of transitions evaluated when the decoder fails to find a state sequence for a segment is added to the total number of transitions required to decode that particular segment.

The only practical method for determining whether a decoded state sequence is optimal, is by comparing it with a reference state sequence. It is therefore very difficult for the decoder to distinguish whether result (b) or (c) occurred. Thus the decoders used in this dissertation cannot recover from errors as a result of (b).

2.2.4.4 Results

We examine the decoding performance of the forward Viterbi-beam decoders when decoding high-order HMMs. For this experiment the following two beam-pruning configurations are investigated, as the order of the HMMs is increased:

- The beam-width is set to a constant width of $B = 20.0$.
- The minimum integer beam-width, for which the decoder correctly decodes all segments, is determined.

Constant beam-width

Fig. 2.6(a) shows the search cost (the number of transition probabilities evaluated) when decoding an ergodic, left-context HMM with a constant beam-width of $B = 20.0$. The cases shown are Viterbi decoding (the special case of Viterbi-beam decoding with an infinite beam) and *forward* Viterbi-beam decoding for both the German (matched) and English (unmatched) development sets. Table 2.1 tabulates the accuracy of decoding the different order HMMs, when using a constant beam. We can see that for a constant beam the decoding accuracy remains fairly consistently above 96%.

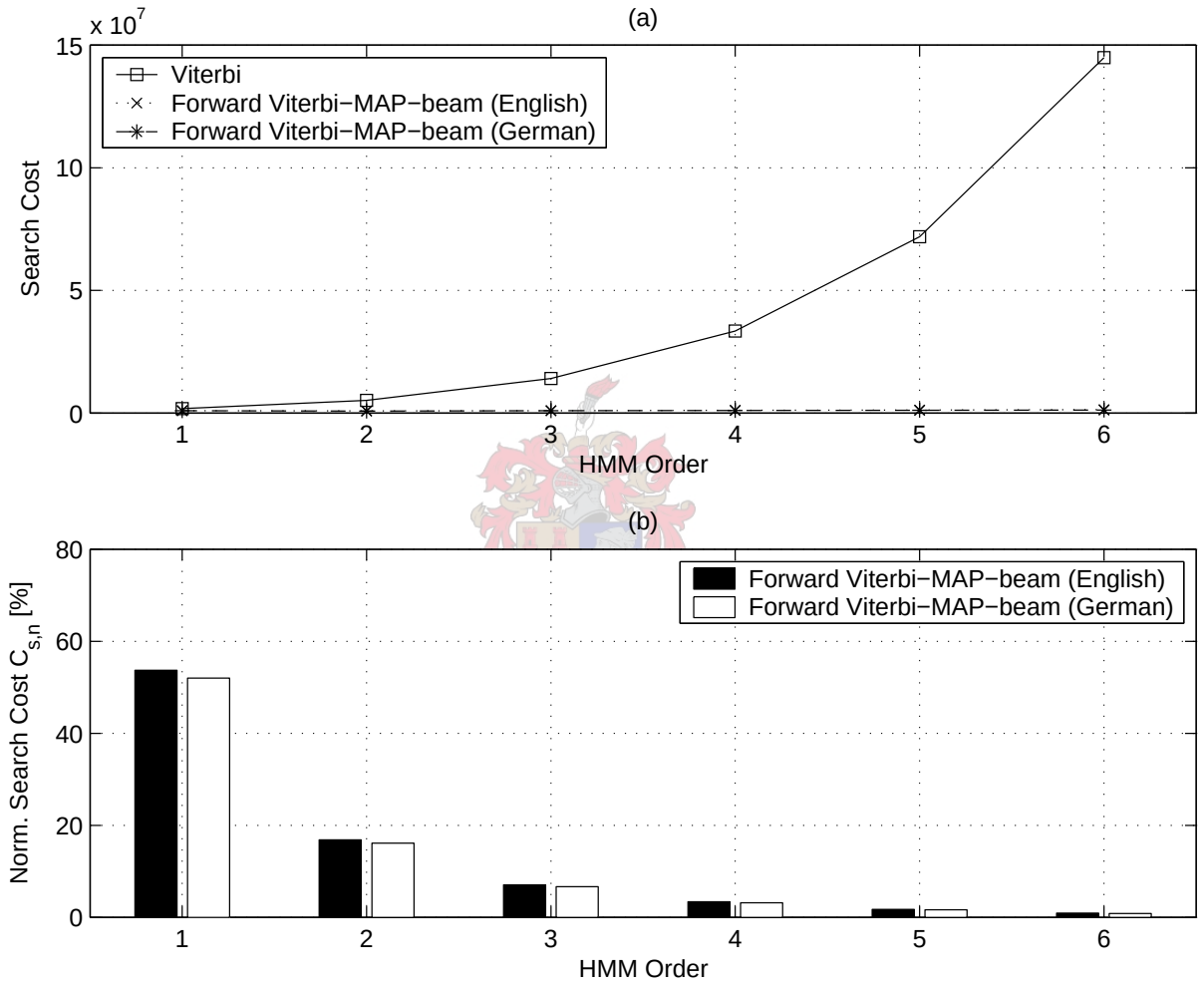
Minimum computational expense of decoder when correctly decoding all segments

Fig. 2.7(a) shows the number of transition probabilities evaluated when decoding an ergodic, left-context HMM and increasing the beam-width until all segments are decoded correctly. The cases shown are Viterbi decoding (the special case of Viterbi-beam decoding with an infinite beam) and *forward* Viterbi-beam decoding for both the German (matched)

Table 2.1: *The computational expense of the Viterbi-beam decoder during the forward decoding of high-order, left-context HMMs, when the decoder is using a constant beam-width of $B = 20.0$.*

	Search cost C_s		**Norm. cost $C_{s,n}$ [%]**		**Decoding Accuracy [%]**	
Order	**English**	**German**	**English**	**German**	**English**	**German**
1	993,708	961,853	53.71	51.99	98.37	99.86
2	870,291	833,683	16.85	16.14	96.60	99.65
3	996,221	942,520	7.06	6.68	97.52	99.65
4	1,134,157	1,059,379	3.39	3.17	97.87	99.31
5	1,252,957	1,161,283	1.74	1.62	97.16	98.75
6	1,356,728	1,232,086	0.94	0.86	96.31	97.50

Figure 2.6: *(a) The search cost (C_s) of the Viterbi-beam decoder, during the forward decoding of high-order, left-context HMMs, when the decoder is using a constant beam-width of $B = 20.0$. (b) The normalised search cost ($C_{s,n}$) of the Viterbi-beam decoder, during the forward decoding of high-order, left-context HMMs, when the decoder is using a constant beam-width of $B = 20.0$.*

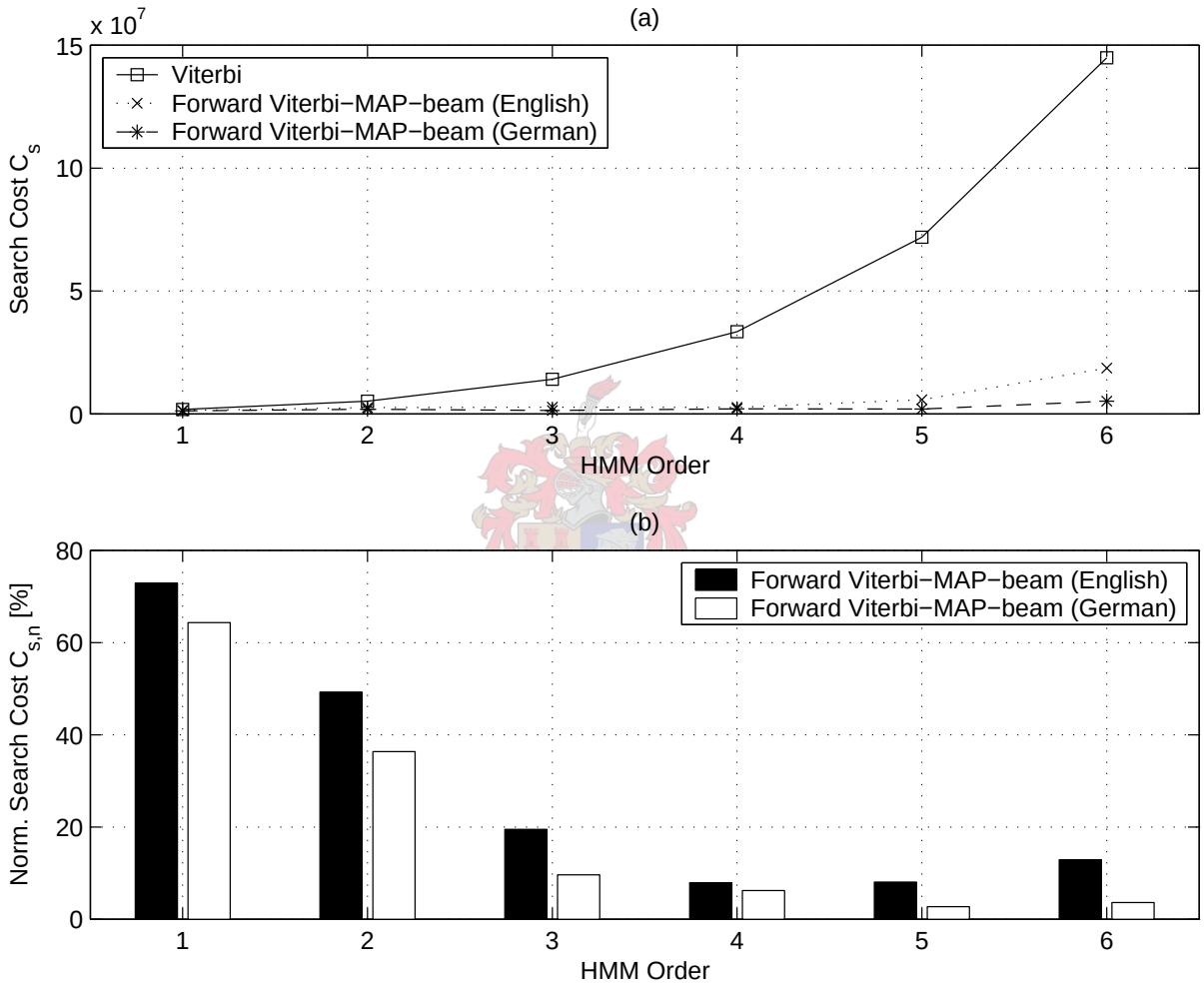


and English (unmatched) development sets. Table 2.2 tabulates the minimum integer beam-width required to correctly decode all segments.

2.2.4.5 Discussion

It is usually accepted that the computational expense of using high-order HMMs will increase exponentially with the order of the HMMs. This is true when Viterbi decoding is used without any beam-pruning, which is guaranteed to always obtain the optimal state sequence. However, the results show that a significant reduction in the computational expense of high-order HMMs can be achieved when using the Viterbi-beam decoder. The

Figure 2.7: *(a) The minimum search cost of the Viterbi-beam decoder, during the forward decoding of high-order, left-context HMMs, with beams set wide enough to decode all segments correctly. (b) The normalised search cost ($C_{s,n}$) of the Viterbi-beam decoder, during the forward decoding of high-order, left-context HMMs, with beams set wide enough to decode all segments correctly.*



disadvantage is that the admissibility of the Viterbi decoder is sacrificed when pruning is employed to reduce the search space.

When the beam-width is kept constant, it appears that the number of transitions that need to be evaluated only increases approximately linearly with the order of the HMM. The decoding accuracy also seems to remain fairly consistent and this is the scenario when Viterbi-beam decoders are employed in practical pattern recognition systems.

When the decoder manages to decode all segments correctly of the German development set (the matched scenario), the required beam-width seems to remain in the range between 24.0 and 35.0, and is the largest when the order is $R = 5$. When the decoder

Table 2.2: *The minimum computational expense of the Viterbi-beam decoder during the forward decoding of high-order, left-context HMMs, with beams set wide enough to decode all segments correctly.*

	Search cost C_s		**Norm. cost $C_{s,n}$ [%]**		**Beam-width B**	
Order	**English**	**German**	**English**	**German**	**English**	**German**
1	1,348,668	1,190,010	72.90	64.32	29.0	25.0
2	2,544,329	1,878,172	49.25	36.36	40.0	33.0
3	2,748,162	1,356,217	19.53	9.60	33.0	24.0
4	2,645,305	2,071,982	7.91	6.20	29.0	27.0
5	5,778,786	1,965,489	8.04	2.74	37.0	25.0
6	18,659,796	5,201,435	12.88	3.62	54.0	35.0

manages to decode all segments correctly of the English development set (the unmatched scenario), the required beam-width seems to remain in the range between 30.0 and 40.0, but significantly increases when the order is larger than $R = 5$. We suspect this is a result of inadequate training data for the high-order HMMs, resulting in poorly estimated high-order transition probabilities. These poorly estimated transition probabilities could result in the decoder requiring a much wider beam-width in order to find all the optimal state sequences. Another possibility is that the mismatch between training and evaluation conditions could be the cause of the increase in required beam-width, since the English development set generally resulted in a wider beam-width in order to final all optimal state sequences. We do note that this method of increasing the beam-width until all segments are correctly decoded is sensitive to outliers.

2.3 Summary

In this chapter we have reviewed the standard theory of hidden Markov models. We discussed the concept of first-order equivalents of high-order HMMs, a technique which allows us to always use the standard first-order algorithms when working with high-order HMMs. This removes the need for separate versions of the standard algorithms for each different order HMM.

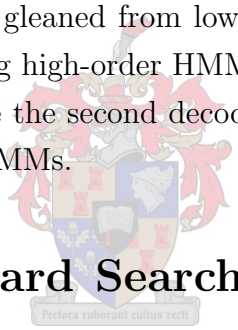
We then discussed different types of decoding algorithms and presented the standard decoding algorithm for first-order HMMs, namely the Viterbi algorithm. We continued by expanding the Viterbi algorithm to include the decoding of general R^{th}-order HMMs. We also discussed the beam-pruning of the decoding search space, which is performed by only considering the most likely states to occur at each time instance. We noted that standard Viterbi-beam decoding bases its state pruning on *partial* observation sequences.

Lastly, we measured the computational expense of forward Viterbi-beam decoding of high-order HMMs. We noted that when the pruning beam-width is kept constant, the decoding accuracy remains fairly consistent. However, when the beam-width is increased until all segments are decoded correctly, a certain order of HMM is reached where the minimum required beam-width increases significantly. We noted that this could be a result of inadequate training data for high-order HMMs, but could also be caused by a mismatch between the training and evaluation conditions. We now have a decoding performance base-line to which we can compare the decoders we will propose in the next chapter.

Chapter 3

Forward-Backward Search of high-order HMMs

In this chapter we discuss the two newly proposed decoders. Both of the decoders attempt to incorporate information gleaned from low-order HMMs in order to reduce the computational expense of decoding high-order HMMs. The first decoder is and adaption of the Viterbi-beam decoder while the second decoder is the application of A* search to the task of decoding high-order HMMs.



3.1 Forward-Backward Search based Viterbi decoding

In the previous chapter we mentioned that the Viterbi-beam decoder bases its beam-pruning on partial observation sequences. We also mentioned that a decoder that bases its state pruning on complete observation sequences might result in a smaller search space, while still managing to decode the optimal state sequences. In the first part of this chapter we examine pruning based on complete observations. The development of this idea led to the first of our proposed decoders: the Forward-Backward Search based Viterbi-beam decoder (FBS-Viterbi-beam decoder).

3.1.1 Description

The FBS-Viterbi-beam decoder is very similar to the standard Viterbi-beam decoder. The difference is that we wish to replace the beam-pruning strategy based on partial observations, with a beam-pruning strategy that is based on complete observations. Since we are using the complete observation this causes the FBS-Viterbi-beam decoder to be a two-pass decoder. During the first pass of the decoder a heuristic function is calculated that is used to prune the search graph. The second pass of the decoder is identical to

the standard Viterbi-beam decoder. We can therefore use the same equations for the calculation of the R^{th}-order, best-path forward probability as is used by the standard Viterbi-beam decoder (as discussed in Section 2.2.2).

The FBS-Viterbi-beam decoder is still a time-synchronous search algorithm. We are simply guiding the state pruning by calculating a heuristic function. Thus, the FBS-Viterbi-beam decoder is not an informed search, since it still grows the search graph in a time-synchronous manner and does not *select* nodes for *expansion* based on a heuristic function.

It is important to note that, whether the Viterbi-beam decoder prunes the search space based on partial or complete observation sequences, the same search graph is being explored. The decoders only differ in the information that is used when choosing which states to omit from the rest of the search. In the next section we will develop the equations that are required to prune the search graph based on complete observations.

3.1.2 A state pruning strategy based on complete observations

Instead of only using the partial observation sequence to determine the probability of a state at a certain time frame, we propose a beam-pruning strategy based on the complete observation sequence. The proposed decoder should determine the subset of promising candidates by calculating the joint probability of sequences of states, ending at the previous time instance, given the *complete* observation sequence $\mathbf{X}_1^T$. Thus, given the complete observation sequence, the decoder needs to calculate the state sequence probability

$$\begin{aligned} & \overrightarrow{\gamma}_t(i_1, i_2, \ldots, i_R) \\ = \; & P(s_{t-R} = i_1, \ldots, s_{t-1} = i_R | \mathbf{X}_1^T, \Phi_{\text{R,lc}}) \\ = \; & \frac{\overrightarrow{\alpha}_{t-1}(i_1, i_2, \ldots, i_R) \overrightarrow{\beta}_{t-1}(i_1, i_2, \ldots, i_R)}{P(\mathbf{X}_1^T | \Phi_{\text{R,lc}})} \end{aligned} \tag{3.1}$$

where $\overrightarrow{\alpha}_{t-1}(i_1, i_2, \ldots, i_R)$ is defined to be the R^{th}-order forward probability $P(s_{t-R} = i_1, \ldots, s_{t-1} = i_R, \mathbf{X}_1^{t-1} | \Phi_{\text{R,lc}})$ and $\overrightarrow{\beta}_{t-1}(i_1, i_2, \ldots, i_R)$ is defined to be the R^{th}-order backward probability $P(s_{t-R} = i_1, \ldots, s_{t-1} = i_R | \mathbf{X}_t^T, \Phi_{\text{R,lc}})$ (which is just the general case of the first-order forward and backward probabilities defined in [38, 39, 40]).

In order to keep the most probable sequence of states at time $t-1$ the decoder does not need to directly calculate $\overrightarrow{\gamma}(i_1, i_2, \ldots, i_R)$, but can instead use the R^{th}-order forward probability and backward probability $\overrightarrow{\alpha}_{t-1}(i_1, \ldots, i_R)$ and $\overrightarrow{\beta}_{t-1}(i_1, \ldots, i_R)$, as we are only interested in the ranking of the state sequences ending at time $t-1$, and $P(s_{t-R} = i_1, \ldots, s_{t-1} = i_R | \mathbf{X}_1^T, \Phi_{\text{lc}}) \sim \overrightarrow{\alpha}_{t-1}(i_1, \ldots, i_R) \overrightarrow{\beta}_{t-1}(i_1, \ldots, i_R)$. Assuming that $\overrightarrow{\alpha}_{t-1}(i_1, \ldots, i_R)$ and $\overrightarrow{\beta}_{t-1}(i_1, \ldots, i_R)$ are equally expensive to calculate, the decoder now has to perform double the amount of calculations. Since low-order HMMs have significantly less parameters, computing a backward probability on a low-order HMM should

be significantly less expensive. Therefore, instead of using the computationally expensive R^{th}-order backward probability, we propose that the decoder could use the $(R-K)$-order backward probability $\overrightarrow{\beta}_{t-1}(i_{K+1},\ldots,i_R)$ where $K<R$. Thus the decoder would base its pruning on the following equation:

$$\begin{aligned}\overrightarrow{\gamma}(i_1,i_2,\ldots,i_R) &\sim P(s_{t-R}=i_1,\ldots,s_{t-1}=i_R,\mathbf{X}_1^{t-1},\Phi_{\text{R,lc}})\times\\ &\quad P(\mathbf{X}_t^T|s_{t-R+K}=i_{K+1},\ldots,s_{t-1}=i_R,\Phi_{\text{R-K,lc}})\\ &= \overrightarrow{\alpha}_{t-1}(i_1,\ldots,i_R)\overrightarrow{\beta}_{t-1}(i_{K+1},\ldots,i_R) \qquad (3.2)\end{aligned}$$

However, when decoding the HMM using a forward Viterbi-beam decoder, the decoder does not compute the forward probability $\overrightarrow{\alpha}_{t-1}(i_1,\ldots,i_R)$, therefore the R^{th}-order *best-path* forward probability $\overrightarrow{\delta}_{t-1}(i_1,\ldots,i_R)$ is used instead, since we assume that $\overrightarrow{\alpha}_{t-1}(i_1,\ldots,i_R)\approx\overrightarrow{\delta}_{t-1}(i_1,\ldots,i_R)$.

In the same manner, let us define the R^{th}-order, best-path *backward* probability as

$$\overrightarrow{\epsilon}_t(i_1,i_2,\ldots,i_R)=\max_{\mathbf{S}_{t+1}^T}\left[P\left(\mathbf{X}_{t+1}^T,\mathbf{S}_{t+1}^T|s_{t-R+1}=i_1,\ldots,s_t=i_R,\Phi_{\text{lc}}\right)\right] \qquad (3.3)$$

The $(R-K)$-order, best-path backward probability $\overrightarrow{\epsilon}_{t-1}(i_{K+1},\ldots,i_R)$ could be used to replace the $(R-K)$-order, backward probability $\overrightarrow{\beta}_{t-1}(i_{K+1},\ldots,i_R)$, once again assuming that $\overrightarrow{\beta}_{t-1}(i_{K+1},\ldots,i_R)\approx\overrightarrow{\epsilon}_{t-1}(i_{K+1},\ldots,i_R)$.

Thus, given the complete observation and guided by a $(R-K)$-order HMM, the estimate of the likelihood of a state sequence occurring at a specific time can be written as:

$$\begin{aligned}\hat{\overrightarrow{\gamma}}_{t-1}(i_1,i_2,\ldots,i_R) &= \overrightarrow{\alpha}_{t-1}(i_1,i_2,\ldots,i_R)\hat{\overrightarrow{\beta}}_{t-1}(i_1,i_2,\ldots,i_R)\\ &= \overrightarrow{\alpha}_{t-1}(i_1,i_2,\ldots,i_R)\overrightarrow{\beta}_{t-1}(i_{K+1},\ldots,i_R)\\ &\approx \overrightarrow{\delta}_{t-1}(i_1,i_2,\ldots,i_R)\overrightarrow{\epsilon}_{t-1}(i_{K+1},\ldots,i_R) \qquad (3.4)\end{aligned}$$

Eq. 3.4 forms the basis of the state pruning strategy used by the proposed FBS-Viterbi-beam decoder.

3.1.3 Calculation of the heuristic function

The heuristic function used to estimate the state likelihood $\hat{\overrightarrow{\gamma}}_{t-1}(i_1,i_2,\ldots,i_R)$ requires the calculation of the best-path, *backward* probability of the $(R-K)$-order HMM, given the state sequence $\mathbf{X}_{t-1}^T$. It should be possible to efficiently calculate this backward probability, if we use a Viterbi-beam decoder that starts decoding from the rear of the observation sequence. We refer to such a decoder as a *backward Viterbi*-beam decoder and will discuss the decoder in more detail in Section 3.3. In the following section we will discuss the derivation of the $(R-K)$-order, left-context HMM, on which the backward decoding will be performed.

3.1.3.1 Derivation of the $(R-K)$-order, left-context HMM

The estimated best-path backward probability, calculated using the $(R-K)$-order, left-context HMM, must be approximately equal to the actual best-path, backward probability calculated using the R^{th}-order, left-context HMM. When deriving the low-order, left-context HMM we can make changes to the set of pdfs or to the state transition probabilities. By requiring that the $(R-K)$-order HMM uses the same set of pdfs as the R^{th}-order HMM, we can ensure that the estimated, best-path backward probability will only differ with respect to the state transition probabilities, given a specific partial state sequence.

Deriving $(R-K)$-order state transition probabilities directly from the R^{th}-order state transition probabilities is not a trivial exercise. However, the R^{th}-order *conditional* state transition probabilities can be written in terms of *joint* state transition probabilities:

$$\begin{aligned}\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}} &= P(s_t = i_{R+1} | s_{t-1} = i_R, \ldots, s_{t-R+1} = i_2, s_{t-R} = i_1) \\ &= \frac{P(s_{t-R} = i_1, s_{t-R+1} = i_2, \ldots, s_t = i_{R+1})}{P(s_{t-R} = i_1, s_{t-R+1} = i_2, \ldots, s_{t-1} = i_R)} \end{aligned} \tag{3.5}$$

If we had access to the R^{th}-order *joint* state probability, it would be possible to calculate the $(R-K)$-order conditional state probability as:

$$\begin{aligned}\overrightarrow{a}_{i_{K+1} i_{K+2} \cdots i_{R+1}} &= P(s_t = i_{R+1} | s_{t-1} = i_R, \ldots, s_{t-R+1+K} = i_{K+2}, s_{t-R+K} = i_{K+1}) \\ &= \frac{P(s_{t-R+K} = i_{K+1}, s_{t-R+K+1} = i_{K+2}, \ldots, s_t = i_{R+1})}{P(s_{t-R+K} = i_{K+1}, s_{t-R+K+1} = i_{K+2}, \ldots, s_{t-1} = i_R)} \end{aligned}$$

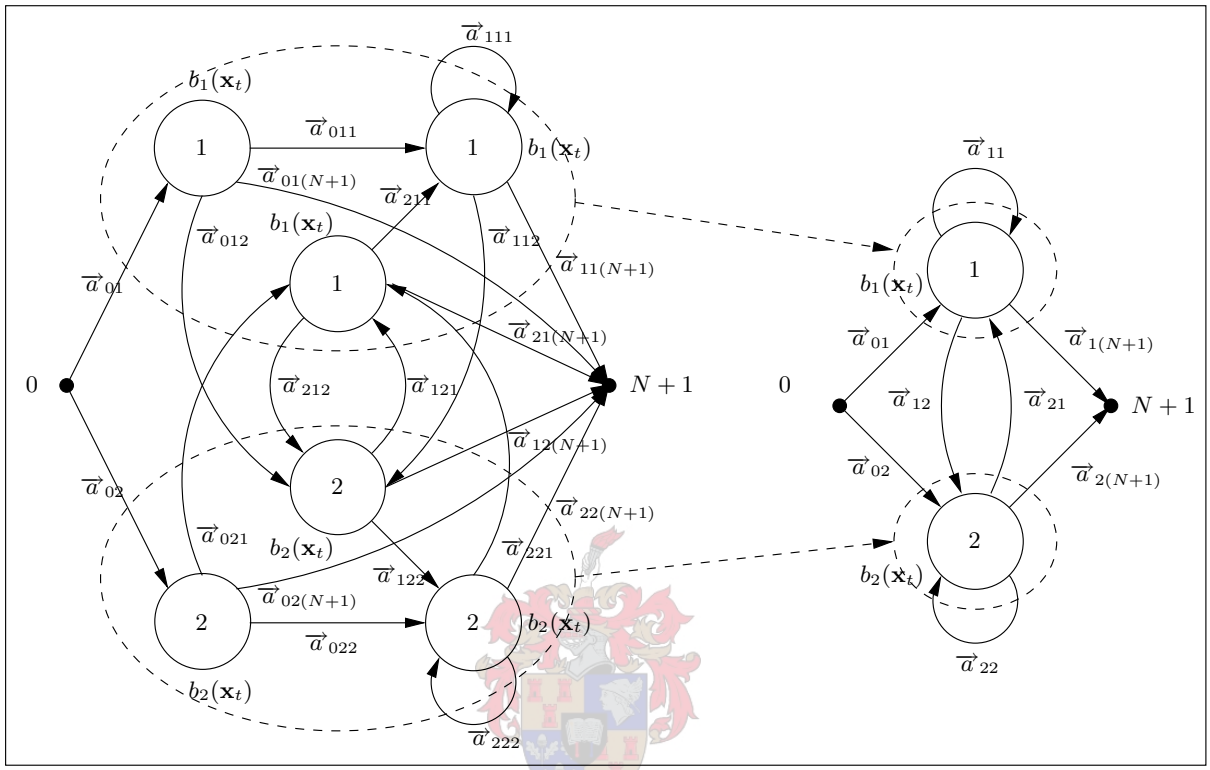
with

$$\begin{aligned}&P(s_{t-R+K} = i_{K+1}, s_{t-R+K+1} = i_{K+2}, \ldots, s_t = i_{R+1}) \\ = &\sum_{i_1, i_2, \cdots, i_K} P(s_{t-R} = i_1, s_{t-R+1} = i_2, \ldots, s_t = i_{R+1}), \\ &\text{with } i_1, i_2, \ldots, i_K \in \{0, \cdots, N\} \end{aligned} \tag{3.6}$$

These joint state probabilities are estimated during the training of the R^{th}-order HMM. Since the HMM state transition parameters are typically stored as conditional state transition probabilities, we derive all the $(R-K)$-order HMMs (with $K = 1, 2, \ldots, R-1$) when training the R^{th}-order HMM[1]. The derivation of the $R-K$-order HMM is illustrated in Fig. 3.1, using first-order equivalent HMMs. In the figure it can be seen how three first-order equivalent states of the second-order HMM map to a single state in the derived first-order HMM. The second-order HMM has a total of 20 state transition probabilities, while the first-order HMM has only 8 state transition probabilities.

[1]It would be possible to derive the $(R-K)$-order HMM during decoding, if the HMM state transition parameters were stored as joint state probabilities.

Figure 3.1: *The derivation of a first-order HMM from a second-order HMM (which is shown in the figure by its first-order equivalent HMM).*



In conclusion, by backward Viterbi-beam decoding the $(R-K)$-order, left-context HMM, we can estimate the $(R-K)$-order, best-path backward probability $\overrightarrow{\epsilon}_{t-1}(i_{K+1},\ldots,i_R)$. This backward probability can be combined with the R^{th}-order, best-path forward probability $\overrightarrow{\alpha}_{t-1}(i_1,\ldots,i_R)$ to give an estimate of the likelihood of a state occurring at time $t-1$. Since this combined likelihood is based on the complete observation sequence, it should result in a smaller search graph, and therefore less computationally expensive decoding.

3.2 Forward-Backward-based A* decoding

In the second part of this chapter we will discuss the application of the A* search algorithm to the task of decoding high-order HMMs.

Even though the Forward-Backward-based Viterbi-beam decoder is a two-pass decoder, it still explores the search space in a time-synchronous manner. Time-asynchronous A* search is another search method capable of incorporating information from low-order HMMs. A* search performs the same decoding task as the Viterbi algorithm; the difference is that the Viterbi algorithm is a dynamic programming algorithm, while A* search

is a graph search algorithm. A significant portion of the A* search discussion will be spent on the development of a novel, admissible heuristic, which is used to guide the A* search. For an in-depth discussion of A* search refer to [16] and [33].

3.2.1 Description of A* search

Given an HMM Φ, A* search is a graph search procedure used to find the state sequence (denoted by $\mathbf{S}^*$) most likely to have generated the observation sequence $\mathbf{X}_1^T$. The search graph G consists of a set of nodes. Certain pairs of nodes in the graph are connected by arcs, and these arcs are directed from one member of the pair to the other. The arc is directed from the parent node to the successor node. The arcs represent the cost of making a transition from one node to another. The root node of G has no parent and is called the root node n_s. The purpose of A* search is to find the best path from the root node to a given node n_g called the goal node. This is done by starting at the root node and growing the search graph G by successively expanding nodes until the goal node is reached. A* is typically used to find the path with the minimum cost, but when decoding HMMs, we are looking for the most likely state sequence, given the observation sequence $\mathbf{X}_1^T$.

Each node n in the graph G has the following attributes:

- The cost function $g(n)$, which is the accumulated probability of the best partial path from the root node n_s to the node n.
- The heuristic function $h(n)$, which is the estimated probability of the remaining best path from the node n to the goal node n_g.
- The evaluation function $f(n) = g(n)h(n)$, which is the estimated total probability of the best path going through node n.
- $p(n)$ is the parent node of node n.
- $M(n)$ is the successor nodes of node n.

The A* search maintains two lists: OPEN, which stores the nodes waiting to be selected for expansion; and CLOSE, which stores the already expanded nodes. The A* search procedure is summarised in the text below.

1. Initialisation: Create a search graph G. Put the root node n_s in the OPEN list and create an initially empty CLOSE list.
2. LOOP: If OPEN is empty, terminate with failure.
3. Select and remove the first node n from OPEN and put n in on CLOSE.
4. If $n = n_g$, terminate successfully with the state sequence obtained by backtracking along the parent pointers from n to n_s in G.
5. Expand n, generating the set of successors nodes $M(n)$.
6. $\forall v \in M(n)$:
   (a) If $v \in$ OPEN: Denote $v \in M(n)$ by v_M and denote $v \in$ OPEN by v_{OPEN}. If the accumulated probability of the new node is bigger than that of the node in OPEN i.e. $g(v_M) > g(v_{\text{OPEN}})$:
      i. Redirect the parent pointer of v_{OPEN} to $p(v_{\text{OPEN}}) = n$.
      ii. Adjust the accumulated probability of v_{OPEN} to $g(v_{\text{OPEN}}) = g(v_M)$ and recalculate $f(v_{\text{OPEN}})$
   (b) If $v \in$ CLOSE: Denote $v \in M(n)$ by v_M and denote $v \in$ CLOSE by v_{CLOSE}. If the accumulated probability of the new node is bigger than that of the node in CLOSE i.e. $g(v_M) > g(v_{\text{CLOSE}})$:
      i. Redirect the parent pointer of v_{OPEN} to $p(v_{\text{OPEN}}) = n$.
      ii. Adjust the accumulated probability of v_{CLOSE} to $g(v_{\text{CLOSE}}) = g(v_M)$ and recalculate $f(v_{\text{CLOSE}})$
      iii. For each descendent of $v \in G$, decide whether to redirect its parent pointer, and adjust its accumulated probability and evaluation function f.
   (c) If ($v \notin$ OPEN and $v \notin CLOSE$): Let $p(v) = n$ and put v onto OPEN.
7. Reorder the OPEN list in increasing order of the evaluation function $f(n)$. Goto (2).

3.2.2 Admissibility of A* search

Let $h^*(n)$ denote the actual probability of the best path from the node n to the goal node n_g. It is usually stated in the literature that A* search is admissible if $h(n)$ is an under-estimate (or lower-bound) of $h^*(n)$. In such cases the A* search is used to find the path with the minimum "cost". When decoding HMMs, we want the path with the maximum

probability. Thus, the A* decoding search of HMMs is admissible if the heuristic function $h(n)$ is an over-estimate of $h^*(n)$ i.e. $h(n) \geq h^*(n)$.

3.2.3 A* decoding of first-order HMMs

In the case of first-order HMMs, a node n in the graph G represents an HMM state i occurring at a certain time t. We therefore define the node $n_t(i) \triangleq (s_t = i)$. The arc between two nodes $n_{t-1}(i)$ and $n_t(j)$ represents the probability "cost" $c\left[n_{t-1}(i), n_t(j)\right]$ of making a transition from state i at time $t-1$, to state j at time t, given the observation sequence $\mathbf{X}_1^T$. This cost can be calculated as:

$$c\left[n_{t-1}(i), n_t(j)\right] = \overrightarrow{a}_{ij} b_j(\mathbf{x}_t) \tag{3.7}$$

The set of successor nodes $M\left[n_{t-1}(i)\right]$, which is generated when the A* search expands node $n_{t-1}(i)$, is defined as:

$$M(n_{t-1}(i)) \triangleq \{n_t(j) | \overrightarrow{a}_{ij} \neq 0, j \in \{1, 2, \ldots, N+1\}\} \tag{3.8}$$

The cost function of the successor nodes is calculated as:

$$\begin{aligned} g\left[n_t(j)\right] &= g\left[n_{t-1}(i)\right] c\left[n_{t-1}(i), n_t(j)\right] \\ &= g\left[n_{t-1}(i)\right] \overrightarrow{a}_{ij} b_j(\mathbf{x}_t) \end{aligned} \tag{3.9}$$

When decoding a first-order HMM Φ_1 given the observation sequence $\mathbf{X}_1^T$, the exact estimate of the heuristic function of node $n_t(i)$ is simply the best-path backward probability i.e.

$$h^*\left[n_t(i)\right] = \overrightarrow{\epsilon}_t(i) \text{ with } i \in \{0, 1, \ldots, N+1\} \tag{3.10}$$

The first-order, best-path backward probability is defined to be

$$\overrightarrow{\epsilon}_t \triangleq \max_{\mathbf{s}_{t+1}^T} \left[P(\mathbf{X}_{t+1}^T | s_t = i, \Phi_{1,\text{lc}})\right] \tag{3.11}$$

3.2.4 A* decoding of high-order HMMs

In the case of R^{th}-order HMMs, a node n in the graph G represents a state sequence $(s_{t-R+1} = i_1, s_{t-R+2} = i_2, \ldots, s_t = i_R)$ ending in state i_R at time t. We therefore define the node $n_t(i_1, i_2, \ldots, i_R) \triangleq (s_{t-R+1} = i_1, s_{t-R+2} = i_2, \ldots, s_t = i_R)$. The arc between two nodes $n_a = n_{t-1}(i_1, i_2, \ldots, i_R)$ and $n_b = n_t(i_2, i_3, \ldots, i_{R+1})$ represents the probability "cost" $c\left[n_a, n_b\right]$ of making a transition, at time $t-1$, from state i_R to state i_{R+1} at time t, given the observation sequence $\mathbf{X}_1^T$ and the state sequence $(s_{t-R+1} = i_1, s_{t-R+2} = i_2, \ldots, s_{t-1} = i_{R-1})$. This cost can be calculated as:

$$c\left[n_a, n_b\right] = \overrightarrow{a}_{i_1 i_2 \cdots i_R i_{R+1}} b_{i_{R+1}}(\mathbf{x}_t) \tag{3.12}$$

When the A* search expands node n_a the set of successor nodes $M(n_a)$ is defined is:

$$M(n_a) \triangleq \left\{ n_t(i_2, i_3, \ldots, i_{R+1}) | \overrightarrow{a}_{i_1 i_2 \cdots i_R i_{R+1}} \neq 0, i_{R+1} \in \{1, 2, \ldots, N+1\} \right\} \tag{3.13}$$

The cost function of the successor nodes is calculated as:

$$\begin{aligned} g\left[n_b\right] &= g\left[n_a\right] c\left[n_a, n_b\right] \\ &= g\left[n_{t-1}(i_1, i_2, \ldots, i_R)\right] \overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}} b_{i_{R+1}}(\mathbf{x}_t) \end{aligned} \tag{3.14}$$

When decoding a R^{th}-order, left-context HMM $\Phi_{\text{R,lc}}$ given the observation sequence $\mathbf{X}_1^T$, the exact estimate of the heuristic function of node $n_t(i_1, i_2, \ldots, i_R)$ is the R^{th}-order, best-path backward probability i.e.

$$h^*\left[n_t(i_1, i_2, \ldots, i_R)\right] = \overrightarrow{\epsilon}_t(i_1, i_2, \ldots, i_R), \text{ with } i_1, i_2, \ldots, i_R = 0, 1, \ldots, N+1 \tag{3.15}$$

The R^{th}-order, best-path backward probability was defined in Eq. 3.3.

3.2.5 Calculation of the heuristic function

The exact estimate of the heuristic function can be obtained by using the backward Viterbi algorithm to compute the best-path backward probability of a R^{th} -order HMM

$$\Phi_{\text{R,lc}} = \left\{ \overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}}, b_{i_{R+1}}(\mathbf{x}_t) | i_1, \ldots, i_{R+1} \in \{0, 1, \ldots, N+1\} \right\}$$

for all states at all times. Computing this exact estimate unfortunately requires the same amount of work as simply using the Viterbi algorithm to perform the backward decoding.

As the computational complexity of the Viterbi-algorithm is dependent on the order of the HMM, decoding HMMs with a Markov order lower than R, should be computationally less expensive than directly decoding the R^{th}-order HMM. To calculate the heuristic function for the A* decoder, we therefore propose using $(R-K)$-order, left-context HMMs (where $K < R$), which are derived from the R^{th}-order HMM. This is similar to our proposal of using $(R-K)$-order HMMs in Section 3.1.3, except that when the A* search grows the search graph, it will use the heuristic to select which nodes to expand.

We cannot directly use the $(R-K)$-order HMMs derived in Section 3.1.3, since this will result in an inadmissible heuristic function. In the next section we show how $(R-K)$-order, pseudo HMMs can be derived from the R^{th}-order HMM. Calculating the heuristic function using these pseudo HMMs will result in an admissible A* search decoder.

3.2.5.1 Derivation of the $R-K$-order, left-context pseudo HMM

We want to derive an $(R-K)$ order pseudo-HMM,

$$\hat{\Phi}_{\text{R-K,lc}} = \left\{ \hat{\overrightarrow{a}}_{i_{K+1} \cdots i_{R+1}}, \hat{b}_{i_{R+1}}(\mathbf{x}_t) | i_{K+1}, \ldots, i_{R+1} \in \{0, 1, \ldots, N+1\} \right\},$$

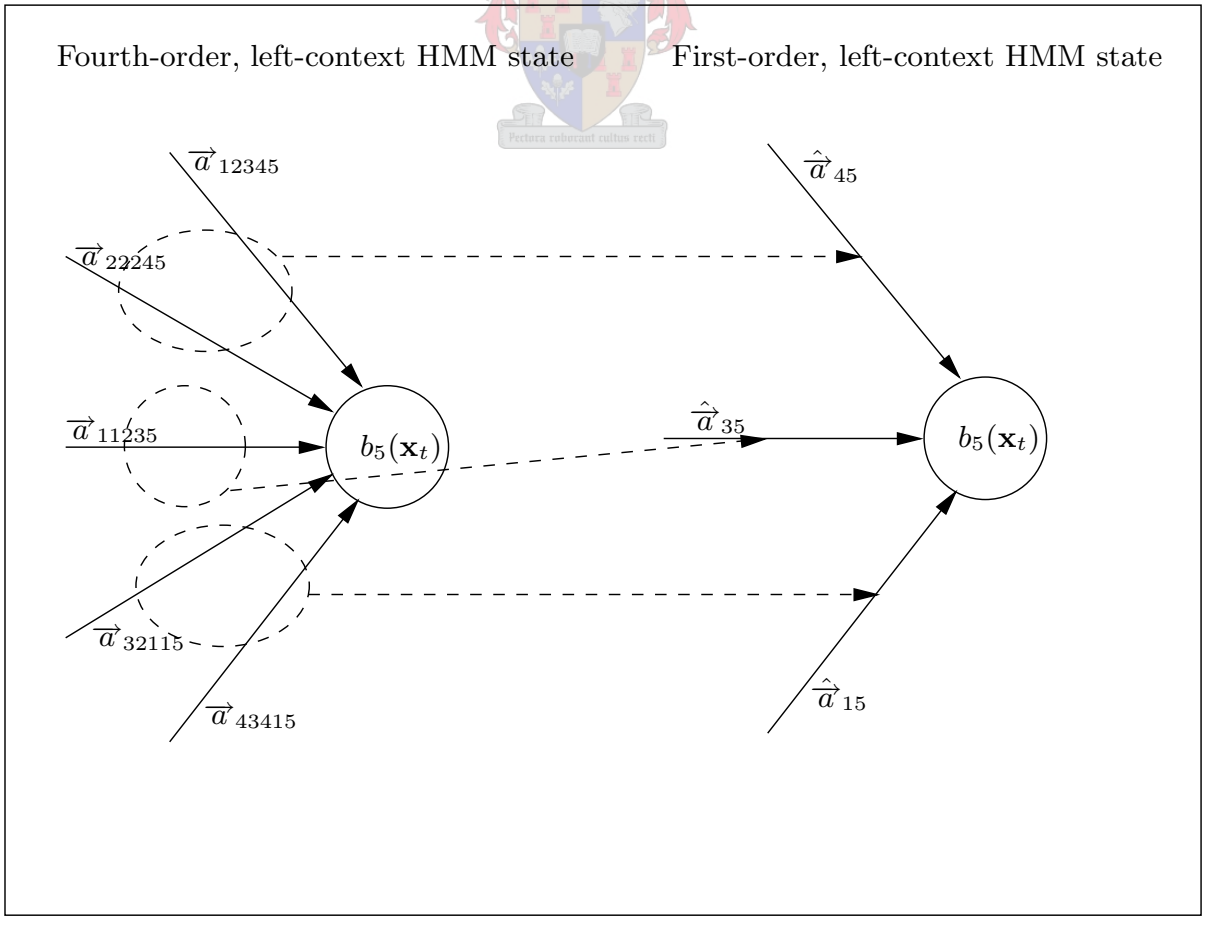
from the R^{th}-order HMM, which will result in an admissible A* search. The requirement for admissibility is that the heuristic function, for *any* node, must be an overestimate of the R^{th}-order, best-path backward probability. By requiring that the $(R-K)$-order pseudo HMM uses the same set of pdfs as the R^{th}-order HMM, we can ensure that the estimated, best-path backward probability $\hat{\overrightarrow{\epsilon}}_t(i_{K+2},\ldots,i_R)$ will only differ with respect to the state transition probabilities.

We can derive a $(R-K)$-order state transition probability from the R^{th}-order state transition probability by ignoring the identity of the first K states on which the R^{th}-order state transition probability is conditionally dependent. In other words, the set of R^{th}-order state transitions $\overrightarrow{a}_{i_1\cdots i_{R+1}}$ will all map to a single $(R-K)$-order state transition i.e.

$$\left\{\overrightarrow{a}_{i_1 i_2\cdots i_K i_{K+1}\cdots i_{R+1}}\right\} \Rightarrow \hat{\overrightarrow{a}}_{i_{K+1}\cdots i_{R+1}} \text{ where } i_1, i_2, \ldots, i_{R+1} \in \{0, 1, \ldots, N+1\} \quad (3.16)$$

This concept is illustrated in Fig. 3.2.

Figure 3.2: *An example of deriving a first-order transition from a fourth-order transition.*



Furthermore, we can ensure that the estimated $(R-K)$-order, best-path backward probability, is always an over-estimate of the R^{th}-order, best-path backward probability, by choosing the value of the $(R-K)$ state transition probability $\overrightarrow{a}_{i_{K+1}\cdots i_{R+1}}$ to be equal to the maximum probability occurring in the set of R^{th}-order state transitions mapping to that state i.e

$$\hat{\overrightarrow{a}}_{i_{K+1}\cdots i_{R+1}} = \max_{i_1,\cdots,i_K} \left[\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}}\right] \tag{3.17}$$

This will cause the derived pseudo-HMM $\hat{\Phi}_{\text{R-K,lc}}$ to not be a true HMM, since the transition probabilities leaving a certain state will not sum to unity i.e.

$$\sum_{i_{R+1}} \hat{\overrightarrow{a}}_{i_{K+1}\cdots i_{R+1}} = \sum_{i_{R+1}} \left[\max_{i_1,\cdots,i_K} \left[\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}}\right]\right] \neq 1 \tag{3.18}$$

This is not a concern, since the backward Viterbi-beam decoder does not require that the transition probabilities sum to unity.

Thus, the parameters of the left-context pseudo-HMM $\hat{\Phi}_{\text{R-K,lc}}$ is derived from the left-context HMM $\Phi_{\text{R,lc}}$ in the following manner:

$$\hat{\Phi}_{\text{R-K,lc}} \triangleq \begin{cases} \hat{\overrightarrow{a}}_{0 i_1 i_2 \cdots i_k} = \overrightarrow{a}_{0 i_1 i_2 \cdots i_k}, \ k = 1, 2, \ldots, K-1 \\ \hat{\overrightarrow{a}}_{i_{K+1}\cdots i_{R+1}} = \max\limits_{i_1,\cdots,i_K} \left[\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}}\right] \\ \hat{b}_{i_{R+1}}(\mathbf{x}_t) = b_{i_{R+1}}(\mathbf{x}_t) \end{cases} \\ \text{with } i_1, \ldots, i_{R+1} \in \{0, 1, \ldots, N+1\} \tag{3.19}$$

The N state output pdfs of the pseudo HMM $\hat{\Phi}_{\text{R-K,lc}}$ are identical to the N state output pdfs of the R^{th}-order HMM $\Phi_{\text{R,lc}}$. The topology and state output pdfs of the $(R-K)$-order, left-context pseudo HMM are identical to that of the $(R-K)$-order, left-context HMM derived for the FBS-Viterbi-beam decoder. The only difference is in the values assigned to the $(R-K)$-order state transition probabilities.

The heuristic function is formally defined to be the best-path backward probability calculated using the derived pseudo-HMM $\hat{\Phi}_{\text{R-K,lc}}$ i.e.

$$h\left[n_t(i_1, i_2, \ldots, i_R)\right] \triangleq \hat{\overrightarrow{\epsilon}}_t(i_{K+2}, \ldots, i_R), \text{ with } i_1, i_2, \ldots, i_R \in \{0, 1, \ldots, N+1\} \tag{3.20}$$

The proof of the admissibility of the this heuristic function, defined to be the best-path backward probability $\hat{\overrightarrow{\epsilon}}_t(i_2, \ldots, i_R)$ calculated using the lower-order pseudo HMM $\hat{\Phi}_{\text{R-1,lc}}$, is presented in Appendix B.1.

3.2.6 Pruning of the A* search space

In a manner similar to the Viterbi-beam decoder, the A* decoder can also use beam-pruning to limit the number of evaluated state sequences. There are two available options of implementing A* search pruning. The first option is to base the pruning strategy

on partial observations, which effectively means that the A* search prunes using the value of the cost function $g(n)$. The second option is to base the pruning strategy on complete observations, which effectively means that the A* search prunes on the value of the evaluation function $f(n)$. We are of the opinion that the second option will result in less computationally expensive decoding, since this is been the case when the Forward-Backward search paradigm has been applied to speech recognition [43].

The pruning is implemented as a logarithmic beam-width B. All nodes at a certain time $t-1$, whose evaluation function does not fall within a certain threshold from the node at time $t-1$ with the best evaluation function (seen thus far by the A* decoder), are omitted from the rest of the search. The nodes are created by the A* search decoder, but omitted from the search by simply not inserting them onto the OPEN list. If it is found, at some point later in the search, that the evaluation function of a node is adjusted so that it does now fall within the pruning threshold, the fact that the node was created by the search means that it can be efficiently inserted onto the OPEN list. The threshold at time index t is calculated as $e^{-B} \max_k f\left[n_t(k)\right], B \geq 0$

3.2.6.1 Pruned decoding of high-order HMMs

In the case of time-asynchronous, beam-pruning search algorithms of R^{th}-order HMMs, the subset of promising candidates is determined by calculating the joint probability of sequences of states, ending at the previous time instance, given the complete observation sequence $\mathbf{X}_1^T$. Less probable sequences of states, ending at the previous time instance, are omitted from the rest of the search at time t. Thus, the A* decoder needs to calculate the state sequence probability at time $t-1$ as

$$\begin{aligned} \overrightarrow{\gamma}_{t-1}(i_1, i_2, \ldots, i_R) &= P(s_{t-R} = i_1, \ldots, s_{t-1} = i_R | \mathbf{X}_1^T, \Phi_{\text{R,lc}}) \\ &= \frac{\overrightarrow{\alpha}_{t-1}(i_1, i_2, \ldots, i_R)\overrightarrow{\beta}_{t-1}(i_1, i_2, \ldots, i_R)}{P(\mathbf{X}_1^T | \Phi_{\text{R,lc}})} \end{aligned} \tag{3.21}$$

In order to keep the most probable sequence of states at time $t-1$ the decoder does not need to directly calculate $\overrightarrow{\gamma}(i_1, i_2, \ldots, i_R)$ but can instead use $\overrightarrow{\alpha}_{t-1}(i_1, \ldots, i_R)$ and $\overrightarrow{\beta}_{t-1}(i_1, \ldots, i_R)$, since we are only interested in the ranking of the state sequences ending at time $t-1$ and $P(s_{t-R} = i_1, \ldots, s_{t-1} = i_R | \mathbf{X}_1^T, \Phi_{\text{lc}}) \sim \overrightarrow{\alpha}_{t-1}(i_1, \ldots, i_R)\overrightarrow{\beta}_{t-1}(i_1, \ldots, i_R)$.

When using a forward A* decoder to decode the HMM, the decoder does not compute the forward probability $\overrightarrow{\alpha}_{t-1}(i_1, \ldots, i_R)$, therefore the *best-path* forward probability $\overrightarrow{\delta}_{t-1}(i_1, \ldots, i_R) = g\left[n_{t-1}(i_1, \ldots, i_R)\right]$ is used instead.

The A* decoder also does not compute the backward probability or even the best-path backward probability. However, the heuristic function is an over-estimate of the best-path backward probability. Thus the A* decoder bases its pruning of a node $n_{t-1}(i_1, \ldots, i_R)$ on the estimated probability of the node at time $t-1$ given the complete observation

sequence $\mathbf{X}_1^T$ i.e.

$$\begin{aligned}\overrightarrow{\gamma}_{t-1}(i_1,\ldots,i_R) &\sim \overrightarrow{\alpha}_{t-1}(i_1,\ldots,i_R)\overrightarrow{\beta}_{t-1}(i_1,\ldots,i_R)\\ &\approx \overrightarrow{\delta}_{t-1}(i_1,\ldots,i_R)\overrightarrow{\epsilon}_{t-1}(i_1,\ldots,i_R)\\ &\leq g\left[n_{t-1}(i_1,\ldots,i_R)\right]h\left[n_{t-1}(i_1,\ldots,i_R)\right]\\ &= f\left[n_{t-1}(i_1,\ldots,i_R)\right] \end{aligned} \tag{3.22}$$

3.3 Backward Viterbi-beam decoding of left-context HMMs

Guiding the proposed decoders with information obtained from low-order HMMs, requires the calculation of best-path backward probabilities. In order to efficiently calculate these backward probabilities we need a decoder similar to the forward Viterbi-beam decoder, except that it calculates the probabilities in the backward direction. We call such a decoder a *backward Viterbi*-beam decoder and will now discuss it in more detail.

3.3.1 Backward decoding of first-order HMMs

In a manner similar to the first-order, best-path forward probability $\overrightarrow{\delta}_t(i)$ we define a new first-order, best-path backward probability[2] to be

$$\overrightarrow{\epsilon}'_t(i) = \max_{\mathbf{S}_{t+1}^T}\left[P(\mathbf{X}_t^T,\mathbf{S}_{t+1}^T|s_t=i,\Phi_{\text{lc}})\right] \tag{3.23}$$

i.e. the probability of the most likely state sequence $\mathbf{S}_t^T$, which has generated the partial observation sequence $\mathbf{X}_t^T$, given that the single state sequence starts at time t in state $s_t = i$ and the HMM Φ_{lc}. We define $\overrightarrow{\Psi}^{\epsilon}_t(i)$ to be the "forward" pointer of state i at time t and it denotes the state most likely to follow state i at time $t+1$. The following induction procedure is used to calculate $\overrightarrow{\epsilon}'_t(i)$:

1. Initialisation: $(t = T)$

$$\left.\begin{aligned}\overrightarrow{\epsilon}'_T(i) &= b_i(\mathbf{x_T})\overrightarrow{a}_{iN+1}\\ \overrightarrow{\Psi}^{\epsilon}_T(i) &= N+1\end{aligned}\right\} i = 1,\ldots,N \tag{3.24}$$

[2]We define the best-path backward probability $\overrightarrow{\epsilon}'_t(j)$ to include the observation at time t. The backward probability $\beta_t(j) = P(\mathbf{X}_{t+1}^T|s_t=j,\Phi_{\text{lc}})$ is conventionally defined not to include the observation at time t. The motivation for this will become clear during the discussion on the backward pruning strategy in Section 3.3.3.

2. Induction: $(t = T-1, T-2, \ldots, 2, 1)$

$$\left.\begin{array}{l} \overrightarrow{\epsilon}'_t(i) = b_i(\mathbf{x}_t) \max\limits_j \left[\overrightarrow{a}_{ij}\, \overrightarrow{\epsilon}'_{t+1}(j)\right] \\ \overrightarrow{\Psi}^{\epsilon}_t(i) = \arg\max\limits_j \left[\overrightarrow{a}_{ij}\, \overrightarrow{\epsilon}'_{t+1}(j)\right] \end{array}\right\} i = 1, \ldots, N \tag{3.25}$$

3. Termination:

$$\begin{array}{l} \overrightarrow{\epsilon}'_1 = \max\limits_i \left[\overrightarrow{a}_{0i}\, \overrightarrow{\epsilon}'_1(i)\right] \\ s^*_1 = \arg\max\limits_i \left[\overrightarrow{a}_{0i}\, \overrightarrow{\epsilon}'_1(i)\right] \end{array} \tag{3.26}$$

4. Backtracking:

$$\begin{array}{l} s^*_t = \overrightarrow{\Psi}^{\epsilon}_{t-1}(s^*_{t-1}),\ t = 2, 3, \ldots, T-1, T \\ \mathbf{S}^* = (s^*_1, s^*_2, \ldots, s^*_T) \end{array} \tag{3.27}$$

3.3.2 Generalisation to high-order HMMs

We redefine the R^{th}-order best-path backward probability as

$$\overrightarrow{\epsilon}'_t(i_1, i_2, \ldots, i_R) = \max_{\mathbf{S}^T_{t+1}} \left[P\left(\mathbf{X}^T_t, \mathbf{S}^T_{t+1} | s_{t-R+1} = i_1, \ldots, s_t = i_R, \Phi_{\text{lc}}\right)\right] \tag{3.28}$$

i.e. the probability of the most likely state sequence $\mathbf{S}^T_{t-R+1}$, which has generated the partial observation sequence $\mathbf{X}^T_t$, given that the single state sequence starts at time $t-R+1$ with state sequence $\mathbf{S}^t_{t-R+1} = \{s_{t-R+1} = i_1, s_{t-R+2} = i_2, \ldots, s_t = i_R\}$ and the HMM Φ_{lc} (if we let $R = 2$ this definition of the best-path probability is similar to the definition of the second-order, backward function $\beta_t(i_1, i_2)$ found in [24], except for the inclusion of the observation $\mathbf{x}_t$ at time t). $\overrightarrow{\Psi}^{\epsilon}_t(i_1, i_2, \ldots, i_R)$ is defined to be the "forward" pointer at time t of the state sequence $\{s_{t-R+1} = i_1, s_{t-R+2} = i_2, \ldots, s_t = i_R\}$ and denotes the most likely state to follow the state sequence at time $t+1$. The following induction procedure is used to calculate the backward best-path probability:

1. Initialisation: $(t = T; i_1, \ldots, i_R = 1, 2, \cdots, N)$

$$\begin{array}{l} \overrightarrow{\epsilon}'_T(i_1, i_2, \ldots, i_R) = b_{i_R}(\mathbf{x}_T)\, \overrightarrow{a}_{i_1 i_2 \cdots i_R (N+1)} \\ \overrightarrow{\Psi}^{\epsilon}_T(i_1, i_2, \ldots, i_R) = N + 1 \end{array} \tag{3.29}$$

2. Induction: $(t = T-1, T-2, \ldots, R;\ i_1, \ldots, i_R = 1, 2, \cdots, N)$

$$\begin{array}{l} \overrightarrow{\epsilon}'_t(i_1, i_2, \ldots, i_R) = b_{i_R}(\mathbf{x}_t) \max\limits_{i_{R+1}} \left[\ \overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}}\, \overrightarrow{\epsilon}'_{t+1}(i_2, i_3, \ldots, i_{R+1})\ \right] \\ \overrightarrow{\Psi}^{\epsilon}_t(i_1, i_2, \ldots, i_R) = \arg\max\limits_{i_{R+1}} \left[\ \overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}}\, \overrightarrow{\epsilon}'_{t+1}(i_2, i_3, \ldots, i_{R+1})\ \right] \end{array} \tag{3.30}$$

3. Induction: $(t = R-1, \ldots, 2, 1; i_1, \ldots, i_R = 1, 2, \cdots, N)$

$$\begin{aligned}\overrightarrow{\epsilon}'_t(i_1, i_2, \ldots, i_t) &= b_{i_t}(\mathbf{x}_t) \max_{i_{t+1}} \left[\overrightarrow{a}_{0 i_1 i_2 \cdots i_{t+1}} \overrightarrow{\epsilon}'_{t+1}(i_1, i_2, \ldots, i_{t+1})\right] \\ \overrightarrow{\Psi}^{\epsilon}_t(i_1, i_2, \ldots, i_t) &= \arg\max_{i_{t+1}} \left[\overrightarrow{a}_{0 i_1 i_2 \cdots i_{t+1}} \overrightarrow{\epsilon}'_{t+1}(i_1, i_2, \ldots, i_{t+1})\right]\end{aligned} \tag{3.31}$$

4. Termination:

$$\begin{aligned}\overrightarrow{\epsilon}'_1 &= \max_{i_1} \left[\overrightarrow{a}_{0 i_1} \overrightarrow{\epsilon}'_1(i_1)\right] \\ s^*_1 &= \arg\max_{i_1} \left[\overrightarrow{a}_{0 i_1} \overrightarrow{\epsilon}'_1(i_1)\right]\end{aligned} \tag{3.32}$$

5. Backtracking:

$$\begin{aligned}s^*_t &= \overrightarrow{\Psi}^{\epsilon}_{t-1}(s^*_{t-R}, s^*_{t-R+1}, \cdots, s^*_{t-1}),\ t = R+1, R+2, \ldots, T-1, T \\ \mathbf{S}^* &= (s^*_1, s^*_2, \ldots, s^*_T)\end{aligned} \tag{3.33}$$

3.3.3 Pruning the decoding search space

The backward Viterbi-beam decoder also uses pruning to limit the number of evaluated state sequences. Instead of retaining all candidates at every time frame, a threshold is used to keep only a subset of promising candidates.

3.3.3.1 Pruned decoding of first-order HMMs

For pruned searches in the backward direction, the decoder needs to calculate the state sequence probabilities at time $t+1$, given the partial observation sequence $\mathbf{X}^T_{t+1}$:

$$\begin{aligned}P(s_{t+1} = j|\mathbf{X}^T_{t+1}, \Phi_{\text{lc}}) &= \frac{P(\mathbf{X}^T_{t+1}, s_{t+1} = j|\Phi_{\text{lc}})}{P(\mathbf{X}^T_{t+1}|\Phi_{\text{lc}})} \\ &= \frac{b_j(\mathbf{x}_{t+1})\overrightarrow{\beta}_{t+1}(j)P(s_{t+1} = j|\Phi_{\text{lc}})}{P(\mathbf{X}^T_{t+1}|\Phi_{\text{lc}})}\end{aligned} \tag{3.34}$$

Therefore, in order to keep the most probable states at time $t+1$ we do not need to directly calculate $P(s_{t+1} = j|\mathbf{X}^T_{t+1}, \Phi_{\text{lc}})$ but can instead use $b_j(\mathbf{x}_{t+1})\overrightarrow{\beta}_{t+1}(j)P(s_{t+1} = j|\Phi_{\text{lc}})$ as we are only interested in the ranking of the states and $P(s_{t+1} = j|\mathbf{X}^T_{t+1}, \Phi_{\text{lc}}) \sim b_j(\mathbf{x}_{t+1})\overrightarrow{\beta}_{t+1}(j)P(s_{t+1} = j|\Phi_{\text{lc}})$.

We can already see that the equation used for state pruning during backward Viterbi-beam decoding differs from the equation used for pruning during forward decoding. Since the conventionally defined backward probability $\overrightarrow{\beta}_{t+1}(j)$ does not include the contribution of the observation at time $t+1$, the pruning strategy needs to multiply $\overrightarrow{\beta}_{t+1}(j)$ with $b_j(\mathbf{x}_{t+1})$. This problem can be solved by defining a new backward probability that *does*

include the contribution of the observation $\mathbf{x}_{t+1}$. The new backward probability is defined as

$$\begin{aligned}\overrightarrow{\beta}'_{t+1}(j) &\triangleq P(\mathbf{X}_{t+1}^{T}|s_t = j, \Phi_{\text{lc}}) \\ &= P(\mathbf{x}_{t+1}|s_t = j, \Phi_{\text{lc}})P(\mathbf{X}_{t+2}^{T}|s_t = j, \Phi_{\text{lc}}) \\ &= b_j(\mathbf{x}_{t+1})\overrightarrow{\beta}_{t+1}(j) \end{aligned} \tag{3.35}$$

The second problem with beam-pruned backward decoding is that, according to Eq. 3.34, the backward search pruning needs to multiply the best-path backward probability $\overrightarrow{\beta}'_j(t+1)$ with the state *prior* probability $P(s_{t+1} = j|\Phi_{\text{lc}})$. This extra *prior* multiplication already decreases the computational efficiency of backward decoding. The third problem is that conventional HMMs typically do not store this state *prior* probability information; only the conditional state transition probabilities are stored. We will therefore investigate the computational efficiency of backward Viterbi-beam pruning with and without these extra state *prior* probabilities. We will denote the state pruning with the state *priors* as *maximum a posteriori* (MAP) beam pruning and the state pruning without the state *priors* as *maximum likelihood* (ML) beam pruning. MAP-beam pruning is equivalent to ML-beam pruning if the state *priors* are uniformly distributed. Since the MAP-beam pruning is similar to a Bayesian classifier, we expect the ML-beam pruning to be more computationally expensive than the MAP-beam pruning.

When using backward Viterbi-beam decoders to decode the HMM, the decoder does not compute the newly defined backward probability $\overrightarrow{\beta}'_{t+1}(j)$, therefore the *best-path* backward probability $\overrightarrow{\epsilon}'_{t+1}(j)$ (which is defined to include the observation $\mathbf{x}_{t+1}$) is used instead, since we assume that $\overrightarrow{\beta}'_{t+1}(j) = b_j(\mathbf{x}_{t+1})\overrightarrow{\beta}_{t+1}(j) \approx \overrightarrow{\epsilon}'_{t+1}(j)$. For backward decoding the MAP pruning threshold is calculated as

$$e^{-B} \max_k \left[\overrightarrow{\epsilon}'_{t+1}(k)P(s_{t+1} = k|\Phi_{\text{lc}})\right], \; B \geq 0 \tag{3.36}$$

When we use the first-order, best-path backward probability $\overrightarrow{\epsilon}'_{t+1}(j)$ in the proposed FBS-based decoders, we also need to define a new first-order, best-path forward probability $\overrightarrow{\delta}'_{t+1}(j)$ that *excludes* the observation at time $t+1$, otherwise when we combine the forward and backward best-path probabilities the likelihood of the observation $\mathbf{x}_{t+1}$ will be included twice. Therefore $\overrightarrow{\delta}'_t(j)$ is defined as:

$$\overrightarrow{\delta}'_t(j) \triangleq \max_{\mathbf{S}_1^{t-1}} \left[P(\mathbf{X}_1^{t-1}, \mathbf{S}_1^{t-1}, s_t = j|\Phi_{\text{lc}})\right] \tag{3.37}$$

3.3.3.2 Pruned decoding of high-order HMMs

In the case of time-synchronous, beam-pruning search algorithms of R^{th}-order HMMs, the subset of promising candidates is determined by calculating the joint probability of sequences of states, ending at the following time instance, given the observations seen thus

far. Less probable sequences of states, ending at the following time instance, are omitted from the rest of the search at time t. For pruned searches in the backward direction, the decoder needs to calculate

$$\begin{aligned} & P(s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1}|\mathbf{X}_{t+1}^T,\Phi_{\text{lc}}) \\ = & \frac{P(s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1},\mathbf{X}_{t+1}^T|\Phi_{\text{lc}})}{P(\mathbf{X}_{t+1}^T|\Phi_{\text{lc}})} \\ = & \frac{b_j(\mathbf{x}_{t+1})\overrightarrow{\beta}_{t+1}(i_2,\ldots,i_{R+1})P(s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1}|\Phi_{\text{lc}})}{P(\mathbf{X}_{t+1}^T|\Phi_{\text{lc}})} \end{aligned} \tag{3.38}$$

In order to keep the most probable states at time $t+1$ we therefore do not need to directly calculate $P(s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1}|\mathbf{X}_{t+1}^T,\Phi_{\text{lc}})$ but can instead use $b_j(\mathbf{x}_{t+1})\overrightarrow{\beta}_{t+1}(i_2,\ldots,i_{R+1})P(s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1}|\Phi_{\text{lc}})$ as we are only interested in the ranking of the states and $P(s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1}|\mathbf{X}_{t+1}^T,\Phi_{\text{lc}}) \sim b_j(\mathbf{x}_{t+1})\overrightarrow{\beta}_{t+1}(i_2,\ldots,i_{R+1})P(s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1}|\Phi_{\text{lc}})$

The same problem of the extra multiplication to include the observation $\mathbf{x}_{t+1}$, occurs with the pruned decoding of high-order HMMs. We therefore define a new R^{th}-order, backward probability as

$$\begin{aligned} & \overrightarrow{\beta}'_{t+1}(i_2,\ldots,i_{R+1}) \\ \triangleq & P(\mathbf{X}_{t+1}^T|s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1},\Phi_{\text{lc}}) \\ = & P(\mathbf{x}_{t+1}|s_t=i_2,\Phi_{\text{lc}})P(\mathbf{X}_{t+2}^T|s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1},\Phi_{\text{lc}}) \\ = & b_j(\mathbf{x}_{t+1})\overrightarrow{\beta}_{t+1}(i_2,\ldots,i_{R+1}) \end{aligned} \tag{3.39}$$

Similar to the backward of first-order HMMs, the newly defined backward probability needs to be multiplied with the following *joint* state *prior* probability $P(s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1}|\Phi_{\text{lc}})$, according to Eq. 3.38.

When using backward Viterbi-beam decoders to decode the HMM, the decoder does not compute the backward probability $\overrightarrow{\beta}_{t+1}(i_2,\ldots,i_{R+1})$, therefore the *best-path* backward probability $\overrightarrow{\epsilon}_{t+1}(i_2,\ldots,i_{R+1})$ is used instead since we assume that

$$\overrightarrow{\beta}'_{t+1}(i_2,\ldots,i_{R+1}) = b_j(\mathbf{x}_{t+1})\overrightarrow{\beta}_{t+1}(i_2,\ldots,i_{R+1}) \approx \overrightarrow{\epsilon}'_{t+1}(i_2,\ldots,i_{R+1})$$

When we use the $(R-K)$-order, best-path backward probability $\overrightarrow{\epsilon}'_{t+1}(i_{K+2},\ldots,i_{R+1})$ in the proposed decoders, we also need to define a new R^{th}-order, best-path forward probability $\overrightarrow{\delta}'_{t+1}(i_2,\ldots,i_{R+1})$ that *excludes* the observation at time $t+1$, otherwise when we combine the forward and backward best-path probabilities the likelihood of the observation $\mathbf{x}_{t+1}$ will be included twice. Therefore $\overrightarrow{\delta}'_{t}(i_2,\ldots,i_{R+1})$ is defined as:

$$\overrightarrow{\delta}'_{t}(i_2,\ldots,i_{R+1}) \triangleq \max_{\mathbf{S}_1^{t-R}} \left[P(\mathbf{X}_1^{t-1},\mathbf{S}_1^{t-R},s_{t-R+1}=i_2,\ldots,s_t=i_{R+1}|\Phi_{\text{lc}})\right] \tag{3.40}$$

3.3.4 Evaluating backward decoding of HMMs

We have made the assumption that the backward decoding of high-order HMMs is computationally equivalent to the forward decoding of the same high-order HMMs. We now investigate the computational expense of backward decoding high-order HMMs to test the validity of our assumption.

We repeat the previous decoding experiment performed in Section 2.2.4, using the same high-order, left-context HMMs. We use the same experimental setup for decoding left-context HMMs as was used in Section 2.2.4, except that we only evaluate the decoding performance using the English development set. This represents a scenario where the training and evaluation conditions are mismatched. In this experiment we investigate the decoding performance of the backward Viterbi-beam decoders when decoding high-order, left-context HMMs.

3.3.4.1 Measuring decoder performance

As before, we use the search cost C_s (the number of transition probabilities evaluated during decoding) as an implementation independent measure of the performance of the backward Viterbi-beam decoder. Over the 1410 segments decoded we compute the average number of evaluated transition probabilities. We compute the normalised search cost $C_{s,n}$ by normalising the number of evaluated transition probabilities with the maximum number of transition probabilities that the standard Viterbi decoder would evaluate.

We measure the search cost for backward Viterbi-beam decoding of high-order HMMs and compare it to the search cost for forward Viterbi-beam decoding, as was measured in the experiment in Section 2.2.4.

3.3.4.2 Results

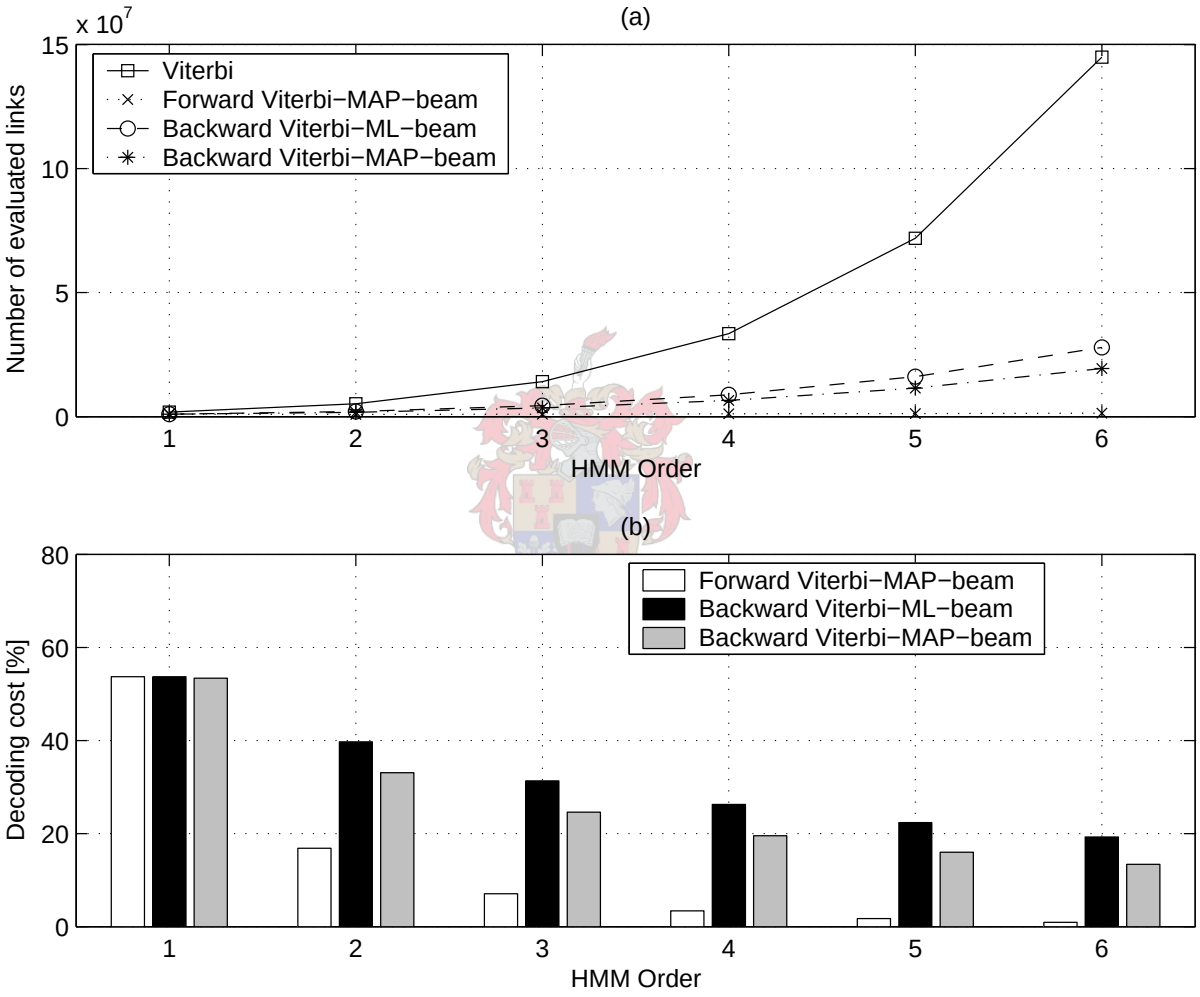
As in the previous experiment, the following two beam-pruning configurations are investigated as the order of the HMMs increases:

- The beam-width is set to a constant $B = 20.0$.
- The minimum integer beam-width is determined for which the decoder correctly decodes all 1410 segments.

Constant beam-width

Fig. 3.3(a) shows the search cost C_s (the number of transition probabilities evaluated) when decoding a left-context HMM. The four cases shown are Viterbi decoding; *forward* Viterbi-beam decoding; *backward* Viterbi-ML-beam decoding and *backward* Viterbi-MAP-beam decoding. From the results we note that backward Viterbi-beam decoding, with

Figure 3.3: *(a) The number of transitions evaluated by the Viterbi-ML-beam and Viterbi-MAP-beam decoders (the search cost C_s), during the backward decoding of high-order, left-context HMMs, when the decoders are using a constant beam-width of $B = 20.0$. (b) The normalised search cost $C_{s,n}$ of the Viterbi-ML-beam and Viterbi-MAP-beam decoders, during the backward decoding of high-order, left-context HMMs, when the decoders are using a constant beam-width of $B = 20.0$.*



either MAP-beam or ML-beam state pruning, is computationally significantly more expensive than forward Viterbi-beam decoding. It is interesting to note that the use of the MAP-beam pruning does reduce the search cost, but it still requires significantly more transition evaluations than the forward Viterbi-beam decoder. Table 3.1 tabulates the accuracy of decoding the different order HMMs, when using a constant beam.

Table 3.1: *The computational expense of the Viterbi-ML-beam and Viterbi-MAP-beam decoders, during the backward decoding of high-order, left-context HMMs, when the decoders are using a constant beam-width of $B = 20.0$.*

Order	Search cost C_s		Norm. cost $C_{s,n}$ [%]		Accuracy [%]	
	ML-beam	MAP-beam	ML-beam	MAP-beam	ML-beam	MAP-beam
1	993,708	987,825	53.71	53.40	98.37	98.87
2	2,051,637	1,707,791	39.71	33.06	98.51	97.87
3	4,421,159	3,470,106	31.31	24.58	99.22	98.94
4	8,783,532	6,528,919	26.27	19.53	98.87	97.80
5	16,069,948	11,499,049	22.36	16.00	97.45	94.68
6	27,887,726	19,366,932	19.25	13.37	94.96	80.78

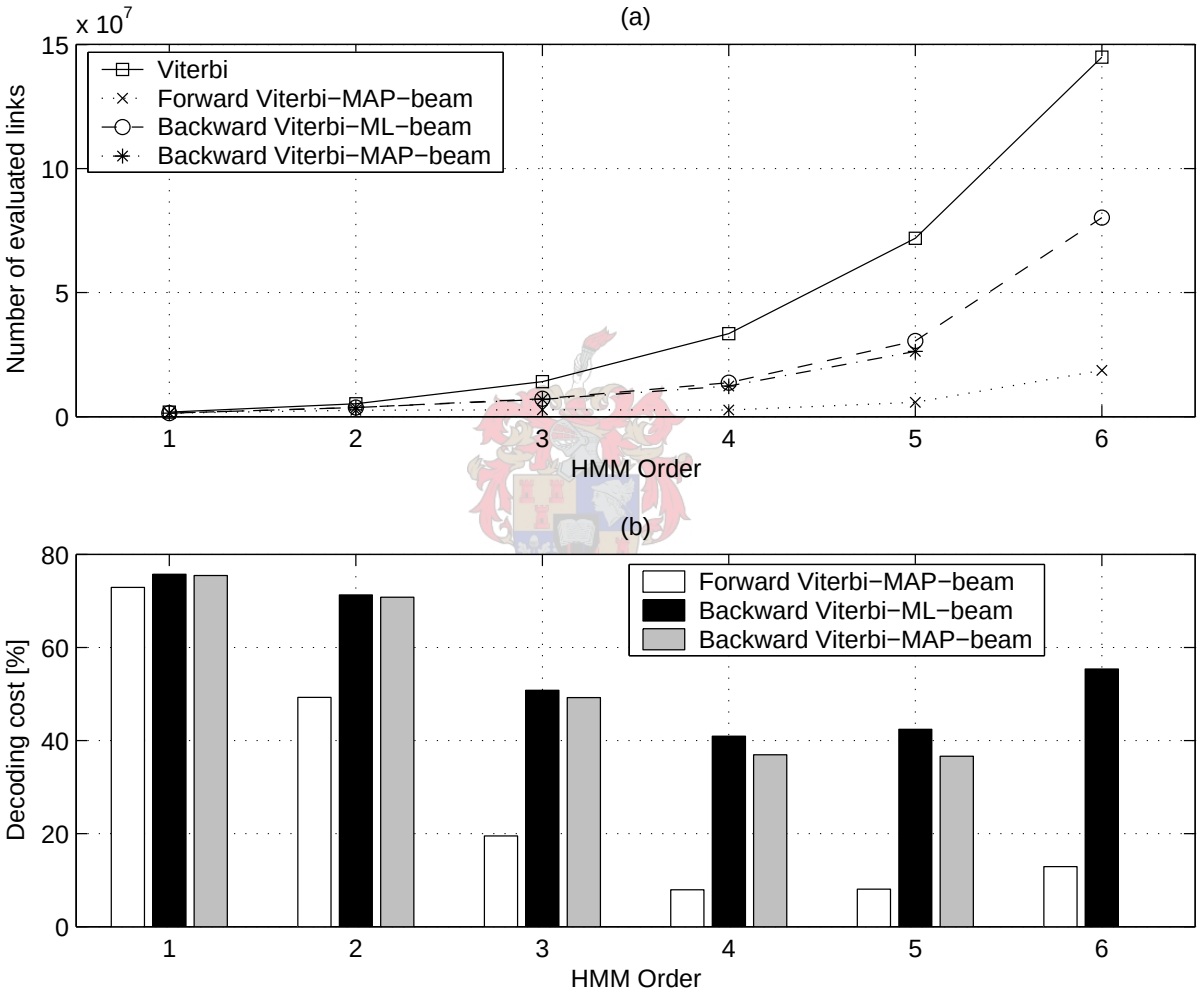
Minimum computational expense of decoder when correctly decoding all segments

Fig. 3.4(b) shows the normalised search cost of forward and backward *Viterbi-beam* decoders. The difference in normalised search cost is initially about 20% and slowly decreases with an increase in the order of the HMM. Table 3.2 tabulates the minimum integer beam-width required to correctly decoded all segments. Once again, backward Viterbi-beam decoding still requires significantly more transition evaluations than forward Viterbi-beam decoding, in order to obtain the same optimal state sequence. Thus our assumption that forward and backward Viterbi-beam decoding is computationally equivalent is not valid.

Table 3.2: *The minimum computational expense of the Viterbi-ML-beam and Viterbi-MAP-beam decoders, during the backward decoding of high-order, left-context HMMs, with beams set wide enough to decode all segments correctly.*

Order	Search cost C_s		Norm. cost $C_{s,n}$ [%]		Beam-width B	
	ML-beam	MAP-beam	ML-beam	MAP-beam	ML-beam	MAP-beam
1	1,400,373	1,395,639	75.67	75.44	31.0	31.0
2	3,683,111	3,655,706	71.32	70.79	37.0	40.0
3	7,168,392	6,952,352	50.80	49.27	30.0	33.0
4	13,683,229	12,348,516	40.98	36.99	28.0	30.0
5	30,471,123	26,313,034	42.48	36.69	32.0	33.0
6	80,206,383	-	55.36	-	45.0	-

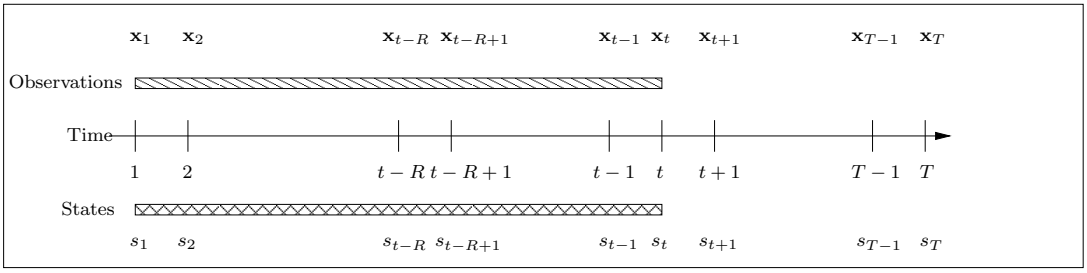
Figure 3.4: *(a) The minimum search cost of the Viterbi-ML-beam and Viterbi-MAP-beam decoders, during the backward decoding of high-order, left-context HMMs, with beams set wide enough to decode all segments correctly. (b) The normalised search cost of the Viterbi-ML-beam and Viterbi-MAP-beam decoders, during the backward decoding of high-order, left-context HMMs, with beams set wide enough to decode all segments correctly.*



3.3.4.3 Discussion

In Section 3.3.3 we discussed some of the differences between forward and backward Viterbi-beam decoding. Specifically, we mentioned that when decoding first-order HMMs using forward Viterbi-beam searches, we can directly use the already calculated best-path forward probability $\overrightarrow{\delta}_t(i)$. However when decoding first-order HMMs using backward Viterbi-beam searches, the best-path backward probability $\overrightarrow{\epsilon}_t(j)$ (which we have defined to include the contribution of the observation at t) must be multiplied with the state *prior* probability $P(s_t = j|\Phi_{\text{lc}})$. For first-order HMMs we saw that the inclusion (or

Figure 3.5: *An illustration of the 'time-synchronicity' of the observation sequence and state sequence during forward Viterbi decoding.*



exclusion) of this state probability $P(s_t = j|\Phi_{\text{lc}})$ does not have a significant impact on the computational efficiency of decoding the fully-connected, left-context HMM. However, when the order of the HMM is increased, the inclusion of the joint state *prior* probability $P(s_{t-R+1} = i_1, \ldots, s_t = i_R|\Phi_{\text{lc}})$ *does not* cause the backward Viterbi-beam decoding to be as computationally efficient as the forward Viterbi-beam decoding. Therefore, using the best-path probability at time $t+1$

$$\overrightarrow{\epsilon}_{t+1}(i_2, \cdots, i_{R+1}) = \max_{\mathbf{S}_{t+2}^T} \left[P(s_{t-R+2} = i_2, \ldots, s_{t+1} = i_{R+1}|\mathbf{X}_{t+1}^T, \Phi_{\text{lc}})\right] \tag{3.41}$$

for state pruning during backward Viterbi-beam decoding, is not as efficient as using the best-path probability at time $t-1$

$$\overrightarrow{\delta}_{t-1}(i_1, \cdots, i_R) = \max_{\mathbf{S}_{t-2}^T} \left[P(s_{t-R} = i_1, \ldots, s_{t-1} = i_R|\mathbf{X}_1^{t-1}, \Phi_{\text{lc}})\right] \tag{3.42}$$

for state pruning during forward Viterbi-beam decoding. Upon closer examination of Eq. 3.42, which is used for state pruning during forward decoding, we note that the state sequence under consideration and the observations are 'time-synchronised', as illustrated in Fig. 3.5. This is not the case when we examine Eq. 3.41, which is used for state pruning during backward decoding, as illustrated in Fig. 3.6. This 'time-synchronicity' in the forward direction and 'time-asynchronicity' in the backward direction is a direct result of the definition of the state transition probabilities and the states with which the observation pdfs are associated.

We suspect the fact that the observations and states are not synchronised results in the backward Viterbi-beam decoder being computationally less efficient than the forward Viterbi-beam decoder. This problem can be solved by adding the contributions of the "missing" observations when performing backward Viterbi-beam decoding; thereby calculating $\max_{\mathbf{S}_{t+2}^T} \left[P(s_{t-R+2} = i_2, \ldots, s_{t+1} = i_{R+1}|\mathbf{X}_{t-R+2}^T, \Phi_{\text{lc}})\right]$. However, this requires even more additional calculations. Furthermore, the number of "missing" observations is dependent on the order R of the HMM, therefore adapting the backward Viterbi-beam

decoder to include the observations will also make the algorithm dependent on the order of the HMM being decoded. This is not desirable as it nullifies the advantage of using first-order equivalent HMMs of high-order HMMs.

3.4 Summary

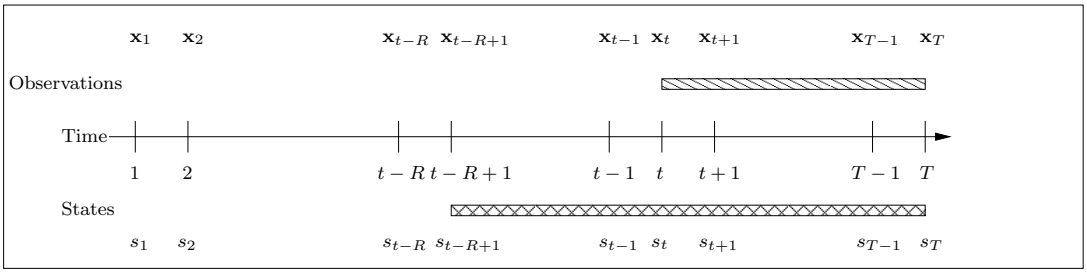
In this chapter we have proposed two new algorithms for decoding high-order HMMs. The motivation for both decoders is to use information obtained from decoding low-order HMMs and to exploit the fact the decoding of low-order HMMs is significantly less expensive than the decoding high-order HMMs.

The first decoder is a time-synchronous decoder which is nearly identical to the Viterbi-beam decoder. It incorporates information, obtained by decoding low-order HMMs, into the state pruning strategy. This extra information results in the pruning strategy being based on the complete observation sequence as opposed to being based on only the partial observation sequence. This should result in a more aggressive pruning of the search space, while still preserving the ability to decode the optimal state sequence.

The second decoder is a time-asynchronous decoder which is an application of the A* search algorithm to the task of decoding high-order HMMs. The novelty of this decoder is the use of low-order HMMs when calculating the heuristic function used to guide the A* search.

Both decoders require the calculation of a best-path backward probability, (which is similar to the best-path forward probability) on low-order HMMs. We have therefore adapted the forward Viterbi-beam decoder so that we can calculate the R^{th}-order, best-path backward probability, assuming that forward and backward decoding of high-order HMMs are computationally equivalent. As the proposed decoders are attempting to exploit the decoding of low-order HMMs, it is necessary that forward and backward decoding of HMMs are computationally equivalent.

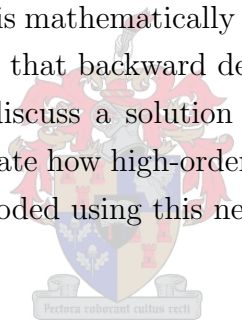
Figure 3.6: *An illustration of the 'time-asynchronicity' of the observation sequence and state sequence during backward Viterbi decoding.*



Lastly, we tested the validity of our assumption of computational equivalence between forward and backward Viterbi-beam decoding. This was done by measuring the computational expense of backward decoding high-order HMMs. We compared the new results to the computational expense of forward decoding the same high-order HMMs. To our surprise we found that the backward Viterbi-beam decoding of high-order HMMs is computationally significantly more expensive than forward Viterbi-beam decoding.

The question arises whether decoding is fundamentally more computationally expensive when time-reversing the observation sequence, than when the observation sequence is kept the same. We suspect that it is *not* fundamental to the time-reversal of the observation sequence, but that the extra computation is caused by the asynchronicity between the observations and states under consideration, when backward decoding. However, this is a direct result of the definition of the state transition probabilities. We therefore believe that the backward Viterbi-beam decoding is fundamentally more expensive than forward Viterbi-beam decoding, *if the same left-context HMM specification is used*. If a different HMM specification is used, which is mathematically equivalent to the original left-context HMM specification, we might find that backward decoding is *not* more expensive.

In the next chapter we will discuss a solution to this problem, by defining a new HMM specification, and demonstrate how high-order HMMs can be backward decoded as efficiently as they are forward decoded using this new HMM specification.



Chapter 4

Right-context, high-order HMMs

4.1 Motivation

In the previous chapter we demonstrated that decoding a high-order, left-context HMM with the forward Viterbi-beam decoder is computationally more efficient than decoding it with the backward Viterbi-beam decoder. The backward Viterbi-beam decoder is a critical component of our newly proposed decoders, as it calculates the best-path backward probability required to guide the decoding searches. If backward decoding is fundamentally more expensive than forward decoding, then any reduction in search cost that we might gain by using low-order HMMs will be negated by the computationally more expensive backward decoding.

The right-context HMM was developed as a solution to this problem of computationally more expensive backward decoding of high-order HMMs. The right-context HMM effectively solves the problem by 'time-synchronising' the observations and the state sequence under consideration, without having to resort to extra calculations in order to add "missing" observations or joint state probabilities. We will demonstrate in this chapter that the right-context HMM, which is mathematically equivalent to the left-context HMM, can be backward Viterbi-beam decoded as computationally efficiently as the left-context HMM can be forward Viterbi-beam decoded.

In the first part of this chapter we will present the right-context HMM. We will show how the Viterbi-beam decoder can be extended to include the decoding of high-order, right-context HMMs. We will demonstrate the equivalence of the left- and right-context HMM by measuring the forward- and backward Viterbi search cost.

In the second part of this chapter we will show how the right-context HMM can be used to calculate the heuristic functions required by both the low-order guided Viterbi-beam decoder and the A* search decoder.

4.2 Derivation and Definition

Given a R^{th}-order HMM, we want to compute the probability of the state sequence $\mathbf{S}_t^{t+R-1}$ given the observation sequence $\mathbf{X}_t^T$ and the HMM Φ (we deliberately do not make any assumptions regarding the model at this stage), i.e.

$$P(s_t = i_1, \ldots, s_{t+R-1} = i_R | \mathbf{X}_t^T, \Phi) = \frac{P(s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{X}_t^T | \Phi)}{P(\mathbf{X}_t^T | \Phi)} \tag{4.1}$$

The joint probability of the state sequence $\mathbf{S}_t^{t+R-1}$ and the observation sequence $\mathbf{X}_t^T$ can be calculated as:

$$\begin{aligned}
& P(s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{X}_t^T | \Phi) \\
= & \sum_{\mathbf{S}_{t+R}^T} P(s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T, \mathbf{X}_t^T | \Phi) \\
= & \sum_{\mathbf{S}_{t+R}^T} \left[\begin{array}{l} P(\mathbf{X}_t^T | s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T, \Phi) \times \\ P(s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T | \Phi) \end{array} \right]
\end{aligned}$$

By applying the output independence assumption, we note that $P(\mathbf{X}_t^T | s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T, \Phi)$ is simply the contributions of the state output pdfs, i.e.

$$P(\mathbf{X}_t^T | s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T, \Phi) = b_{i_1}(\mathbf{x}_t) \cdots b_{i_R}(\mathbf{x}_{t+R-1}) \prod_{\tau=t+R}^{T} b_{s_\tau}(\mathbf{x}_\tau) \tag{4.2}$$

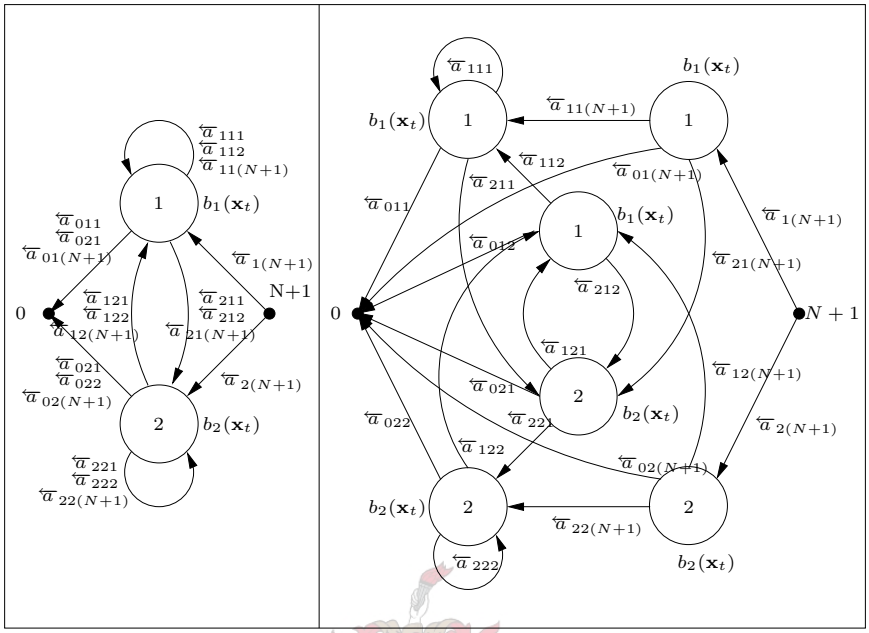
Furthermore, we note that by applying Bayes' Theorem, we can calculate the state sequence probability $P(s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T | \Phi)$ as:

$$\begin{aligned}
& P(s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T | \Phi) \\
= & P(s_t = i_1 | s_{t+1} = i_2, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T, \Phi) \times \\
& P(s_{t+1} = i_2, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T | \Phi) \\
= & P(s_t = i_1 | s_{t+1} = i_2, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T, \Phi) \times \\
& P(s_{t+1} = i_2 | s_{t+2} = i_3, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T | \Phi) \ldots P(s_{T-1} | s_T, \Phi) P(s_T | \Phi)
\end{aligned} \tag{4.3}$$

This calculation is very similar to the calculation of the forward probability, except that the state transition probabilities are conditioned on the *subsequent* states. If we make the further assumption that for a R^{th}-order, right-context HMM the state transitions are only conditionally dependent on the *subsequent* R states i.e.

$$\begin{aligned}
\overleftarrow{a}_{i_1 i_2 \ldots i_{R+1}} & \triangleq P(s_t = i_1 | s_{t+1} = i_2, \ldots, s_{t+R} = i_{R+1}) \text{ with} \\
\sum_{i_1=0}^{N+1} \overleftarrow{a}_{i_1 i_2 \ldots i_{R+1}} & = 1, \; i_2, \ldots, i_R \in \{1, \ldots, N+1\}
\end{aligned} \tag{4.4}$$

Figure 4.1: *(a) A two emitting-state, second-order, right-context HMM. (b) First-order equivalent of (a).*



then we can simplify the state sequence probability to be

$$P(s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T | \Phi) = \overleftarrow{a}_{i_1 i_2 \cdots i_R s_{t+R}} \cdots \overleftarrow{a}_{s_{T-1} s_T} \overleftarrow{a}_{s_T (N+1)} \tag{4.5}$$

If we define the R^{th}-order, backward probability of the right-context HMM to be

$$\overleftarrow{\beta}_t(i_1, i_2, \ldots, i_R) = \max_{\mathbf{S}_{t+R}^T} \left[P(\mathbf{X}_t^T, s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T | \Phi_{\text{rc}}) \right] \tag{4.6}$$

we can calculate $P(s_t = i_1, \ldots, s_{t+R-1} = i_R | \mathbf{X}_t^T, \Phi)$ as

$$\begin{aligned} P(s_t = i_1, \ldots, s_{t+R-1} = i_R | \mathbf{X}_t^T, \Phi) &= \frac{P(s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{X}_t^T | \Phi)}{P(\mathbf{X}_t^T | \Phi)} \\ &= \frac{\overleftarrow{\beta}_t(i_1, i_2, \ldots, i_R)}{P(\mathbf{X}_t^T | \Phi)} \end{aligned} \tag{4.7}$$

and therefore the most likely states at time t given the *subsequent* observations can be determined without having to resort to extra calculations.

The definition of the right-context HMM is thus identical to the definition of the left-context HMM except for the definition of the state transition probabilities. The right-context HMM is obtained by redefining the conditional state transition probability $a_{i_1 i_2 \ldots i_{R+1}}$.

A right-context, N emitting-state HMM Φ_{rc} is defined by the parameter set

$$\Phi_{\text{rc}} = \left\{ \overleftarrow{a}_{i_1 i_2 \ldots i_{R+1}}, b_{i_1}(\mathbf{x}), i_1, i_2, \ldots, i_{R+1} \in \{0, \cdots, N+1\} \right\} \tag{4.8}$$

Fig. 4.1(a) illustrates a two-state, right-context HMM of the second order. We see that the right-context HMM is very similar to the left-context HMM shown in Fig. 2.2(a) and once again two states are coupled by multiple transition probabilities. Fig. 4.1(b) illustrates the first-order equivalent of the right-context HMM which bears a striking resemblance to the first-order equivalent of the left-context HMM (shown in Fig. 2.2(b)).

4.3 Decoding of Right-context HMMs

In this section we will show how the forward- and backward Viterbi-beam decoders can be adapted to the decoding of high-order, right-context HMMs. The forward Viterbi-beam decoding of right-context HMMs is not strictly necessary, but we include the derivation for completeness. We will see that forward decoding of right-context HMMs suffer from the same problems as backward decoding of left-context HMMs.

4.3.1 Forward Viterbi-beam decoding of right-context HMMs

4.3.1.1 Forward Viterbi decoding of first-order, right-context HMMs

The right-context, best-path forward probability $\overleftarrow{\delta}_t(i)$ is defined as

$$\overleftarrow{\delta}_t(i) = \max_{\mathbf{S}_1^{t-1}} \left[P(\mathbf{X}_1^t, \mathbf{S}_1^{t-1} | s_t = i, \Phi_{\text{rc}}) \right] \tag{4.9}$$

$\overleftarrow{\delta}_t(i)$ is the probability of the most likely state sequence $\mathbf{S}_1^{t-1}$, which has generated the partial observation sequence $\mathbf{X}_1^t$, given that the state sequence ends at time t in state i and the HMM Φ_{rc}. $\overleftarrow{\Psi}_t^{\delta}(i)$ is the back pointer of state $s_t = i$ and denotes the state most likely to precede the state i at time $t-1$. The following induction procedure is used to calculate $\overleftarrow{\delta}_t(i)$:

1. Initialisation: $(t = 1; i = 1, 2, \ldots, N)$

$$\begin{aligned} \overleftarrow{\delta}_1(i) &= \overleftarrow{a}_{0i} b_i(\mathbf{x}_1) \\ \overleftarrow{\Psi}_1^{\delta}(i) &= 0 \end{aligned} \tag{4.10}$$

2. Induction: $(t = 2, 3, \ldots, T;\ j = 1, 2, \ldots, N)$

$$\begin{aligned} \overleftarrow{\delta}_t(j) &= \left[\max_i \overleftarrow{\delta}_{t-1}(i) \overleftarrow{a}_{ij} \right] b_j(\mathbf{x}_t) \\ \overleftarrow{\Psi}_t^{\delta}(i) &= \arg\max_i \left[\overleftarrow{\delta}_{t-1}(i) \overleftarrow{a}_{ij} \right] \end{aligned} \tag{4.11}$$

3. Termination:

$$\begin{aligned} \overleftarrow{\delta}_T &= \max_i \left[\overleftarrow{\delta}_T(i) \overleftarrow{a}_{i(N+1)} \right] \\ s_T^* &= \arg\max_i \left[\overleftarrow{\delta}_T(i) \overleftarrow{a}_{i(N+1)} \right] \end{aligned} \tag{4.12}$$

4. Backtracking:

$$\begin{aligned} s_t^* &= \overleftarrow{\Psi}^{\delta}_{t+1}(s^*_{t+1}),\ t = T-1, T-2, \ldots, 2, 1 \\ \mathbf{S}^* &= (s_1^*, s_2^*, \ldots, s_T^*) \end{aligned} \tag{4.13}$$

4.3.1.2 Generalisation to high-order, right-context HMMs

The right-context, best-path forward probability $\overleftarrow{\delta}_t(i_2, i_3, \ldots, i_{R+1})$ is defined as

$$\overleftarrow{\delta}_t(i_2, i_3, \ldots, i_{R+1}) = \max_{\mathbf{S}_1^{t-1}} \left[P(\mathbf{X}_1^t, \mathbf{S}_1^{t-1} | s_t = i_2, \ldots, s_{t+R-1} = i_{R+1}, \Phi_{\text{rc}}) \right] \tag{4.14}$$

$\overleftarrow{\delta}_t(i_2, i_3, \ldots, i_{R+1})$ is the probability of the most likely state sequence $\mathbf{S}_1^{t+R-1}$, which has generated the partial observation sequence $\mathbf{X}_1^t$, given that the state sequence ends at time $t+R-1$ in $\mathbf{S}_t^{t+R-1} = \{s_t = i_2, s_{t+1} = i_3, \ldots, s_{t+R-1} = i_{R+1}\}$ and the HMM Φ_{rc}. $\overleftarrow{\Psi}^{\delta}_t(i_2, i_3, \ldots, i_{R+1})$ is the back pointer of state sequence $\{s_t = i_2, s_{t+1} = i_3, \ldots, s_{t+R-1} = i_{R+1}\}$ and denotes the state most likely to precede the state sequence at time $t-1$. The following induction procedure is used to calculate $\overleftarrow{\delta}_t(i_2, i_3, \ldots, i_{R+1})$:

1. Initialisation: $(t = 1; i_1, i_2, \ldots, i_R = 1, 2, \ldots, N)$

$$\begin{aligned} \overleftarrow{\delta}_1(i_1, i_2, \ldots, i_R) &= \overleftarrow{a}_{0 i_1 i_2 i_R} b_{i_1}(\mathbf{x}_1) \\ \overleftarrow{\Psi}^{\delta}_1(i_1, i_2, \ldots, i_R) &= 0 \end{aligned} \tag{4.15}$$

2. Induction: $(t = 2, 3, \ldots, T-R;\ i_2, \ldots, i_{R+1} = 1, 2, \ldots, N)$

$$\begin{aligned} \overleftarrow{\delta}_t(i_2, i_3, \ldots, i_{R+1}) &= \left[\max_{i_1} \overleftarrow{\delta}_{t-1}(i_1, i_2, \ldots, i_R) \overleftarrow{a}_{i_1 i_2 \cdots i_{R+1}} \right] b_{i_2}(\mathbf{x}_t) \\ \overleftarrow{\Psi}^{\delta}_t(i_2, i_3, \ldots, i_{R+1}) &= \arg\max_{i_1} \left[\overleftarrow{\delta}_{t-1}(i_1, i_2, \ldots, i_R) \overleftarrow{a}_{i_1 i_2 \cdots i_{R+1}} \right] \end{aligned} \tag{4.16}$$

3. Induction: $(t = T-R+1, \ldots, T;\ i_{t-T+R}, \ldots, i_R = 1, 2, \ldots, N)$

$$\begin{aligned} &\overleftarrow{\delta}_t(i_{t-T+R}, i_{t-T+R+1}, \ldots, i_R) = \\ &\max_{i_{t-T+R-1}} \left[\begin{array}{l} \overleftarrow{\delta}_{t-1}(i_{t-T+R-1}, i_{t-T+R}, \ldots, i_R) \times \\ \overleftarrow{a}_{i_{t-T+R-1} i_{t-T+R} \cdots i_R (N+1)} \end{array} \right] b_{i_{t-T+R}}(\mathbf{x}_t) \\ &\overleftarrow{\Psi}^{\delta}_t(i_{t-T+R+1}, i_{t-T+R+2}, \ldots, i_{R+1}) = \\ &\arg\max_{i_{t-T+R}} \left[\begin{array}{l} \overleftarrow{\delta}_{t-1}(i_{t-T+R}, i_{t-T+R+1}, \ldots, i_R + 1) \times \\ \overleftarrow{a}_{i_{t-T+R} i_{t-T+R+1} \cdots i_R i_{R+1} (N+1)} \end{array} \right] \end{aligned} \tag{4.17}$$

4. Termination:

$$\begin{aligned} \overleftarrow{\delta}_T &= \max_{i_1} \left[\overleftarrow{\delta}_T(i_1) \overleftarrow{a}_{i_1 (N+1)} \right] \\ s_T^* &= \arg\max_{i_1} \left[\overleftarrow{\delta}_T(i_1) \overleftarrow{a}_{i_1 (N+1)} \right] \end{aligned} \tag{4.18}$$

5. Backtracking:

$$\begin{aligned} s_t^* &= \begin{cases} \overleftarrow{\Psi}^{\delta}_{t+1}(s^*_{t+1}, s^*_{t+2}, \cdots, s^*_T), & t = T-1, T-2, \ldots, T-R+1 \\ \overleftarrow{\Psi}^{\delta}_{t+1}(s^*_{t+1}, s^*_{t+2}, \cdots, s^*_{t+R}), & t = T-R, \ldots, 2, 1 \end{cases} \\ \mathbf{S}^* &= (s_1^*, s_2^*, \ldots, s_T^*) \end{aligned} \tag{4.19}$$

Note that the R^{th}-order, best-path forward probability $\overleftarrow{\delta}_t(i_2, i_3, \ldots, i_{R+1})$ of a right-context HMM is now the *conditional* probability of the observation sequence *given* the state sequence. It also excludes the contribution of the observation sequence $\mathbf{X}_{t+1}^{t+R-1}$, although it is given the information that the HMM occupies the state sequence $\mathbf{S}_{t+1}^{t+R-1} = (s_{t+1} = i_3 \ldots, s_{t+R-1} = i_{R+1})$. Thus $\overleftarrow{\delta}_t(i_2, i_3, \ldots, i_{R+1})$ has the same time-asynchronicity between the observations and the sequence of states under consideration as the R^{th} order, best-path backward probability $\overrightarrow{\epsilon}'_t(i_1, i_2, \ldots, i_R)$ of a left-context HMM has. We therefore suspect that the forward Viterbi-ML-beam decoding of a right-context HMM will be significantly less efficient than the backward Viterbi-MAP-beam decoding (the opposite is true when decoding left-context HMMs).

4.3.2 Backward Viterbi-beam decoding of right-context HMMs

In this section we present the backward Viterbi-beam decoding of right-context HMMs. This decoding algorithm will be used to calculate heuristic functions using the derived $(R-K)$-order HMMs in both the FBS-Viterbi-beam decoder and the A* search decoder.

4.3.2.1 Backward Viterbi decoding of first-order, right-context HMMs

In a manner similar to the first-order, best-path forward probability $\overleftarrow{\delta}_t(i)$ we define the first-order, best-path backward probability to be

$$\overleftarrow{\epsilon}_t(i) = \max_{\mathbf{S}_{t+1}^T} \left[P(\mathbf{X}_t^T, s_t = i, \mathbf{S}_{t+1}^T | \Phi_{\text{rc}}) \right] \tag{4.20}$$

i.e. the probability of the most likely state sequence $\mathbf{S}_t^T$, which has generated the partial observation sequence $\mathbf{X}_t^T$, and ends at time t in state $s_t = i$, given the HMM Φ_{rc}. We define $\overleftarrow{\Psi}^{\epsilon}_t(i)$ to be the "forward" pointer of state i at time t and it denotes the state most likely to follow state i at time $t+1$. The following induction procedure is used to calculate $\overleftarrow{\epsilon}_t(i)$:

1. Initialisation: $(t = T)$

$$\left.\begin{aligned} \overleftarrow{\epsilon}_T(i) &= b_i(\mathbf{x}_T)\overleftarrow{a}_{i(N+1)} \\ \overleftarrow{\Psi}^{\epsilon}_T(i) &= N+1 \end{aligned}\right\} i = 1, \ldots, N \tag{4.21}$$

2. Induction: $(t = T-1, T-2, \ldots, 2, 1)$

$$\left.\begin{aligned} \overleftarrow{\epsilon}_t(i) &= b_i(\mathbf{x}_t) \max_j [\overleftarrow{a}_{ij} \overleftarrow{\epsilon}_{t+1}(j)] \\ \overleftarrow{\Psi}^{\epsilon}_t(i) &= \arg\max_j [\overleftarrow{a}_{ij} \overleftarrow{\epsilon}_{t+1}(j)] \end{aligned}\right\} i = 1, \ldots, N \tag{4.22}$$

3. Termination:

$$\begin{aligned} \overleftarrow{\epsilon}_1 &= \max_i [\overleftarrow{a}_{0i} \overleftarrow{\epsilon}_1(i)] \\ s_1^* &= \arg\max_i [\overleftarrow{a}_{0i} \overleftarrow{\epsilon}_1(i)] \end{aligned} \tag{4.23}$$

4. Backtracking:

$$\begin{aligned} s_t^* &= \overleftarrow{\Psi}^{\epsilon}_{t-1}(s_{t-1}^*),\ t = 2, 3, \ldots, T-1, T \\ \mathbf{S}^* &= (s_1^*, s_2^*, \ldots, s_T^*) \end{aligned} \tag{4.24}$$

4.3.2.2 Generalisation to high-order, right-context HMMs

We define the R^{th}-order, best-path backward probability

$$\overleftarrow{\epsilon}_t(i_1, i_2, \ldots, i_R) = \max_{\mathbf{S}_{t+R}^T} \left[P\left(\mathbf{X}_t^T, s_t = i_1, \ldots, s_{t+R-1} = i_R, \mathbf{S}_{t+R}^T | \Phi_{\text{rc}}\right)\right] \tag{4.25}$$

i.e. the probability of the most likely state sequence $\mathbf{S}_t^T$, which has generated the partial observation sequence $\mathbf{X}_t^T$ and starts at time t with state sequence $\mathbf{S}_t^{t+R-1} = \{s_t = i_1, s_{t+1} = i_2, \ldots, s_{t+R-1} = i_R\}$, given the HMM Φ_{rc}. $\overleftarrow{\Psi}^{\epsilon}_t(i_1, i_2, \ldots, i_R)$ is the "forward" pointer at time t of the state sequence $\{s_t = i_1, s_{t+1} = i_2, \ldots, s_{t+R-1} = i_R\}$ and denotes the most likely state to follow the state sequence at time $t+1$. The following induction procedure is used to calculate the backward best-path probability:

1. Initialisation: $(t = T-1, \ldots, T-R+1; i_1, \ldots, i_R = 1, 2, \cdots, N)$

$$\begin{aligned} \overleftarrow{\epsilon}_T(i_R) &= b_{i_R}(\mathbf{x}_T) \overleftarrow{a}_{i_R(N+1)} \\ \overleftarrow{\epsilon}_t(i_{t-T+R}, \ldots, i_R) &= b_{i_{t-T+R}}(\mathbf{x}_t) \overleftarrow{a}_{i_{t-T+R} \cdots i_R(N+1)} \overleftarrow{\epsilon}_{t+1}(i_{t-T+R+1}, \ldots, i_R) \\ \overleftarrow{\Psi}^{\epsilon}_T(i_1, i_2, \ldots, i_R) &= N+1 \end{aligned} \tag{4.26}$$

2. Induction: $(t = T-R, T-R-1, \ldots, 1;\ i_1, \ldots, i_R = 1, 2, \cdots, N)$

$$\begin{aligned} \overleftarrow{\epsilon}_t(i_1, i_2, \ldots, i_R) &= b_{i_1}(\mathbf{x}_t) \max_{i_{R+1}} \left[\ \overleftarrow{a}_{i_1 i_2 \cdots i_{R+1}} \overleftarrow{\epsilon}_{t+1}(i_2, i_3, \ldots, i_{R+1}) \ \right] \\ \overleftarrow{\Psi}^{\epsilon}_{t+R-1}(i_1, i_2, \ldots, i_R) &= \arg\max_{i_{R+1}} \left[\ \overleftarrow{a}_{i_1 i_2 \cdots i_{R+1}} \overleftarrow{\epsilon}_{t+1}(i_2, i_3, \ldots, i_{R+1}) \ \right] \end{aligned} \tag{4.27}$$

3. Termination:

$$\begin{aligned} \overleftarrow{\epsilon}_1 &= \max_{i_1, \ldots, i_R} [\overleftarrow{a}_{0 i_1 \cdots i_R} \overleftarrow{\epsilon}_1(i_1, \ldots, i_R)] \\ \{s_1^*, \ldots, s_R^*\} &= \arg\max_{i_1, \ldots, i_R} [\overleftarrow{a}_{0 i_1 \cdots i_R} \overleftarrow{\epsilon}_1(i_1, \ldots, i_R)] \end{aligned} \tag{4.28}$$

4. Backtracking:

$$\begin{aligned} s_t^* &= \overleftarrow{\Psi}^{\epsilon}_{t-1}(s^*_{t-R}, s^*_{t-R+1}, \cdots, s^*_{t-1}),\ t = R+1, R+2, \ldots, T-1, T \\ \mathbf{S}^* &= (s_1^*, s_2^*, \ldots, s_T^*) \end{aligned} \tag{4.29}$$

4.4 Evaluating decoding of right-context HMMs

Since we have defined the right-context HMM and have adapted the Viterbi-beam decoder we are now ready to test whether the backward decoding of high-order, right-context HMMs is computationally equivalent to the forward decoding of high-order, left-context HMMs.

We repeat the previous decoding experiment using the equivalent right-context HMMs. We use the same experimental setup for decoding right-context HMMs as was used in Sections 2.2.4 and 3.3.4 when decoding left-context HMMs. We use both the German and English development set to evaluate the decoding. The German development set represents the scenario where the training and evalution conditions are matched. The evaluation performed on the English development set represents the scenario where there is a mismatch between the training and evaluation conditions.

The parameters of the right-context HMMs were estimated concurrently with the parameters of the left-context HMMs, using the FIT algorithm. The purpose of the experiment is to measure and compare the forward and backward search cost of right-context, high-order HMMs.

4.4.1 Measuring decoder performance

As before, we use the search cost C_s (the number of transition probabilities evaluated during decoding) as an implementation independent measure of the performance of the backward Viterbi-beam decoder. Over the 1410 segments decoded we compute the average number of evaluated transition probabilities. We compute the normalised search cost $C_{s,n}$ by normalising the number of evaluated transition probabilities with the maximum number of transition probabilities that the standard Viterbi decoder would evaluate.

4.4.2 Results

As in the two previous experiments, the following two beam-pruning configurations are investigated as the order of the HMMs increase:

- The beam-width is set to a constant $B = 20.0$.
- The minimum integer beam-width is determined for which the decoder correctly decodes all 1410 segments.

Constant beam-width

Figure 4.2: *(a) The search cost (C_s) of the forward Viterbi-ML-beam and backward Viterbi-MAP-beam decoders, during the decoding of high-order, right-context HMMs, when the decoders are using a constant beam-width of $B = 20.0$. (b) The normalised search cost ($C_{s,n}$) of the forward Viterbi-ML-beam and backward Viterbi-MAP-beam decoders, during the decoding of high-order, right-context HMMs, when the decoders are using a constant beam-width of $B = 20.0$.*

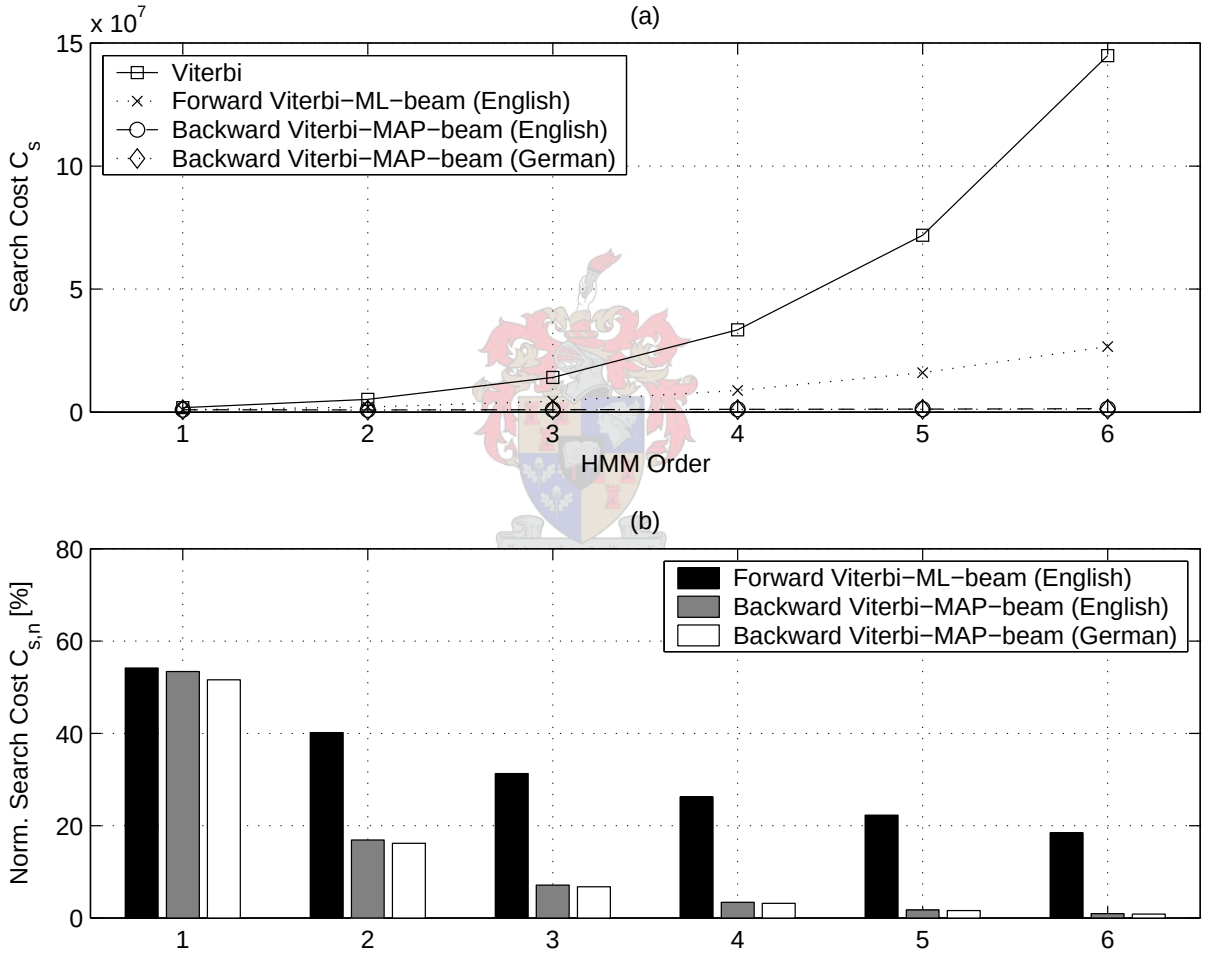


Fig. 4.2 illustrates the results for Viterbi decoding, *forward* Viterbi-ML-beam decoding and *backward* Viterbi-MAP-beam decoding when a logarithmic beam of $B = 20.0$. Table 4.1 tabulates the accuracy of decoding the different order HMMs, when using a constant beam. We have included the forward Viterbi-ML-beam decoding results in order to test our prediction that forward Viterbi-ML-beam decoding is computationally more expensive than backward Viterbi-MAP-beam decoding (of right-context HMMs).

As suspected the forward Viterbi-ML-beam decoder is significantly less efficient than the backward Viterbi-MAP-beam decoder. The right-context HMM seems to exhibit

Table 4.1: *The minimum computational expense of the Viterbi-MAP-beam decoder, during the backward decoding of high-order, right-context HMMs, when the decoder is using a constant beam-width of $B = 20.0$.*

	Search cost C_s		**Norm. cost $C_{s,n}$ [%]**		**Decoding Accuracy [%]**	
Order	**English**	**German**	**English**	**German**	**English**	**German**
1	987,825	954,282	53.40	51.58	98.37	99.93
2	871,345	834,039	16.87	16.14	98.23	99.86
3	1,005,864	952,662	7.12	6.75	99.15	99.86
4	1,132,977	1,062,294	3.39	3.18	99.09	99.51
5	1,247,568	1,153,796	1.74	1.60	98.37	98.68
6	1,358,567	1,228,074	0.94	0.86	95.23	96.39

the same decoding behaviour as the left-context HMM when the order is increased. It is interesting to note that, when the beam-width is kept constant, the search cost of the matched scenario (the German development set) and the mismatched scenario (the English development set) is very similar.

Minimum computational expense of decoder when correctly decoding all segments

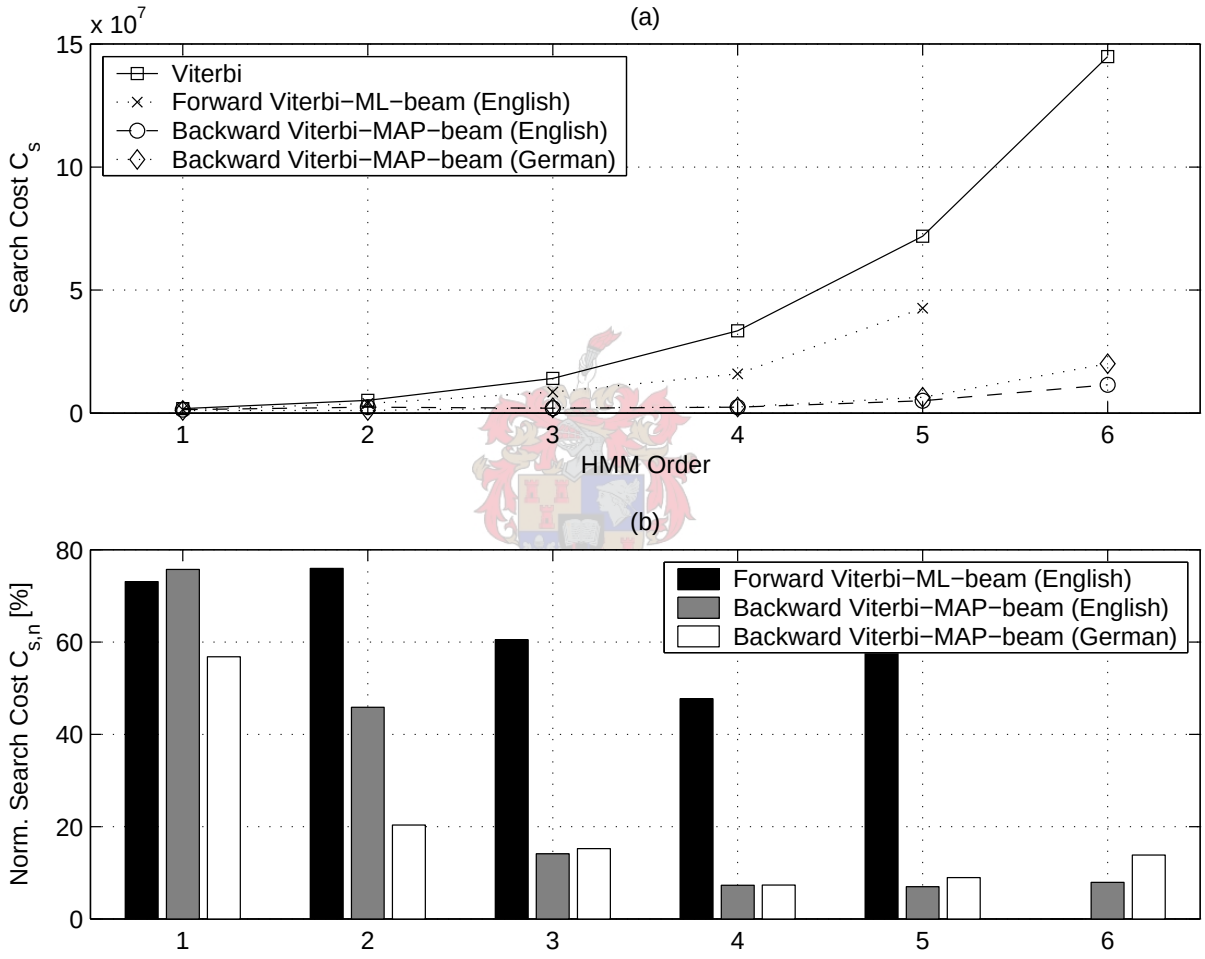
Fig. 4.3 illustrates the results for Viterbi decoding, forward Viterbi-ML-beam and backward Viterbi-MAP-beam decoding when all segments are decoded correctly. Table 4.2 tabulates the minimum integer beam-width required to correctly decode all segments. Once again we see that, upto the order of $R = 4$, the backward search cost is similar in both the matched and mismatched scenarios. When the HMM order is further increased, the minimum required beam-width increases significantly for the German development set (the matched scenario).

A comparison of the decoding of equivalent left- and right-context HMMs

Fig. 4.4 compares the search cost as well as the normalised search cost of equivalent left-context and right-context HMMs, when a constant beam-width of $B = 20.0$ is used. It can be clearly seen that the backward decoding of the right-context HMM is as efficient as the forward decoding of the left-context HMM, when the beam-width is kept constant.

It is interesting to note that decoding the second-order HMM is computationally less expensive than decoding the first-order HMM. Higher order HMMs are more complex and are typically better able to model the training data, but with the increased complexity comes the danger of having high-order transition probabilities that are poorly estimated as a result of insufficient training data. We suspect that the second-order HMM is less

Figure 4.3: *(a) The minimum search cost (C_s) of the forward Viterbi-ML-beam and backward Viterbi-MAP-beam decoders, during the decoding of high-order, right-context HMMs, when the decoders correctly decode all segments. (b) The normalised search cost ($C_{s,n}$)of the forward Viterbi-ML-beam and backward Viterbi-MAP-beam decoders, during the decoding of high-order, right-context HMMs, with beams set wide enough to decode all segments correctly.*



expensive to decode, since it is a better representation of the data than the first-order HMM, and the order of the HMM is still low enough that the second-order transition probabilities are well estimated.

Fig. 4.5 compares the search cost (the number of evaluated transitions) as well as the normalised search cost of equivalent left-context and right-context HMMs, when all segments are decoded correctly. It is interesting to note that, for the mismatched scenario (the English development set), there is a much larger difference between the computational expense of forward and backward decoding than in Fig. 4.4. It could be that backward decoding the right-context HMM is fundamentally more efficient than forward-decoding

the left-context HMM. Since increasing the beam-width until all segments are decoded correctly is sensitive to outliers in the data, we suspect that it is more likely that the difference in computational expense is random. This suspicion is supported by the results of the matched scenario (the German development set), which shows an even bigger difference between the computational expense of forward and backward decoding. In order to verify this suspicion, it would be useful to repeat the comparison of backward and forward decoding on some of the other language pairs of the CallFriend speech corpus.

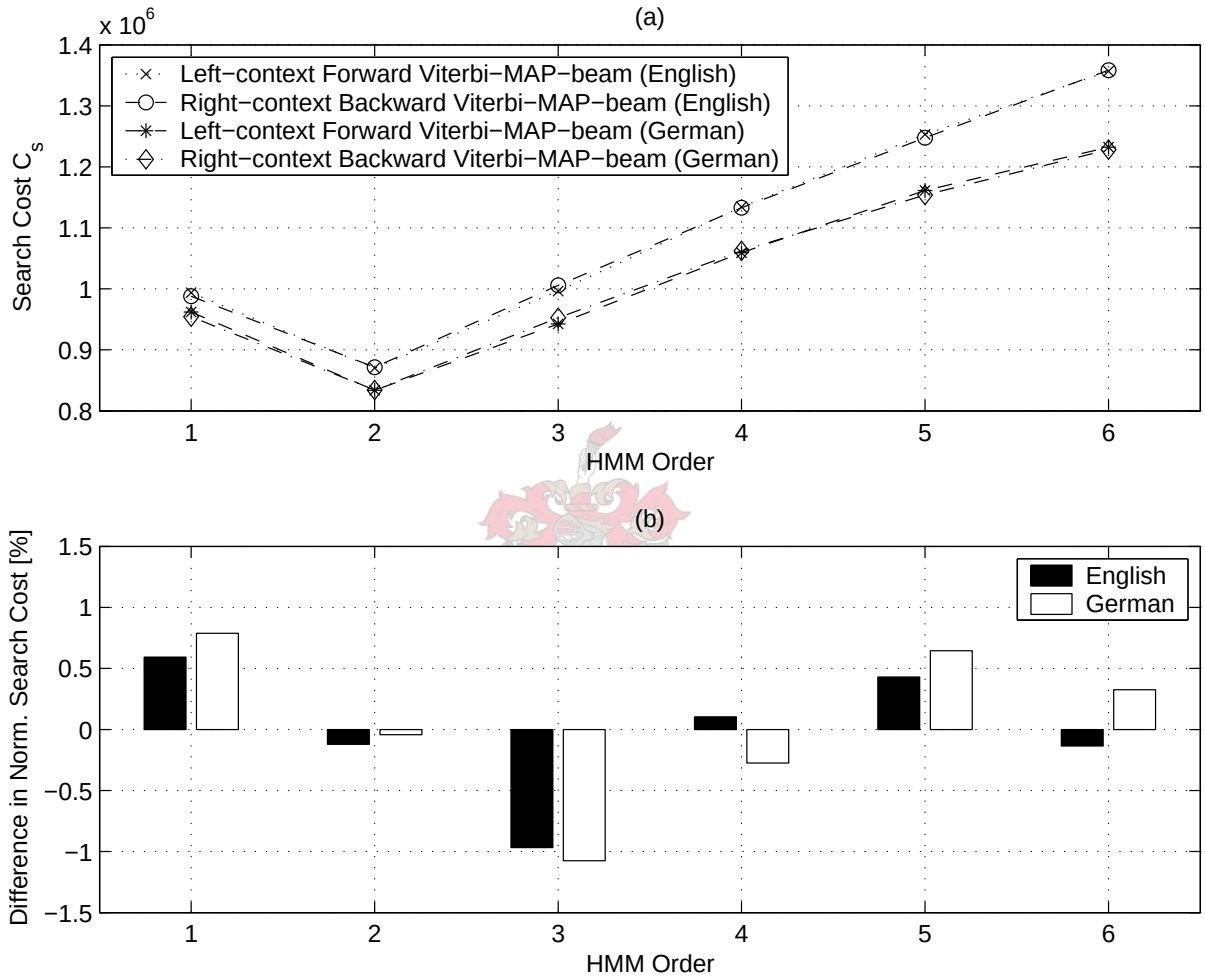
4.4.3 Discussion

The results seem to indicate that the 'time-synchronicity' between observations at the states under consideration is the reason why forward Viterbi-beam decoding is less expensive than backward decoding of left-context HMMs. This statement is supported by the fact that, when the beam-widths are equal, backward Viterbi-beam decoding of right-context HMMs is as efficient as forward decoding of left-context HMMs. The fact that forward decoding of right-context HMMs, which now also suffer from time-asynchronicity of observations and state sequences, is computationally more expensive than backward decoding, also supports this statement. We now have a computationally efficient method of determining the right-context, best-path backward probability. Unfortunately, we cannot directly use the right-context, best-path backward probability when using the proposed decoders to forward decode high-order, left-context HMMs, since $\overrightarrow{\epsilon}'_t(i_1, \ldots, i_R) \neq \overleftarrow{\epsilon}_t(i_1, \ldots, i_R)$. In the following section we will derive the relationship between left-context and right-context, best-path backward probabilities.

Table 4.2: *The minimum computational expense of the Viterbi-MAP-beam decoder, during the backward decoding of high-order, right-context HMMs, with beams set wide enough to decode all segments correctly.*

	Search cost C_s		**Norm. cost $C_{s,n}$ [%]**		**Beam-width B**	
Order	**English**	**German**	**English**	**German**	**English**	**German**
1	1,400,373	1,051,089	75.70	56.82	31.0	22.0
2	2,369,132	1,051,343	45.86	20.35	38.0	23.0
3	1,994,318	2,147,897	14.12	15.23	28.0	30.0
4	2,439,604	2,453,055	7.30	7.34	28.0	29.0
5	5,010,755	6,415,398	6.97	8.93	35.0	40.0
6	11,466,957	20,042,568	7.91	13.96	45.0	58.0

Figure 4.4: *A comparison of (a) the search cost, and (b) the normalised search cost of the Viterbi-MAP-beam decoder (during the forward decoding of high-order, left-context HMMs) and the Viterbi-MAP-beam decoder (during the backward decoding of high-order, right-context HMMs), when the decoders are using a constant beam-width of* $B = 20.0$.

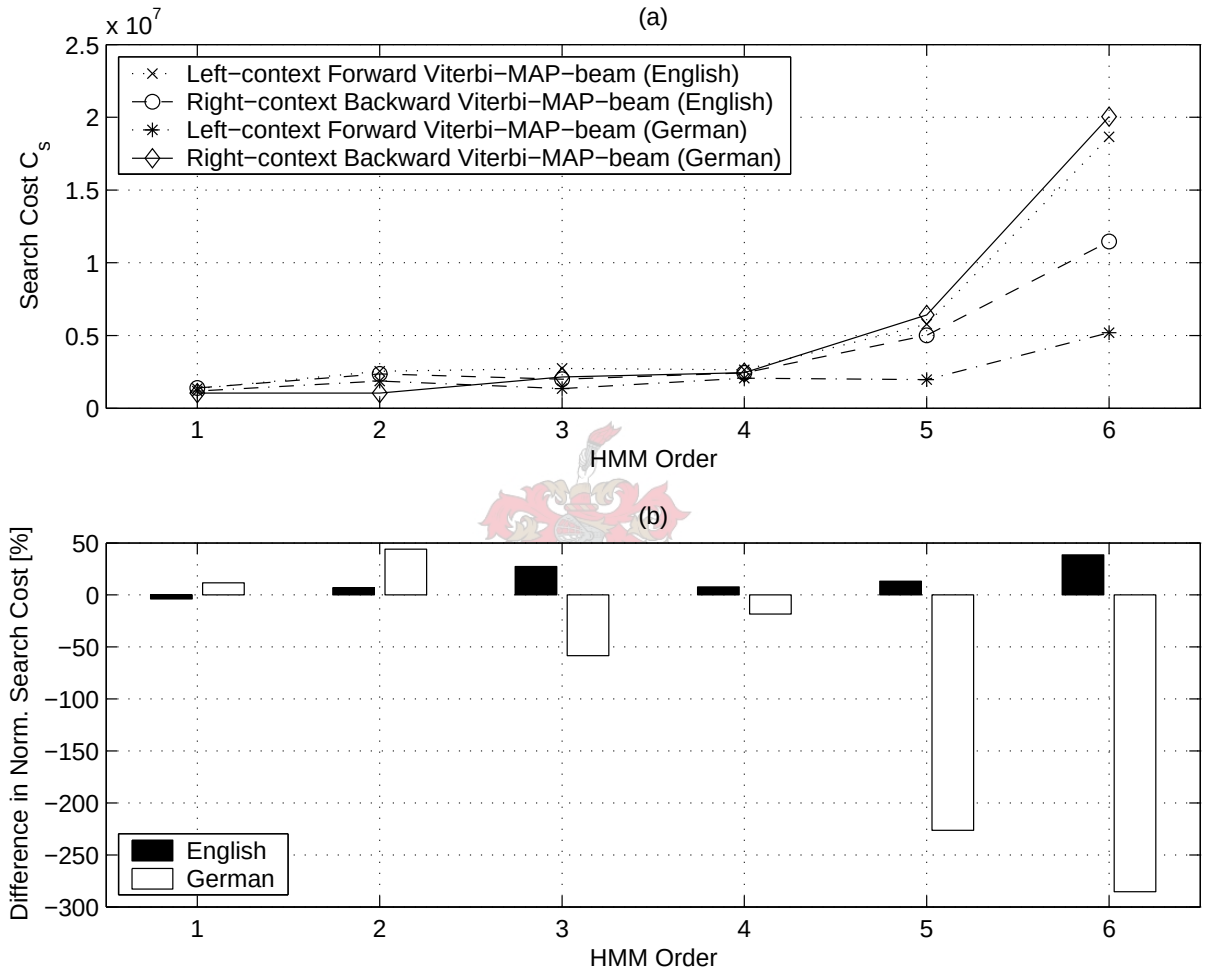


4.5 Computing Heuristics by using Right-context HMMs

We have managed to make the backward Viterbi-beam decoding of right-context HMMs computationally efficient and the algorithm independent of the order of the HMM. Our newly proposed decoders combines the left-context, best-path forward and backward probabilities, when performing state pruning. We therefore need to relate the left-context, best-path backward probability to the right-context, best-path backward probability.

For $t = R, R+1, \ldots, T$, the relationship between the two backward probabilities can

Figure 4.5: *A comparison of (a) the search cost, and (b) the normalised search cost of the Viterbi-MAP-beam decoder (during the forward decoding of high-order left-context HMMs) and the Viterbi-MAP-beam decoder (during the backward decoding of high-order, right-context HMMs), with beams set wide enough to decode all segments correctly.*



be derived as follows:

$$
\begin{aligned}
& \overleftarrow{\epsilon}_{t-R+1}(i_1,\ldots,i_R) \\
= \; & \max_{\mathbf{S}_{t+1}^T} \left[P(\mathbf{X}_{t-R+1}^T, \mathbf{S}_{t+1}^T, s_{t-R+1}=i_1,\ldots,s_t=i_R|\Phi)\right] \\
= \; & b_{i_1}(\mathbf{x}_{t-R+1})\ldots b_{i_{R-1}}(\mathbf{x}_{t-1}) \max_{\mathbf{S}_{t+1}^T} \left[P(\mathbf{X}_t^T, \mathbf{S}_{t+1}^T, s_{t-R+1}=i_1,\ldots,s_t=i_R|\Phi)\right] \\
= \; & b_{i_1}(\mathbf{x}_{t-R+1})\ldots b_{i_{R-1}}(\mathbf{x}_{t-1}) \max_{\mathbf{S}_{t+1}^T} \left[\begin{array}{l} P(\mathbf{X}_t^T, \mathbf{S}_{t+1}^T|s_{t-R+1}=i_1,\ldots,s_t=i_R,\Phi)\times \\ P(s_{t-R+1}=i_1,\ldots,s_t=i_R|\Phi) \end{array}\right] \\
= \; & b_{i_1}(\mathbf{x}_{t-R+1})\ldots b_{i_{R-1}}(\mathbf{x}_{t-1}) P(s_{t-R+1}=i_1,\ldots,s_t=i_R|\Phi) \times \\
& \max_{\mathbf{S}_{t+1}^T} \left[P(\mathbf{X}_t^T, \mathbf{S}_{t+1}^T|s_{t-R+1}=i_1,\ldots,s_t=i_R,\Phi)\right]
\end{aligned}
$$

$$= b_{i_1}(\mathbf{x}_{t-R+1}) \dots b_{i_{R-1}}(\mathbf{x}_{t-1}) P(s_{t-R+1} = i_1, \dots, s_t = i_R|\Phi) \overrightarrow{\epsilon}'_t(i_1, \dots, i_R) \quad (4.30)$$

According to Section 4.3.2.2, for $t = R-1, R-2, \dots, 2, 1$ the left-context, best-path backward probability is:

$$\begin{aligned}
& \overrightarrow{\epsilon}'_t(i_1, i_2, \dots, i_t) \\
= \; & b_{i_t}(\mathbf{x}_t) \max_{i_{t+1}} \left[\overrightarrow{a}_{0 i_1 \cdots i_{t+1}} \overrightarrow{\epsilon}'_{t+1}(i_1, i_2, \dots, i_{t+1}) \right] \\
= \; & \max_{i_{t+1}} \left[b_{i_t}(\mathbf{x}_t) \overrightarrow{a}_{0 i_1 \cdots i_t i_{t+1}} b_{i_{t+1}}(\mathbf{x}_{t+1}) \max_{i_{t+2}} \left[\overrightarrow{a}_{0 i_1 \cdots i_{t+1} i_{t+2}} \overrightarrow{\epsilon}'_{t+1}(i_1, \dots, i_{t+1}, i_{t+2}) \right] \right] \\
= \; & \max_{i_{t+1}, \cdots, i_R} \left[\begin{array}{l} b_{i_t}(\mathbf{x}_t) \overrightarrow{a}_{0 i_1 \cdots i_t i_{t+1}} b_{i_{t+1}}(\mathbf{x}_{t+1}) \overrightarrow{a}_{0 i_1 \cdots i_{t+1} i_{t+2}} b_{i_{t+2}}(\mathbf{x}_{t+2}) \dots \\ b_{i_{R-1}}(\mathbf{x}_{R-1}) \overrightarrow{a}_{0 i_1 \cdots i_t i_{t+1} \cdots i_R} \overrightarrow{\epsilon}'_R(i_1, i_2, \dots, i_t, i_{t+1}, \dots, i_R) \end{array} \right] \\
= \; & \max_{i_{t+1}, \cdots, i_R} \left[\begin{array}{l} b_{i_t}(\mathbf{x}_t) \overrightarrow{a}_{0 i_1 \cdots i_{t+1}} b_{i_{t+1}}(\mathbf{x}_{t+1}) \overrightarrow{a}_{0 i_1 \cdots i_{t+2}} b_{i_{t+2}}(\mathbf{x}_{t+2}) \dots \\ b_{i_{R-1}}(\mathbf{x}_{R-1}) \overrightarrow{a}_{0 i_1 \cdots i_t i_{t+1} \cdots i_R} \times \\ \frac{\overleftarrow{\epsilon}_1(i_1, i_2, \dots, i_t, i_{t+1}, \dots, i_R)}{b_{i_1}(\mathbf{x}_1) \dots b_{i_t}(\mathbf{x}_t) b_{i_{t+1}}(\mathbf{x}_{t+1}) \dots b_{i_{R-1}}(\mathbf{x}_{R-1}) P(s_1 = i_1, \dots, s_t = i_t, s_{t+1} = i_{t+1}, \dots, s_R = i_R|\Phi)} \end{array} \right] \\
= \; & \max_{i_{t+1}, \cdots, i_R} \left[\frac{\overrightarrow{a}_{0 i_1 \cdots i_t i_{t+1}} \overrightarrow{a}_{0 i_1 \cdots i_{t+2}} \dots \overrightarrow{a}_{0 i_1 \cdots i_t i_{t+1} \cdots i_R} \overleftarrow{\epsilon}_1(i_1, i_2, \dots, i_t, i_{t+1}, \dots, i_R)}{b_{i_1}(\mathbf{x}_1) \dots b_{i_{t-1}}(\mathbf{x}_{t-1}) P(s_1 = i_1, \dots, s_t = i_t, i_{t+1} = i_{t+1}, \dots, s_R = i_R|\Phi)} \right] \\
= \; & \max_{i_{t+1}, \cdots, i_R} \left[\frac{P(s_1 = 0, s_1 = i_1, \dots, s_t = i_t, s_{t+1} = i_{t+1}, \dots, s_R = i_R|\Phi) \overleftarrow{\epsilon}_1(i_1, i_2, \dots, i_t, i_{t+1}, \dots, i_R)}{P(s_1 = 0, s_1 = i_1, \dots, s_t = i_t|\Phi) b_{i_1}(\mathbf{x}_1) \dots b_{i_{t-1}}(\mathbf{x}_{t-1}) P(s_1 = i_1, \dots, s_t = i_t, s_{t+1} = i_{t+1}, \dots, s_R = i_R|\Phi)} \right] \\
= \; & \max_{i_{t+1}, \cdots, i_R} \left[\frac{\overleftarrow{a}_{0 i_1 \cdots i_t i_{t+1} \cdots i_R} \overleftarrow{\epsilon}_1(i_1, i_2, \dots, i_t, i_{t+1}, \dots, i_R)}{b_{i_1}(\mathbf{x}_1) \dots b_{i_{t-1}}(\mathbf{x}_{t-1}) P(s_1 = i_1, \dots, s_t = i_t|\Phi)} \right] \\
= \; & \frac{\max_{i_{t+1}, \cdots, i_R} \left[\overleftarrow{a}_{0 i_1 \cdots s_R} \overleftarrow{\epsilon}_1(i_1, i_2, \dots, i_t, i_{t+1}, \dots, s_R) \right]}{b_{i_1}(\mathbf{x}_1) \dots b_{i_{t-1}}(\mathbf{x}_{t-1}) P(s_1 = i_1, \dots, s_t = i_t|\Phi)}
\end{aligned} \quad (4.31)$$

Thus, the left-context, best-path backward probability can be written in terms of the right-context, best-path backward probability as follows:

$$\begin{aligned}
& t = T, T-1, \dots, R: \\
& \quad \overrightarrow{\epsilon}'_t(i_1, i_2, \dots, i_R) = \frac{\overleftarrow{\epsilon}_{t-R+1}(i_1, i_2, \dots, i_R)}{b_{i_1}(\mathbf{x}_{t-R+1}) \dots b_{i_{R-1}}(\mathbf{x}_{t-1}) P(s_{t-R+1} = i_1, \dots, s_t = i_R|\Phi)} \\
& t = R-1, \dots, 1: \\
& \quad \overrightarrow{\epsilon}'_t(i_1, i_2, \dots, i_t) = \frac{\max_{i_{t+1}, \cdots, i_R} \left[\overleftarrow{a}_{0 i_1 \cdots s_R} \overleftarrow{\epsilon}_1(i_1, i_2, \dots, i_t, i_{t+1}, \dots, s_R) \right]}{b_{i_1}(\mathbf{x}_1) \dots b_{i_{t-1}}(\mathbf{x}_{t-1}) P(s_1 = i_1, \dots, s_t = i_t)}
\end{aligned} \quad (4.32)$$

Upon examination of Eq. 4.32 we see that we need to convert the right-context, best-path backward probability to a left-context, best-path backward probability, before we can incorporate the probability into our new decoders. However, the conversion requires extra multiplications and is also dependent on the order of the HMM being decoded. Therefore, it might seem as if we have not gained anything by using a right-context HMM to calculate the best-path backward probability. We could simply have performed the extra multiplications when backward decoding the left-context HMM (as mentioned in Section 3.3.4.3). However, this would require at least R extra calculations for each state

in the search graph. In the worst-case scenario, where little or no state pruning occurs during the backward Viterbi-beam decoding of the $(R-K)$-order HMM, the conversion will require $(N_{feq})T(R-K)$ extra calculations, where N_{feq} is the number of emitting states of the first-order equivalent $(R-K)$-order HMM.

We therefore propose an alternative method of converting the right-context, best-path backward probabilities to left-context, best-path backward probabilities that will require at most $2(N_{feq})T$ extra calculations. This alternative method uses the "forward" pointers $\overleftarrow{\Psi}^{\epsilon}_{t}(i_1, i_2, \ldots, i_R)$ which are determined as part of the backward Viterbi-beam decoding process. The process is similar to the backtracking that is used when obtaining the optimal state sequence, except that backtracking is performed for all states and all time indexes.

4.5.1 Alternate conversion of best-path backward probability

If we are given the left-context, best-path "forward" pointers $\overrightarrow{\Psi}^{\epsilon}_{t}(i_1, i_3, \ldots, i_R)$, we could easily calculate the left-context, best-path backward probability $\overrightarrow{\epsilon}'_{t}(i_1, i_3, \ldots, i_R)$. This is done by starting at the final state $s_T = N+1$ and recursively calculating $\overrightarrow{\epsilon}_{t}(i_1, \ldots, i_R)$ as:

$$\begin{aligned}
&\text{For } t = T-1, \ldots, R: \\
&\qquad \overrightarrow{\epsilon}'_{t}(i_1, \ldots, i_R) = b_{i_R}(\mathbf{x}_t)\,\overrightarrow{a}_{i_1 \cdots i_{R+1}}\,\overrightarrow{\epsilon}'_{t+1}(i_2, i_3, \ldots, \overrightarrow{\Psi}^{\epsilon}_{t}(i_1, \ldots, i_R)) \\
&\text{For } t = R-1, \ldots, 1: \\
&\qquad \overrightarrow{\epsilon}'_{t}(i_1, \ldots, i_t) = b_{i_t}(\mathbf{x}_t)\,\overrightarrow{a}_{0 i_1 \cdots i_{t+1}}\,\overrightarrow{\epsilon}'_{t+1}(i_1, i_2, \ldots, \overrightarrow{\Psi}^{\epsilon}_{t}(i_1, \ldots, i_t))
\end{aligned} \tag{4.33}$$

and noting that

$$\begin{aligned}
\overrightarrow{\epsilon}'_{T}(i_1, \ldots, i_R, N+1) &= 1.0 \\
\overrightarrow{\Psi}^{\epsilon}_{T}(i_1, \ldots, i_R) &= N+1
\end{aligned} \tag{4.34}$$

In the previous section we have proven the relationship between the left-context, best-path forward probability and the right-context, best-path forward probability. This can be used to prove that the left-context forward pointers are equal to the right context forward pointers $\overrightarrow{\Psi}^{\epsilon}_{t}(i_1, \ldots, i_R) = \overleftarrow{\Psi}^{\epsilon}_{t}(i_1, \ldots, i_R)$. For $t = T, T-1, \ldots, R$ the left-context "forward" pointers can be determined as follows:

$$\begin{aligned}
&\overrightarrow{\Psi}^{\epsilon}_{t}(i_1, i_2, \ldots, i_R) \\
&= \arg\max_{i_{R+1}} \left[\; \overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}}\,\overrightarrow{\epsilon}'_{t+1}(i_2, i_3, \ldots, i_{R+1}) \;\right] \\
&= \arg\max_{i_{R+1}} \left[\frac{\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}}\,\overleftarrow{\epsilon}_{t-R+2}(i_2, i_3, \ldots, i_{R+1})}{b_{i_2}(\mathbf{x}_{t-R+2}) \ldots b_{i_R}(\mathbf{x}_t) P(s_{t-R+2} = i_2, \ldots, s_{t+1} = i_{R+1} | \Phi)} \right] \\
&= \arg\max_{i_{R+1}} \left[\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}} \frac{\overleftarrow{\epsilon}_{t-R+2}(i_2, i_3, \ldots, i_{R+1})}{P(s_{t-R+2} = i_2, \ldots, s_{t+1} = i_{R+1} | \Phi)} \right]
\end{aligned}$$

$$
\begin{aligned}
&= \arg\max_{i_{R+1}} \left[\frac{P(s_{t-R+1}=i_1,\ldots,s_{t+1}=i_{R+1}|\Phi)\overleftarrow{\epsilon}_{t-R+2}(i_2,i_3,\ldots,i_{R+1})}{P(s_{t-R+1}=i_1,\ldots,s_t=i_R|\Phi)P(s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1}|\Phi)} \right] \\
&= \arg\max_{i_{R+1}} \left[\frac{P(s_{t-R+1}=i_1,\ldots,s_{t+1}=i_{R+1}|\Phi)\overleftarrow{\epsilon}_{t-R+2}(i_2,i_3,\ldots,i_{R+1})}{P(s_{t-R+2}=i_2,\ldots,s_{t+1}=i_{R+1}|\Phi)} \right] \\
&= \arg\max_{i_{R+1}} \left[\overleftarrow{a}_{i_1\ldots i_{R+1}} \overleftarrow{\epsilon}_{t-R+2}(i_2,i_3,\ldots,i_{R+1}) \right] \\
&= \overleftarrow{\Psi}^{\epsilon}_{t-R+1}(i_1,i_2,\ldots,i_R) \qquad (4.35)
\end{aligned}
$$

Thus, for $t \geq R$ the left-context and right-context "forward" pointers are identical. For $t = R-1,\ldots,1$ the left-context "forward" pointers can be determined as follows:

$$
\begin{aligned}
&\overrightarrow{\Psi}^{\epsilon}_{t}(i_1,i_2,\ldots,i_t) \\
&= \arg\max_{i_{t+1}} \left[\overrightarrow{a}_{0i_1i_2\cdots i_{t+1}} \overrightarrow{\epsilon}'_{t+1}(i_1,i_2,\ldots,i_{t+1}) \right] \\
&= \arg\max_{i_{t+1}} \left[\frac{\overrightarrow{a}_{0i_1i_2\cdots i_{t+1}} \max_{i_{t+2},\cdots,i_R} \left[\overleftarrow{a}_{0i_1\cdots s_R} \overleftarrow{\epsilon}_1(i_1,\ldots,i_{t+1},i_{t+2},\ldots,s_R) \right]}{b_{i_1}(\mathbf{x}_1)\ldots b_{i_t}(\mathbf{x}_t)P(s_1=i_1,\ldots,s_t=i_{t+1}|\Phi)} \right] \\
&= \arg\max_{i_{t+1}} \left[\max_{i_{t+2},\cdots,i_R} \left[\frac{\overrightarrow{a}_{0i_1i_2\cdots i_{t+1}} \overleftarrow{a}_{0i_1\cdots i_{t+1}i_{t+2}\cdots i_R} \overleftarrow{\epsilon}_1(i_1,i_2,\ldots,i_{t+1},i_{t+2},\ldots,i_R)}{b_{i_1}(\mathbf{x}_1)\ldots b_{i_t}(\mathbf{x}_t)P(s_1=0,s_1=i_1,\ldots,s_{t+1}=i_{t+1}|\Phi)} \right] \right] \\
&= \arg\max_{i_{t+1}} \left[\max_{i_{t+2},\cdots,i_R} \left[\frac{\overleftarrow{a}_{0i_1\cdots i_{t+1}i_{t+2}\cdots i_R} \overleftarrow{\epsilon}_1(i_1,\ldots,i_{t+1},i_{t+2},\ldots,i_R)}{b_{i_1}(\mathbf{x}_1)\ldots b_{i_t}(\mathbf{x}_t)P(s_1=0,s_1=i_1,\ldots,s_t=i_t|\Phi)} \right] \right] \\
&= \arg\max_{i_{t+1}} \left[\max_{i_{t+2},\cdots,i_R} \left[\overleftarrow{a}_{0i_1\cdots i_{t+1}i_{t+2}\cdots i_R} \overleftarrow{\epsilon}_1(i_1,\ldots,i_{t+1},i_{t+2},\ldots,i_R) \right] \right] \qquad (4.36)
\end{aligned}
$$

If we more close examine the result of Eq. 4.36, we note that an maximisation occurs with regards to various state index variable i.e.

$$
\max_{i_{t+2},\cdots,i_R} \left[\overleftarrow{a}_{0i_1\cdots i_{t+1}i_{t+2}\cdots i_R} \overleftarrow{\epsilon}_1(i_1,\ldots,i_{t+1},i_{t+2},\ldots,i_R) \right] \qquad (4.37)
$$

This maximisation bears a striking similarity to the maximisation that occurs as part of the Viterbi algorithm. In the next chapter we will show that this maximisation can be efficiently calculated using the Viterbi algorithm. In summary, the left-context "forward" pointers can be written in terms of the right-context "forward" pointers as follows:

$$
\begin{aligned}
&\text{For } t = T-1,\ldots,R: \\
&\qquad \overrightarrow{\Psi}^{\epsilon}_{t}(i_1,\ldots,i_R) = \overleftarrow{\Psi}^{\epsilon}_{t}(i_1,\ldots,i_R) \\
&\text{For } t = R-1,\ldots,1: \qquad (4.38) \\
&\qquad \overrightarrow{\Psi}^{\epsilon}_{t}(i_1,\ldots,i_t) = \arg\max_{i_{t+1}} \left[\max_{i_{t+2},\cdots,i_R} \left[\overleftarrow{a}_{0i_1\cdots i_{t+1}\cdots i_R} \overleftarrow{\epsilon}_1(i_1,\ldots,i_{t+1},\ldots,i_R) \right] \right]
\end{aligned}
$$

Since the left-context forward pointers are equal to the right context forward pointers (except for $t < R$), we can use the computationally efficient backward Viterbi-beam decoder on the right-context HMM to calculate $\overleftarrow{\Psi}^{\epsilon}_{T}(i_1,\ldots,i_R)$ and them efficiently calculate the left-context, best-path backward probability $\overrightarrow{\epsilon}_t(i_1,\ldots,i_R)$ recursively. This

alternative method of calculating the left-context, best-path backward probability is used in both the FBS-Viterbi-beam decoder and the A* search decoder when calculating the respective heuristic functions.

4.5.2 Computing the Viterbi-beam heuristic with low-order, right-context HMMs

In section 3.1.3.1 we discussed how an $(R-K)$-order, left-context HMM can be built for use with the low-order guided Viterbi-beam decoder. The heuristic function was calculated by backward Viterbi-beam decoding the $(R-K)$-order, left-context HMM $\Phi_{\mathrm{R-K,lc}}$. Since, it is computationally expensive to backward decode $\Phi_{\mathrm{R-K,lc}}$, we propose deriving an equivalent $(R-K)$-order, right-context HMM $\Phi_{\mathrm{R-K,rc}}$. By backward decoding $\Phi_{\mathrm{R-K,lc}}$, we should be able to more efficiently calculate the heuristic function used to guide the state pruning of the low-order guided Viterbi-beam decoder.

The pdfs of the $\Phi_{\mathrm{R-K,rc}}$ are identical to the pdfs of the R^{th}-order, left-context HMM $\Phi_{\mathrm{R,lc}}$. Furthermore, the right-context state transition probabilities are derived from the same joint state probabilities that was used to derive the left-context state transition probabilities of the $(R-K)$-order, left-context HMM.

$$\begin{aligned}
\overleftarrow{a}_{i_{K+1}i_{K+2}\cdots i_{R+1}} &= P(s_t = i_{K+1} | s_{t+1} = i_{K+2}, \ldots, s_{t-K+R} = i_{R+1}) \\
&= \frac{P(s_t = i_{K+1}, s_{t+1} = i_{K+2}, \ldots, s_{t-K+R} = i_{R+1})}{P(s_{t+1} = i_{K+2}, \ldots, s_{t-K+R} = i_{R+1})}
\end{aligned}$$

with

$$\begin{aligned}
& P(s_t = i_{K+1}, s_{t+1} = i_{K+2}, \ldots, s_{t-K+R} = i_{R+1}) \\
= & \sum_{i_1, i_2, \cdots, i_K} P(s_t = i_{K+1}, \ldots, s_{t-K+R} = i_{R+1}),\ i_1, i_2, \ldots, i_K \in \{0, \cdots, N+1\} \quad (4.39)
\end{aligned}$$

4.5.3 Computing the A* heuristic with right-context, pseudo HMMs

The derivation of the $(R-K)$-order, right-context pseudo HMM $\hat{\Phi}_{\mathrm{R-K,rc}}$ is not as simple as that of the $(R-K)$-order, right-context HMM $\Phi_{\mathrm{R-K,rc}}$ (derived in the previous section). This is because the parameters of the HMM $\hat{\Phi}_{\mathrm{R-K,rc}}$ must be chosen so that the left-context, best-path "forward" pointers are equal to the right-context, best-path "forward" pointers at all time indexes t and for all states i.e.

$$\begin{aligned}
\hat{\overrightarrow{\Psi}}^{\epsilon}_t(i_{K+1}, i_{K+2}, \ldots, i_R) &= \hat{\overleftarrow{\Psi}}^{\epsilon}_t(i_{K+1}, i_{K+2}, \ldots, i_R), \text{ where } t = 1, 2, \ldots, T \\
&\quad \text{and } i_{K+1}, i_{K+2}, \ldots, i_R \in \{0, 1, \ldots, N+1\} \quad (4.40)
\end{aligned}$$

If we ensure that the HMM $\hat{\Phi}_{\mathrm{R-K,rc}}$ uses the same set of pdfs as the R^{th}-order, left context HMM $\Phi_{\mathrm{R,lc}}$, then only the state transition probabilities can cause the forward pointers to differ.

The $(R-K)$-order, left-context pseudo transition probability is obtained by ignoring the identity of the first K states on which the R^{th}-order, left-context transition probability is dependent. The *value* of $(R-K)$-order transition probability $\hat{\overrightarrow{a}}_{i_{K+1}\cdots i_{R+1}}$ is associated with a single R^{th}-order transition probability i.e.

$$\begin{aligned}\hat{\overrightarrow{a}}_{i_{K+1}\cdots i_{R+1}} &= \overrightarrow{a}_{i_1^*\cdots i_K^* i_{K+1}\cdots i_{R+1}} \\ &= \frac{P(s_{t-R+1}=i_1^*,\cdots,s_{t-R+K-1}=i_K^*,s_{t-R+K}=i_{K+1},\cdots,s_t=i_{R+1})}{P(s_{t-R+1}=i_1^*,\cdots,s_{t-R+K-1}=i_K^*,s_{t-R+K}=i_{K+1},\cdots,s_{t-1}=i_R)}\end{aligned}$$

where

$$\{i_1^*,\ldots,i_K^*\} = \arg\max_{i_1^*,\ldots,i_K^*}\left[\overrightarrow{a}_{i_1\cdots i_K i_{K+1}\cdots i_{R+1}}\right] \tag{4.41}$$

In order to ensure that the right-context forward pointers are equal to the left-context forward pointers, we want the $(R-K)$-order, right-context transition probabilities to be associated with the *same* R^{th}-order state sequence.

If the parameters of the right-context, pseudo-HMM $\hat{\Phi}_{\mathrm{R-K,rc}}$ are derived from the left-context HMM $\Phi_{\mathrm{R,lc}}$ in the following manner:

$$\hat{\Phi}_{\mathrm{R-K,rc}} \triangleq \begin{cases} \hat{\overleftarrow{a}}_{i_k\cdots i_R(N+1)} = \overleftarrow{a}_{i_k\cdots i_R(N+1)}, & k=K+1,\ldots,R \\ \hat{\overrightarrow{a}}_{i_{K+1}i_{K+2}\cdots i_{R+1}} = \overrightarrow{a}_{i_1^*\cdots i_K^* i_{K+1}\cdots i_{R+1}} \frac{P(s_{t-R+K}=i_{K+1},\ldots,s_{t-1}=i_R)}{P(s_{t-R+K+1}=i_{K+2},\ldots,s_t=i_{R+1})} & \\ \hat{b}_{i_{R+1}}(\mathbf{x}_t) = b_{i_{R+1}}(\mathbf{x}_t) & \end{cases} \tag{4.42}$$

where $i_2,\ldots,i_{R+1} \in \{0,1,\ldots,N+1\}$

which will result in the "forward" pointers being equal. This is proven in Appendix B.2.

The heuristic function for the A* search decoder is therefore calculated by first backward Viterbi-beam decoding a $(R-K)$-order, right-context pseudo HMM $\hat{\Phi}_{\mathrm{R-K,rc}}$ to obtain the "forward" pointers $\hat{\overleftarrow{\Psi}}^{\epsilon}_t(i_{K+1},i_{K+2},\ldots,i_R)$. These "forward" pointers are then used to calculate the $(R-K)$-order, left-context, best-path forward probability $\hat{\overleftarrow{\epsilon}}'_t(i_{K+1},i_{K+2},\ldots,i_R)$, which is the heuristic function required by the A* search decoder $h\left[n_t(i_1,i_2,i_3,\ldots,i_R)\right] = \hat{\overleftarrow{\epsilon}}'_t(i_{K+1},i_{K+2},\ldots,i_R)$ (as described in Section 4.5.1).

4.6 Summary

This chapter started with the introduction of the right-context HMM. The new right-context HMM is our proposed solution to the problem of computational discrepancy between forward and backward Viterbi-beam decoding of high-order, left-context HMMs.

After adapting the decoding algorithms to also include right-context HMMs, we demonstrated that the mathematically equivalent right-context HMM can be backward Viterbi-beam decoded as computationally efficiently as the left-context HMM can be forward Viterbi-beam decoded. This supports our belief that backward decoding of HMMs is not fundamentally more expensive than forward decoding of HMMs. This also leads us to conclude that the conventional left-context HMM is only one way of viewing the underlying stochastic process. The newly derived right-context HMM is simply another way of viewing the same underlying process. The left- and right-context HMMs differ in that they are suited to different decoding approaches.

Since we are primarily interested in using information obtained from decoding low-order HMMs to guide the decoding of high-order HMMs, we spent the rest of the chapter discussing how the right-context HMM can be used to calculate the heuristic functions that are used by the proposed decoders (as discussed in the previous chapter). We mentioned that we cannot use the right-context, best-path backward probabilities directly. It would also be prohibitively expensive to convert the already calculated right-context, best-path backward probabilities to left-context, backward probabilities. We therefore showed that it is possible to efficiently calculate the left-context, backward probabilities by using the right-context, "forward" pointers, which are calculated as part of the backward decoding process. Lastly we discussed how $(R - K)$-order, right-context HMMs can be derived from the R^{th}-order HMM, so that the heuristic function can be calculated.

In the next chapter we will discuss some of the practical issues that need to be addressed when implementing the low-order guided decoders.

Chapter 5

Implementation Issues

In this section we will discuss some of the practical issues that arise when implementing the newly proposed decoders. The first two issues are applicable to both decoders, while the implementation of the A* decoder caused some issues unique to asynchronous decoders.

The first issue involves the conversion of the right-context, best-path backward probabilities into left-context, best-path backward probabilities. Specifically the issue revolves around obtaining a one-to-one mapping of states between the equivalent left- and right-context HMMs. The second issue involves a method of avoiding redundant observation likelihood calculations, and arose because the derived HMMs share the same set of pdfs as the high-order HMM.

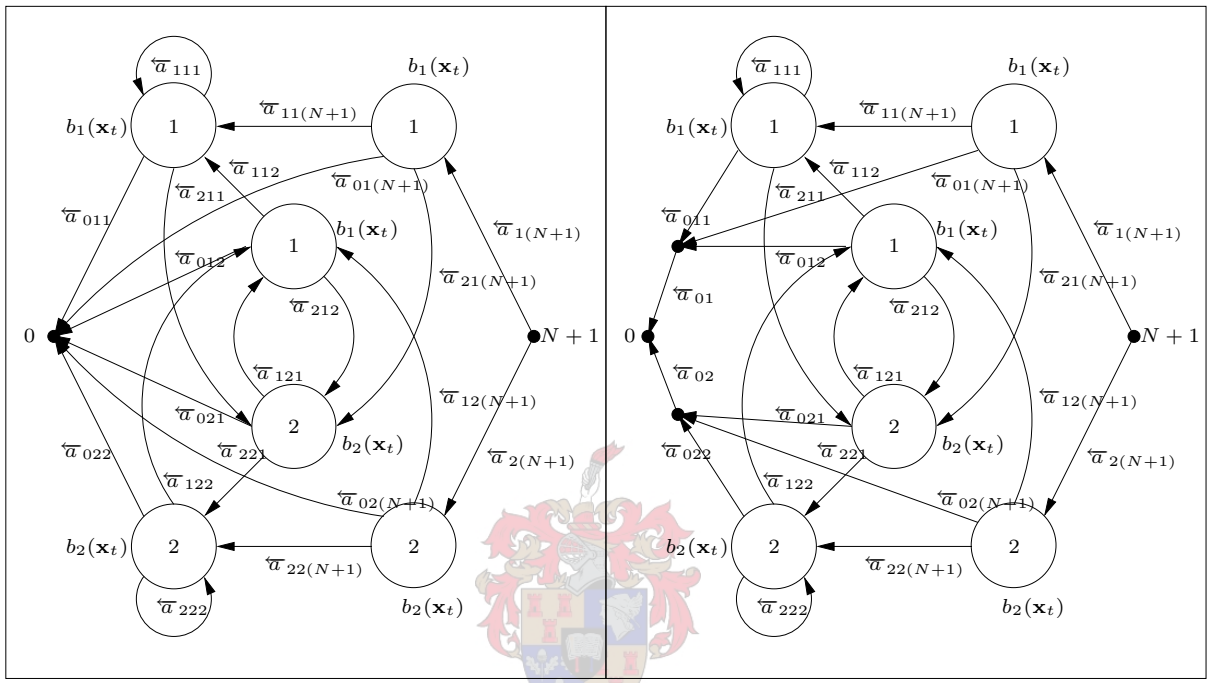
The A* decoder requires that additional bookkeeping overhead be performed to sort the OPEN list and to determine which nodes in the search graph have already been expanded. The first of the A* decoder implementation issues involves the choice of the data structures used to represent the intermediate calculations, so that the overhead can be kept to a minimum. Since the A* decoder is an asynchronous decoder it must store information regarding the states occurring at any time instance. The second of the A* implementation issues involves the efficient management of the memory space that is occupied by nodes constructed during the search process.

5.1 Best-path backward probability conversion

In Section 4.5.1 we derived the left-context "forward" pointers in terms of the right-context, best-path backward probabilities as follows:

$$\begin{aligned}
&\text{For } t = T-1, \ldots, R: \\
&\qquad \overrightarrow{\Psi}^{\epsilon}_{t}(i_1, \ldots, i_R) = \overleftarrow{\Psi}^{\epsilon}_{t}(i_1, \ldots, i_R) \\
&\text{For } t = R-1, \ldots, 1: \\
&\qquad \overrightarrow{\Psi}^{\epsilon}_{t}(i_1, \ldots, i_t) = \arg\max_{i_{t+1}} \left[\max_{i_{t+2}, \cdots, i_R} \left[\overleftarrow{a}_{0 i_1 \cdots i_{t+1} \cdots i_R} \overleftarrow{\epsilon}_{1}(i_1, \ldots, i_{t+1}, \ldots, i_R) \right] \right]
\end{aligned}$$

Figure 5.1: *(a) The first-order equivalent HMM of a two emitting-state, second-order, right-context HMM. (b) The first-order equivalent HMM of a two emitting-state, second-order, right-context HMM, with the extra null states added to the beginning of the HMM.*



Therefore, for $t \geq R$ we have a one-to-one mapping between states in the left-context HMM and states in the equivalent right-context HMM. However, for $t < R$, we have a many-to-one mapping of states, since we have to perform a maximisation over the state sequence $\mathbf{S}_{t+2}^{R} = \{s_{t+2} = i_{t+2}, \ldots, s_R = i_R\}$. For efficient implementation, it would be preferable to have a one-to-one mapping between the left- and right-context HMMs at every time index.

The problem is that a state sequence, which starts at time $t = 1$ and is shorter than R i.e. $\mathbf{S}_1^t = \{s_1 = i_1, \ldots, s_t = i_t\}$, corresponds to a unique state in the left-context, first-order equivalent HMM, but does not correspond to a unique state in the right-context, first-order equivalent HMM. However, the maximisation over the state sequence $\mathbf{S}_{t+2}^{R}$ can be efficiently performed using the Viterbi algorithm. Therefore, the solution is to add extra null states to the beginning of the right-context, first-order equivalent HMM, in such a way that such a state sequence *does* correspond to a unique state. This concept is illustrated in Fig. 5.1. We define the "extra" best-path backward probabilities of the right-context HMM to be

$$\overleftarrow{\epsilon}_1(i_1, i_2, \ldots, i_t) = \max_{i_{t+1}} \left[\overleftarrow{a}_{0 i_1 \cdots i_{t+1}} \overleftarrow{\epsilon}_1(i_1, i_2, \ldots, i_{t+1}) \right] \tag{5.1}$$

with $\overleftarrow{a}_{0i_1\cdots i_t} = 1$ for $t = 1, 2, \ldots, R-1$. Note that this operation if similar to the normal Viterbi maximisation step. The extra "forward" pointers can now be determined as

$$\begin{aligned}
\overleftarrow{\Psi}^{\epsilon}_{1}(i_1, i_2, \ldots, i_t) &= \arg\max_{i_{t+1}} \left[\overleftarrow{a}_{0i_1\cdots i_{t+1}} \overleftarrow{\epsilon}_{1}(i_1, i_2, \ldots, i_{t+1}) \right] \\
&= \arg\max_{i_{t+1}} \left[\max_{i_{t+2},\cdots,i_R} \left[\overleftarrow{a}_{0i_1\cdots i_{t+1}\cdots i_R} \overleftarrow{\epsilon}_{1}(i_1, \ldots, i_{t+1}, \ldots, i_R) \right] \right] \\
&= \overrightarrow{\Psi}^{\epsilon}_{1}(i_1, i_2, \ldots, i_t) \qquad (5.2)
\end{aligned}$$

Therefore, if the expanded right-context HMM is backward Viterbi-beam decoded, the left-context "forward" pointers for $t < R$ are calculated as part of the decoding process. Therefore, the addition of extra null states to the right-context HMM allows the backward Viterbi-beam algorithm to calculate and store all the "forward" pointers necessary for calculating the heuristic function. During backward Viterbi-beam decoding the extra null states will only be used at time $t = 1$, and therefore will not significantly influence the computational expense of the decoder.

5.2 Observation density likelihood cache

High-order HMMs and their derived, low-order HMMs share the same set of pdfs. Since the pdf likelihoods are all calculated when the heuristic function is determined (by backward Viterbi-beam decoding of the low-order HMM), we store the pdf likelihoods in a cache, so that the subsequent search algorithm (A* or forward Viterbi-beam on the high-order HMM) does not have to recalculate the same pdf likelihoods. The cache is implemented as a matrix of floating point values and has a size of $N \times T$, where N is the number of first-order emitting states and T is the length of the observation sequence.

5.3 A* Implementation

In this section we discuss some of the issues that need to be addressed when implementing an A* search algorithm. The A* search does not create the complete search graph, but constructs only the relevant section of the graph as the search progresses. This is done by starting at the root node and creating new nodes as the search progresses. Unlike the time-synchronous Viterbi-beam decoder, which only has to keep track of the search graph at time indexes t and $t-1$, the A* search needs to keep track of the whole search graph that has been constructed thus far. In order to do this efficiently the data structures used to implement the A* search need to be chosen carefully.

5.3.1 Data Structures

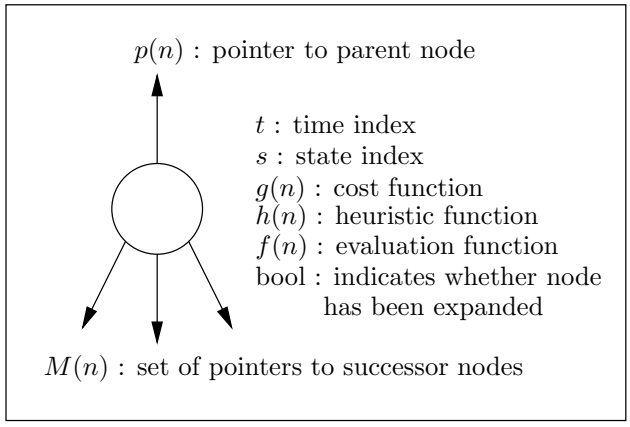
We have decided to implement each node as a single object in memory. The node object has the following attributes:

- t the time index of the node n.
- s the state index of the node n.
- The cost function $g(n)$, which is the accumulated probability of the best partial path from the root node n_s to the node n.
- The heuristic function $h(n)$, which is the estimated probability of the remaining best path from the node n to the goal node n_g.
- The evaluation function $f(n) = g(n)h(n)$, which is the estimated total probability of the best path going through node n.
- $p(n)$ A pointer to the parent node of the node n.
- $M(n)$ The set of pointers to the successor nodes of the node n.
- A boolean variable indicating if the node n has already been expanded.

which is also illustrated in Fig. 5.2.

In our implementation of A* search we use an OPEN and ALL list. The OPEN list is the list of all nodes that can be expanded. The ALL list is a list of all the nodes that already exists in the search graph (both expanded and unexpanded). The use of an ALL list does not differ from the discussion of the A* algorithm in Section 3.2.1, since the ALL

Figure 5.2: *The parameters of an A* node object.*



list is simply the union of the OPEN and CLOSED list. We have chosen to use an ALL list since it simplifies determining whether a node has been constructed.

The OPEN list must be sorted according to the evaluation function of the nodes on the list. We also want to quickly remove the node with the highest evaluation function from the OPEN list. Therefore, we have chosen to use a binary red-black tree to represent the OPEN list. The binary red-black tree is sorted during the insertion of new entries onto the list. Insertion into the list is of the order $\log_2(d)$, where d is the size of the list [44]. Removing the node with the highest evaluation function is not dependent on the size of the list. The OPEN list does not contain the node objects themselves, but contains references (pointers) to the node objects. Therefore it becomes necessary for a node object to relay its state and time index information.

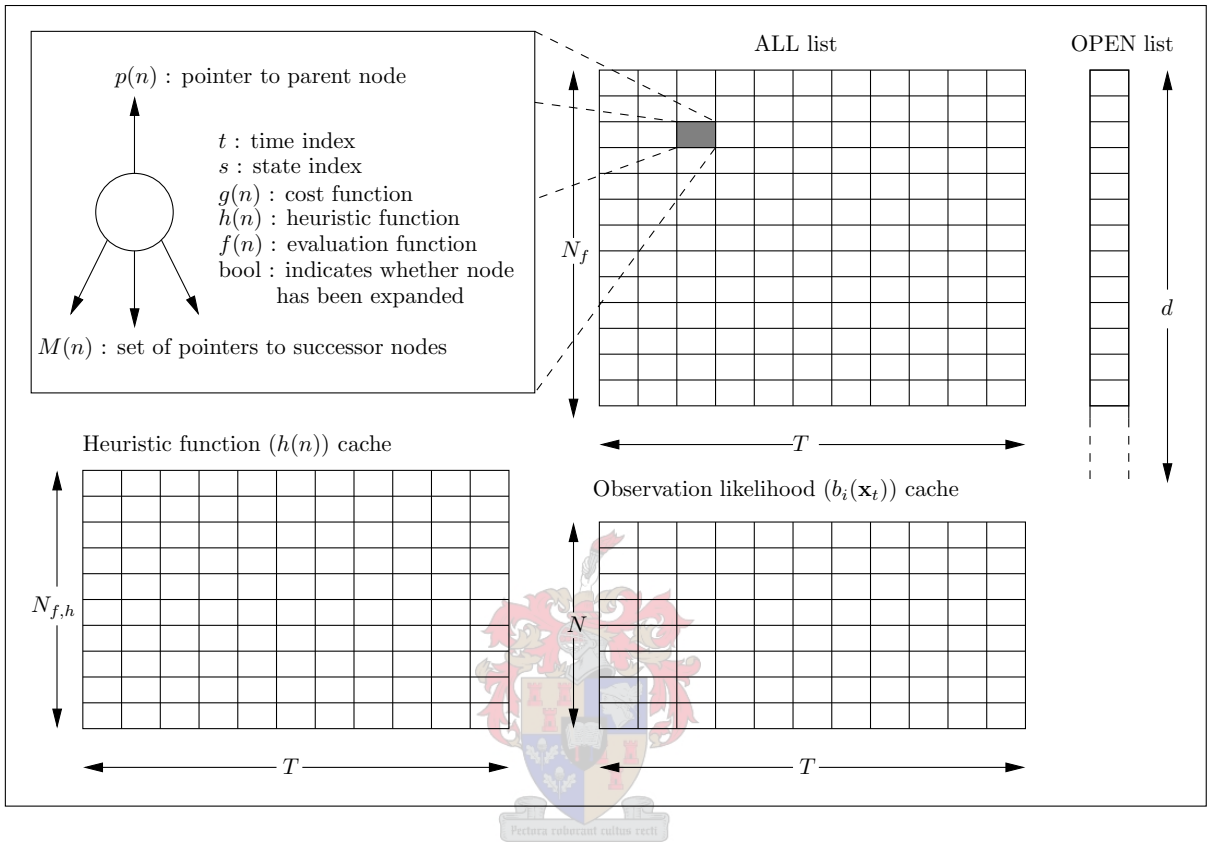
Since we wanted to quickly determine whether a node exists in the search graph, we have chosen a data structure for the ALL list for which access is not dependent on the size of the structure. The ALL list is therefore represented by a matrix of references (pointers) to node objects. The time and state index information is used as indices into the matrix representing the ALL list. If the pointer to the node object is NULL, it signifies that the particular node has not been created. The maximum size of the ALL list for an R^{th}-order HMM is $N^R T$, where N is the number of first-order emitting states and T is the length of the observation sequence. The data structures required for the A* search is illustrated in Fig. 5.3 where N_f is the number of states of the equivalent first-order HMM and $N_{f,h}$ is the number of states of the equivalent first-order HMM of the derived, low-order HMM.

5.3.2 Memory Management

Searching the search graph using the A* search can result in the creation of a large number of node objects. In the previous section we mentioned that the maximum size of the ALL list, and therefore the maximum number of node objects, is equal to $N^R T$. If we are decoding a $T = 2000$ length observation sequence using fifth-order HMM with $N = 10$ first-order emitting states, this can result in the creation of a total of 200 million node objects. Creating and destroying such a large number of node objects during the decoding of each segment will result in a significant portion of the decoding time being spent on managing the memory space the node objects occupy.

In order to reduce the time spent on managing memory, we have decided to create a large pool of node objects during the initialisation of the A* decoder. When the A* search process requires new nodes (as it does when a node is expanded), it simply retrieves a reference (pointer) to an unused node object in the pool, and initialises the parameters of the node object before using it. If there are no more unused node objects in the pool, the pool simply doubles its size by creating new unused node objects and allocating memory space for them. The pool only has to keep track of which nodes are used, thus when a new

Figure 5.3: *A graphic representation of the data structures used during the implementation of the A* search algorithm.*



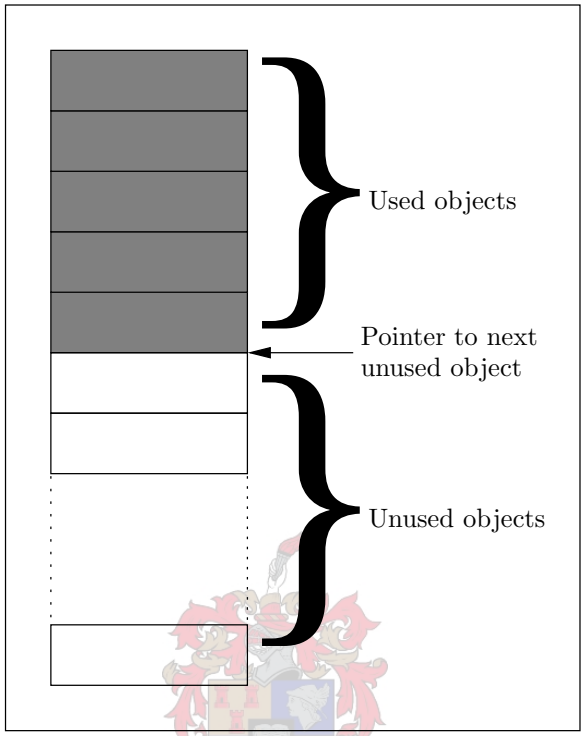
A* search is begun, the pool resets the number of used objects to zero. Thus, the use of a pool of node objects avoids the redundant allocation and deallocation of memory space and efficiently manages the memory. The pool of node objects is illustrated in Fig. 5.4.

5.4 Summary

In this chapter we discussed some of the practical issues that need to be addressed when implementing the proposed decoders. A one-to-one mapping of states in the left-context and equivalent right-context HMM was obtained by adding extra null states to the beginning of the right-context HMM. This allows *all* the left-context "forward" pointers, which are required when calculating the heuristic function, to be equivalent to the right-context "forward" pointers that are determined during the backward decoding of the derived, low-order, right-context HMMs. A pdf likelihood cache was introduced to the search algorithms to avoid recalculating pdfs that were already determined during the calculation of the heuristic function.

In order to minimise the bookkeeping overhead and memory management we discussed

Figure 5.4: *A graphic representation illustrating the pre-allocation of memory for the pool of A* node objects.*



the choice of data structures that were used for implementing the A* search. The memory was efficiently managed by introducing a pool of node objects, for which memory is allocated during the initialition of the A* search. This pool of node objects avoids the wasteful allocation and deallocation of memory during the decoding of multiple segments.

In the next chapter we will present the experimental results of decoding high-order HMMs using our newly developed decoders.

Chapter 6

Experimental investigation

The purpose of this chapter is to present the experimental evaluation results of the proposed FBS-based decoders. The intention is not to build an accurate, practical pattern recognition system, or even to show the benefit of using high-order HMMs. Therefore, the focus is not on the modelling uses of high-order HMMs, but rather on the fact that *using* high-order HMMs are not as computationally expensive as commonly accepted. When decoding any order HMM, there exists a path through the HMM which is optimal, i.e. most likely to have generated the observations. We are not concerned with the accuracy of the optimal path when compared to the correct reference, or even if the optimal path is representative of the observations. The optimal path is the best result that the particular HMM can deliver, and we are simply interested in *finding* that best path with the minimum effort.

We first define the experimental setup that is used to evaluate the computational efficiency of the proposed FBS-based decoders.

6.1 Experimental setup

We use the same experimental setup for all the experiments presented in this chapter. All our experiments have been performed on a single language pair of the CallFriend speech corpus. We have also limited our evaluation of high-order HMMs to topologies that are initialised in the first-order as a fully-connected topology.

6.1.1 Corpus

We evaluate the computational expense of decoding high-order HMMs using the CallFriend [8] speech corpus.

High-order HMMs were trained with the 40 recordings which form the German training set of CallFriend. These HMMs were then evaluated on the 40 recordings which form the English development set of CallFriend. Due to memory constraints whole recordings were

not used during training and decoding. The recordings used for training were divided into 60s segments. The recordings used for evaluation were divided into 20s segments for a total of 1410 segments that were decoded.

6.1.2 Observation Feature extraction

We use a variety of signal processing and feature extraction techniques that have become fairly standard in speech processing literature [9, 16, 18, 40]. Each recording of speech is blocked into $20ms$ frames and subsequent frames overlap by $10ms$. Each frame is windowed with an equal length Hamming function in order to reduce spectral leakage at the ends of each frame.

Mel-frequence cepstral coefficients (MFCCs) are extracted by using a bank of 22 triangular filters equally spaced according to the *mel*-frequency scale. For each frame the energy of each filter is converted to 12^{th}-order MFCCs by using the discrete cosine transform. The features are post-processed with ceptral mean subtraction (CMS), augmented with velocity (Δ) and acceleration ($\Delta\Delta$) features, and reduced to 21-dimensional features using a linear discriminant analysis (LDA) based dimension reduction technique [10, 20].

6.1.3 Training of high-order HMMs

Instead of directly training R-th order HMMs on the data, we use the method of Fast Incremental Training (FIT) [12]. This has the advantage of also training all the HMMs with order lower than R. The state output pdfs of an N emitting-state, first-order HMM are initialised by first dividing the training data into N regions utilising vector quantisation. The means of the N regions are then used to initialise diagonal covariance Gaussian output density functions. The variances of the Gaussian pdfs are initialised to be 1.0. The transition probabilities are initialised to be equal. The FIT algorithm trains the R-th order HMM by starting with a fully-connected, first-order HMM and increasing the order incrementally. Between each increase in order the HMM parameters are trained using the Viterbi-re-estimation algorithm. State transition probabilities which drop below a threshold of 10^{-5} are removed from the HMM. As the order of the HMMs increases when applying the method of FIT, transitions with low probabilities will be removed from the high-order HMMs.

The Fast Incremental Training (FIT) algorithm can be summarised as follows [11]:

1. Set up a first-order HMM for the application at hand.

2. Run the training algorithm on the first-order model. Non-viable transitions will disappear.

3. Convert the optimised first-order model to a second-order model by expanding the subscripts of the remaining non-zero transition probabilities with one extra prior state. These expanded transition probabilities are initialised with the value of the lower-order transition probability they were extended from.

4. Use the Order REDucing (ORED) algorithm to create a first-order equivalent of this model.

5. Now, by repeating the algorithm from step 2, train this model. This will refine the transition probabilities to their required higher-order values. Repeating this process trains successively higher-order models.

6.1.4 Measuring computational expense of a decoder

We repeat the discussion of the method used for measuring decoder performance first discussed in chapter 2. We use the total number of transition probabilities evaluated during decoding as an implementation independent measure of the computational expense of the decoder. In the case of the FBS-based decoders, the total decoding cost C_{tot} (i.e. the total number of transitions evaluated) consists of three separate costs, namely the heuristic cost C_h, the heuristic conversion cost C_c and the search cost C_s. The heuristic cost C_h is the number of transitions that are evaluated when calculating the right-context, best-path backward probability (as discussed in Section 4.3.2). The heuristic conversion cost C_c is the total number of transitions evaluated when converting the right-context, best-path backward probabilities into left-context, best-path backward probabilities (as discussed in Section 4.5). The search cost C_s is the number of transitions evaluated when performing the forward search of the graph by using either the forward Viterbi-MAP-beam algorithm (see Section 3.1) or the A* search algorithm (see Section 3.2). Thus the total decoding cost is equal to

$$C_{\text{tot}} = C_s + C_h + C_c \tag{6.1}$$

There is no heuristic cost or conversion cost when using the forward Viterbi-MAP-beam decoder, therefore the total decoding cost is equal to the search cost i.e. $C_{\text{tot}} = C_s$.

There are two main reasons why we want an implementation independent measure of the decoders. The first reason is that we want a method of measuring expense without having to resort to code optimisation techniques. If we only used decoding time (as measured by the processing time required for the decoding), our computational expense measure would be influenced by the amount of code optimisation that was performed on the decoders. The second reason is that the A* decoder requires some computational overhead for the bookkeeping associated with the OPEN and CLOSED lists. This overhead

is generally more expensive than the bookkeeping associated with the standard Viterbi-MAP-beam decoder (and the proposed FBS-Viterbi-beam decoder). Therefore, processing a single transition probability during the search part of the A* search is computationally more expensive than processing a single transition during the calculation of the heuristic function. However, the exact ratio is once again dependent on the implementation of the decoders. By only measuring the number of evaluated transition probabilities, we are effectively ignoring the computational expense of the bookkeeping overhead. This does give us an indication of the algorithmic behaviour of the decoders as the size and order of the HMMs are increased. Since the computational expense of the overhead will never completely disappear, we also measure the implementation dependent decoding time of the decoders.

Over the total number of segments decoded, we compute the average number of evaluated transition probabilities and the total decoding time. We compute the normalised total decoding cost $C_{\text{tot,n}}$ by normalising the number of evaluated transition probabilities C_{tot} with the maximum number of transition probabilities that the standard Viterbi decoder would evaluate. This maximum number of evaluated transition probabilities is equal to the total number of transition probabilities of the HMM, multiplied by the length of the feature vector T.

As the decoders are not guaranteed to find an optimal state sequence, the following three results occur when a segment is decoded:

(a) the decoder does not find any state sequence,

(b) the decoder finds a state sequence, but it is not the optimal state sequence, and

(c) the decoder finds the optimal state sequence.

The segment is regarded as being correctly decoded only when the decoding results in (c).

Result (a) occurs (the decoder does not find any state sequence) because the width of the beam used for pruning was too narrow, resulting in a too small search space in which no single state sequence survives the beam-pruning. A decoder can recover from this error by simply decoding the segment again, this time with a wider beam-width, so that a larger search space is examined. The number of transitions evaluated when the decoder fails to find a state sequence for a segment is added to the total number of transitions required to decode that particular segment.

The only practical method for determining whether a decoded state sequence is optimal, is by comparing it with a reference state sequence. It is therefore very difficult for the decoder to distinguish whether result (b) or (c) occurred. Thus the decoders used in this dissertation cannot recover from errors as a result of (b).

6.2 Expense of determining the heuristic function

Motivation

The heuristic function, used for state pruning in the FBS-Viterbi-beam decoder and used for ranking nodes in the A* decoder, is calculated by backward decoding the low-order right-context HMMs. We have shown in the previous chapters that high-order, right-context HMMs can be efficiently backward decoded. As the HMM derived for the FBS-Viterbi-beam decoder are true HMMs, they should exhibit the same decoding behaviour, with respect to the number of transitions evaluated by the decoder, as was reported in the previous chapter. However, the question remains whether the low-order, right-context pseudo HMMs can also be decoded efficiently. The purpose of the experiments in this section is to measure the computational expense of backward decoding the derived, low-order HMMs, since this will directly influence the heuristic cost C_h of the FBS-based decoders.

6.2.1 Decoding of derived low-order, right-context HMMs

Motivation

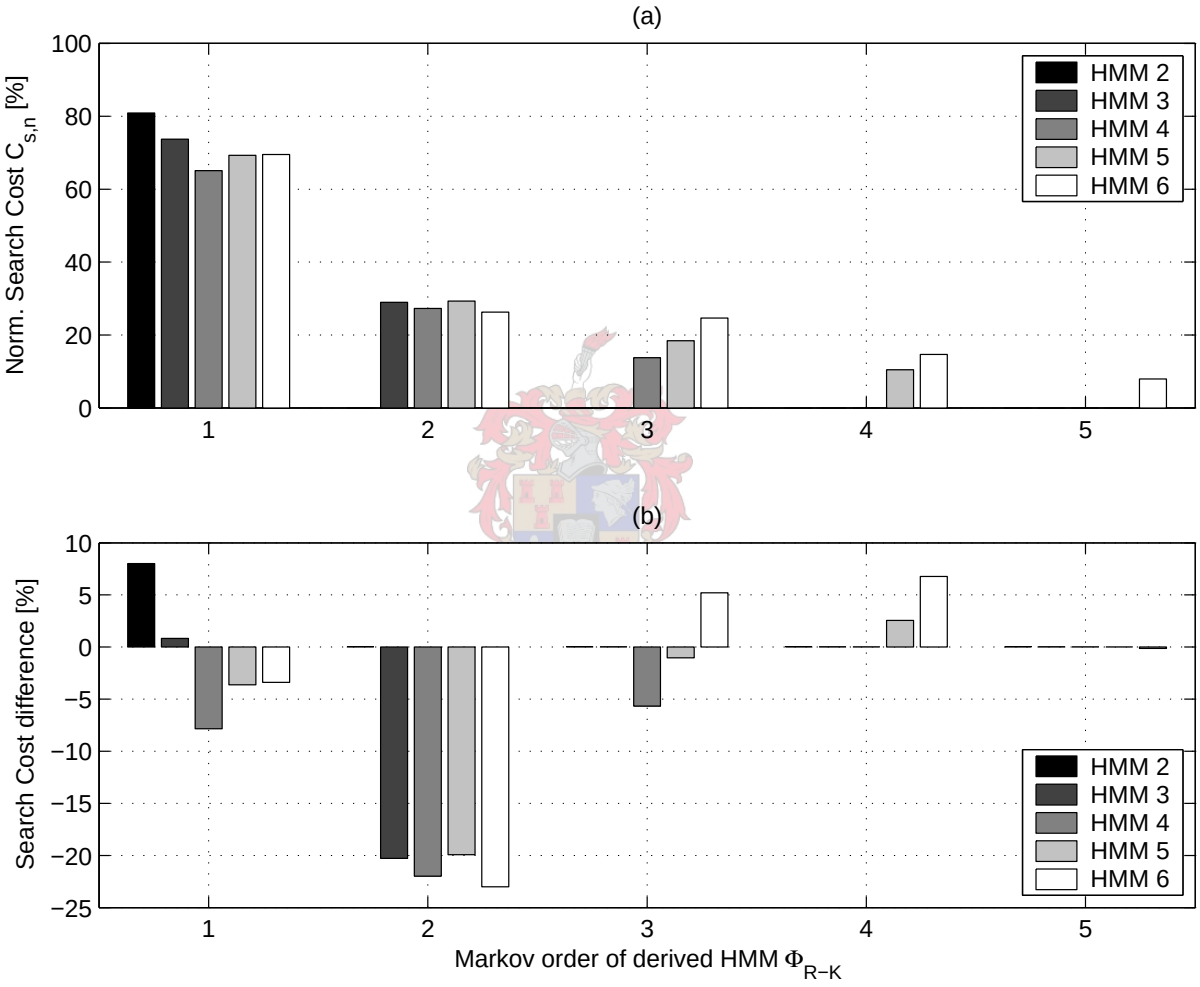
In Section 4.5.2 we discussed how $(R-K)$-order, right-context HMMs can be derived from R^{th}-order, left-context HMMs. We also showed that right-context HMMs can be backward decoded as efficiently as left-context HMMs can be forward decoded. The purpose of this experiment is to verify that the *derived* right-context HMMs can also be efficiently backward decoded, with respect to the number of transitions evaluated. Since the derived, low-order right-context HMMs are true HMMs, we expect the backward search cost to be similar to the cost of forward decoding equivalent order left-context HMMs. The backward search cost of the low-order, right-context HMMs will give us an indication of the heuristic cost of the FBS-Viterbi-beam decoder.

Experimental setup

Exactly the same experimental setup is used for decoding the low-order, right-context HMMs, as was used in Sections 2.2.4, 3.3.4 and 4.4. The $(R-K)$-order, right-context HMMs derived during the training of the high-order HMMs is backward decoded using the Viterbi-MAP-beam decoder. The parameters of the derived, low-order right-context HMMs were estimated concurrently with the parameters of the left-context HMMs using the FIT algorithm.

As before, we use the search cost C_s (the number of transitions evaluated during decoding) as an implementation independent measure of the computational expense of the

Figure 6.1: *(a) The normalised search cost of the Viterbi-MAP-beam decoder during the backward decoding of the derived low-order, right-context HMMs, with beams set wide enough to decode all segments correctly. (b) The difference in normalised search cost between the backward Viterbi-MAP beam decoding of the derived low-order, right-context HMMs and the forward Viterbi-MAP-beam decoding of the equivalent order left-context HMM, with beams set wide enough to decode all segments correctly.*



backward Viterbi-MAP-beam decoder. Over the 1410 segments decoded, we compute the average number of evaluated transition probabilities. We compute the normalised search cost by normalising the number of evaluated transition probabilities with the maximum number of transition probabilities that the standard Viterbi decoder would evaluate.

Results

Fig. 6.1(a) illustrates the search cost of backward decoding the derived HMMs. For reference, the computational expense of backward decoding the derived low-order right-

context HMMs are also tabulated in Table C.1. We can clearly see that the search cost decreases as the derived order of the right-context HMMs is increased, which is similar to the backward decoding behaviour of right-context HMMs illustrated in Fig. 4.3.

In Fig. 6.1(b) we show the difference in forward decoding the high-order, left-context HMMs and backward decoding the derived right-context HMMs of equivalent order. For example, the difference is calculated between backward decoding a third-order, left-context HMM and the derived third-order, right-context HMMs (which are derived from the fourth-, fifth- and sixth-order left-context HMMs).

Interpretation

The results verify our assumption that the backward decoding behaviour of the derived right-context HMMs is similar to the forward decoding behaviour of the high-order, left-context HMMs. We can see that the computational expense of backward decoding the derived low-order, right-context HMMs does not increase by more than 10% when compared to the search cost of forward decoding the high-order, left-context HMMs.

It is interesting to note that in some cases the backward search cost is actually less than the forward search cost. For the case of second-order HMMs, we see that the search cost for the derived right-context HMMs is more than 20% less than that of the left-context HMM. This can be explained by the fact that the second-order HMM required quite a large minimum beam-width of $B = 40$ in order to correctly decode all 1410 segments (see Table 2.2). For correctly decoding all 1410 segments with the derived right-context HMMs, the minimum required beam-widths are in the range of $25 \leq B \leq 28$, which would account for these large differences in decoding cost. For the fifth-order case, where the minimum required beam-width for decoding the left-context HMM is the same for decoding the derived right-context HMM, we see that there is very little difference in search cost. Therefore, if the beam-width is kept constant, we suspect that the difference in search cost between backward decoding the derived right-context HMMs and forward decoding the left-context HMMs will be negligible.

6.2.2 Decoding of derived low-order, right-context pseudo HMMs

Motivation

In Section 4.5.3 we discussed how $(R - K)$-order, right-context pseudo HMMs can be derived from R^{th}-order, left-context HMMs. In Appendix B.2 we prove that the derived, right-context pseudo HMM will result in the same "forward" pointers as the derived, left-context pseudo HMMs. However, this does not imply that the backward search cost will be similar to the forward search cost of left-context HMMs. Therefore, the purpose of

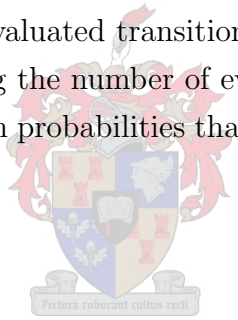
this experiment is to verify that the *derived* right-context HMMs can also be efficiently backward decoded, with respect to the number of transitions evaluated.

The backward search cost of the low-order, right-context pseudo HMMs will give us an indication of the heuristic cost of the A* decoder.

Experimental setup

The same experimental setup is used for decoding the low-order, right-context HMMs, as was used in Sections 2.2.4, 3.3.4 and 4.4, except that we only evaluate the decoding performance using the English development set. The parameters of the derived, low-order right-context HMMs were estimated concurrently with the parameters of the left-context HMMs using the FIT algorithm.

As before, we use the search cost (the number of transition probabilities evaluated during decoding) as an implementation independent measure of the computational expense of the backward Viterbi-MAP-beam decoder. Over the 1410 segments decoded we compute the average number of evaluated transition probabilities. We compute the normalised search cost by normalising the number of evaluated transition probabilities with the maximum number of transition probabilities that the standard Viterbi decoder would evaluate.



Results

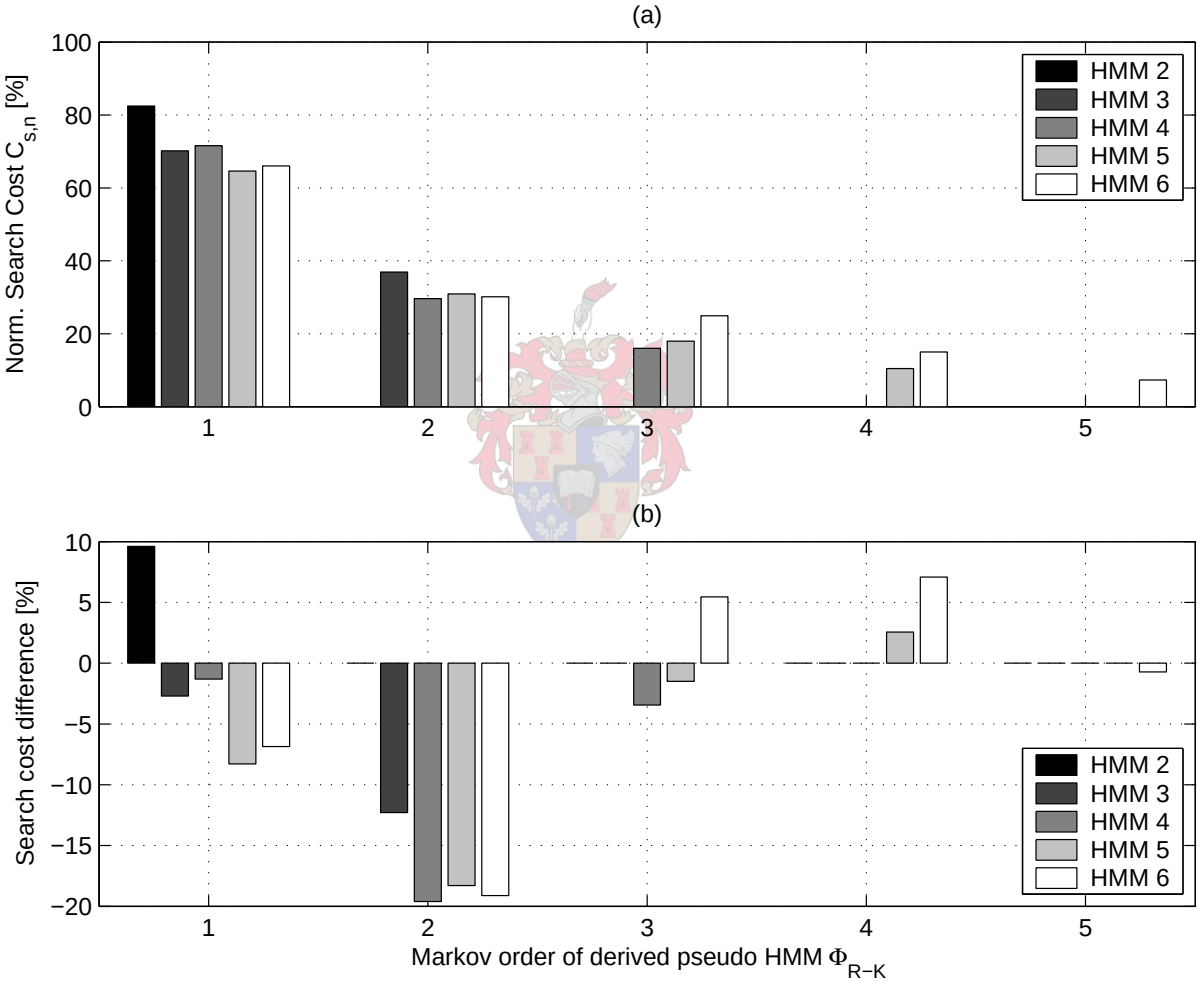
Fig. 6.2(a) illustrates the search cost of the derived HMMs. For reference, the minimum computational expense of backward decoding the derived, low-order, right-context pseudo HMMs are tabulated in Table C.2. We can clearly see that the search cost also decreases as the derived order of the right-context, pseudo HMMs is increased.

In Fig. 6.2(b) we show the difference in forward decoding the high-order, left-context HMMs and backward decoding the derived, right-context, pseudo HMMs of equivalent order. For example, the difference is calculated between backward decoding a third-order, left-context HMM and the derived third-order, right-context, pseudo HMMs (which are derived from the fourth-, fifth- and sixth-order, left-context HMMs).

Interpretation

The results again verify our assumption that the backward decoding behaviour of the derived right-context, pseudo HMMs are similar to the forward decoding behaviour of the high-order, left-context HMMs. We can see that the computational expense of backward decoding the derived low-order, right-context HMMs does not increase by more than 10% when compared to the search cost of forward decoding the high-order, left-context HMMs.

Figure 6.2: *(a) The normalised search cost of the Viterbi-MAP-beam decoder during the backward decoding of the derived low-order, right-context pseudo HMMs, with beams set wide enough to decode all segments correctly. (b) The difference in normalised search cost between the backward Viterbi-MAP beam decoding of the derived low-order, right-context pseudo HMMs and the forward Viterbi-MAP-beam decoding of the equivalent order left-context HMM, with beams set wide enough to decode all segments correctly.*



We again note the large difference in search cost for the case of second-order HMMs. We believe this is also caused by the fact the second-order HMM minimum required a beam-width of $B = 40$ (see Table 2.2), to correctly decode all the segments, while the minimum required beam-widths for the derived, right-context, pseudo HMMs are in the range of $25 \leq B \leq 32$. For the fifth-order case, where the minimum required beam-width of $B = 37$ (for decoding the left-context HMM) is the nearly same as the minimum required beam-width of $B = 36$ (for decoding the derived right-context HMM), we again

see that there is very little difference in search cost.

It is interesting to note that when the order of the derived, right-context pseudo HMMs is much lower than the order of the left-context HMM from which it was derived, then the minimum beam-width required for correctly decoding all segments is much smaller than for the derived, right-context HMM. For example, in the fifth-order case, the minimum required beam-width for the derived right-context, pseudo HMM is only $B = 19$, while the minimum required beam-width for the derived, right-context HMM is $B = 26$. However, the search cost for the right-context, pseudo HMM is 66.05% while the search cost for the right-context HMM is 69.52%, which is only a difference of 3.47%. Thus, decoding the pseudo HMM is nearly as expensive as decoding the normal HMM, although the minimum required beam-width is smaller. This can be explained by the fact the transition probabilities of the pseudo HMM do not sum to unity. This causes the probabilities of states at a specific time index, given the partial observation sequence, to be closer to each other than in the case of the true HMM, although the state probabilities are still ranked in the same order as for the true HMM.

6.2.3 Summary

We have measured the computational expense (with respect to the number of evaluated transitions) of both the derived, low-order HMM and the derived, low-order pseudo HMM. In both cases the computational expense of backward decoding the derived HMMs is similar to the computational expense of decoding the high-order, left- and right-context HMMs reported in the previous chapters. We can therefore safely use the derived, low-order HMMs to calculate heuristic functions for the FBS-based decoders, since we have shown that the derived HMMs can be decoded efficiently.

6.3 Forward-Backward Search of high-order HMMs

Motivation

The purpose of the following series of experiments is to determine the computational efficiency of using the newly proposed FBS-based decoders to decode high-order, left-context HMMs. Specifically, we want to determine whether the inclusion of information obtained from decoding low-order HMMs does reduce the computational expense of decoding high-order HMMs.

Experimental setup common to all experiments

The same experimental setup is used for decoding the left-context, high-order HMMs, as was used in Sections 2.2.4, 3.3.4 and 4.4, except that we only evaluate the decoding performance using the English development set. The parameters of the derived, low-order right-context HMMs were estimated concurrently with the parameters of the left-context HMMs using the FIT algorithm.

We define the total decoding cost C_{tot} to be the total number of transition probabilities evaluated during decoding. This include the transition probabilities evaluated as part of the calculation and conversion of the heuristic function. We use this total decoding cost as an implementation independent measure of the computational expense of the decoders. Over the 1410 segments decoded we compute the total decoding cost as the average number of evaluated transition probabilities. We compute the normalised total decoding cost by normalising the number of evaluated transition probabilities with the maximum number of transition probabilities that the standard Viterbi decoder would evaluate. Lastly, we also measure the implementation dependent total decoding time.

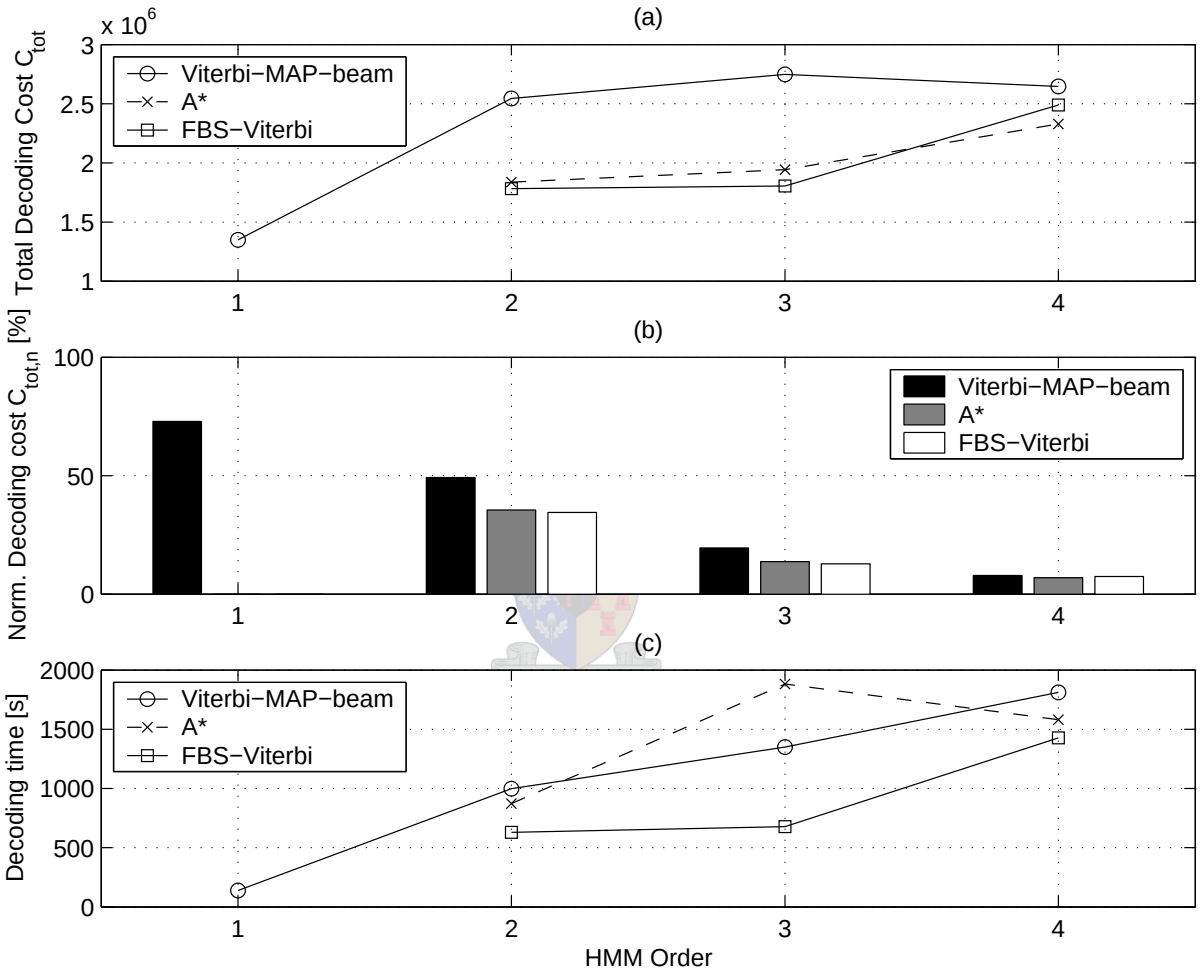
As it is not possible to compare the decoders based on the value of a common pruning beam-width B, we have decided to compare the decoders at the point where each beam is set wide enough so that all segments are correctly decoded. For the FBS-based decoders we also determine the best total decoding cost for each order of derived HMM i.e. $(R-K) = 1, 2, \ldots, R-1$.

6.3.1 FBS-based decoding vs. Viterbi-MAP-beam decoding

Motivation

In Section 2.2.4 we have determined the computational efficiency of decoding high-order HMMs using the base-line Viterbi-MAP-beam decoder. The purpose of this experiment is to compare the computational expense of the newly proposed decoders against the baseline expense of the Viterbi-MAP-beam decoder, with respect to the total decoding

Figure 6.3: *A comparison of (a) the total decoding cost (the total number of transitions evaluated), (b) the normalised total decoding cost, and (c) the total decoding time of the FBS-Viterbi-beam, A* and Viterbi-MAP-beam decoders, during the forward decoding of high-order left-context HMMs, with beams set wide enough to decode all segments correctly.*

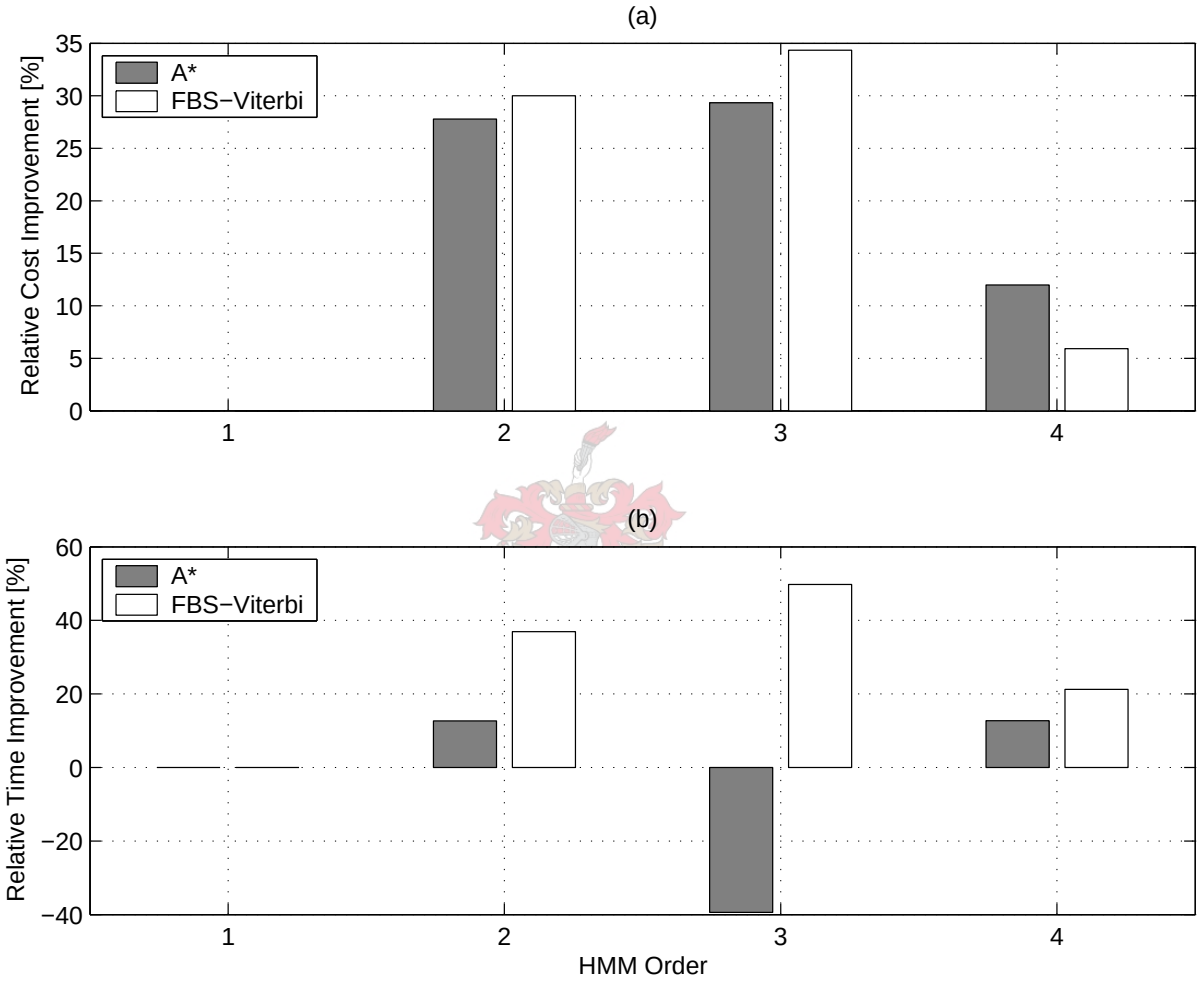


cost and the total decoding time.

Experimental setup

The initial, fully-connected, first-order HMM has a total of 40 emitting states and 1,680 state transition probabilities (925 transitions after training). The first-order equivalent HMM of the fifth-order HMM has a total of 15,335 emitting states (sharing 40 pdfs) and 35,927 state transition probabilities.

Figure 6.4: *The improvement in (a) the normalised total decoding cost, and (b) the total decoding time of the FBS-Viterbi-beam and A* decoders, relative to the Viterbi-MAP-beam decoder, during the forward decoding of high-order left-context HMMs, with beams set wide enough to decode all segments correctly.*



Results

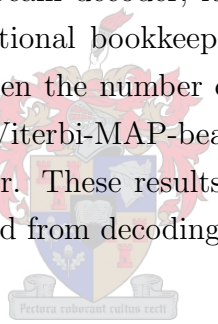
Fig. 6.3(a) illustrate the total decoding cost of each of the Viterbi-MAP-beam, A* and FBS-based Viterbi-beam decoders. Fig. 6.3(b) illustrates the normalised total decoding cost while Fig. 6.3(c) shows the implementation dependent total decoding time. For reference, the total computational expense of decoding the high-order HMMs are tabulated in Table C.3.

Fig. 6.4(a) and (b) respectively illustrates the improvement in the normalised total decoding cost and the total decoding time of the FBS-based decoders relative to the Viterbi-MAP-beam decoder.

Interpretation

Figs. 6.3(a) and (b) clearly indicate that, in order to correctly decode all segments, both the A* and the FBS-based decoder evaluate significantly less transitions than the Viterbi-MAP-beam decoder evaluates. When decoding the fourth-order HMM, the Viterbi-MAP-beam decoder has nearly the same computational expense as the FBS-based decoders, which is the case for which the Viterbi-MAP-beam decoder requires the smallest beam-width of $B = 29$ in order to correctly decode all segments.

Fig. 6.3(c) shows that the FBS-based decoders correctly decodes all the investigated HMMs faster than the Viterbi-MAP-beam decoder does. The FBS-Viterbi-beam decoder is in the worst case approximately 20% faster than the Viterbi-MAP-beam decoder and in the best case it is approximately 50% faster (it correctly decodes all the segments in roughly half the time the Viterbi-MAP-beam decoder decodes them). As we can see when the third-order HMM is decoded, although the A* search decoder always evaluates less transitions than the Viterbi-MAP-beam decoder, it does not always decode the HMMs faster. This is caused by the additional bookkeeping the A* decoder has to perform, which is not taken into account when the number of evaluated transitions is measured. It would also seem as if the FBS-Viterbi-MAP-beam decoder is computationally more efficient than the A* search decoder. These results clearly indicate the advantage that is gained when information, obtained from decoding low-order HMMS, is included in the search algorithm.



6.3.2 The influence of the HMM emitting state size N

Motivation

In section 6.3.1 we showed that the FBS-based decoders are computationally more efficient than the Viterbi-MAP-beam decoder, when decoding high-order HMMs. However, we have only investigated high-order HMMs with $N = 40$ first-order emitting states and DC-Gaussian state output pdfs. In this experiment we want to determine how sensitive the new FBS-based decoders are to the number of emitting states N of the first-order HMM, since this directly influences the size of the high-order HMMs, which are obtained by applying the method of FIT when training high-order HMMs.

Experimental setup

We investigate the decoding of high-order HMMs for $N = 10$, $N = 40$ and $N = 50$. The state output pdfs of an N emitting-state, first-order HMM are initialised by first dividing the training data into N regions utilising vector quantisation. The means of the N regions are then used to initialise diagonal covariance Gaussian output density functions. We have chosen the simplest pdfs for this experiment so that the observation likelihood calculations

Table 6.1: *The number of transitions and states of $N = 10$, $N = 40$ and $N = 50$ first-order emitting state, high-order, left-context HMMs. All HMMs use diagonal covariance state output pdfs.*

HMM order R	$N = 10$		$N = 40$		$N = 50$	
	# states	C_{tot}	**# states**	C_{tot}	**# states**	C_{tot}
1	12	99	42	925	52	1,277
2	78	247	678	2,583	934	3,615
3	193	531	2,060	7,060	2,882	9,904
4	417	999	6,112	16,715	8,700	23,786
5	816	1,742	15,335	35,927	22,246	52,153
6	1,551	3,072	34,009	72,437	-	-
7	2,781	5,159	-	-	-	-
8	4,744	8,409	-	-	-	-
9	7,901	13,445	-	-	-	-

do not dominate the decoding. The transition probabilities are initialised to be equal. The FIT algorithm trains the R-th order HMM by starting with a fully-connected, first-order HMM and increasing the order incrementally. Between each increase in order the HMM parameters are trained using the Viterbi-re-estimation algorithm. State transition probabilities which drop below a threshold of 10^{-5} are removed from the HMM.

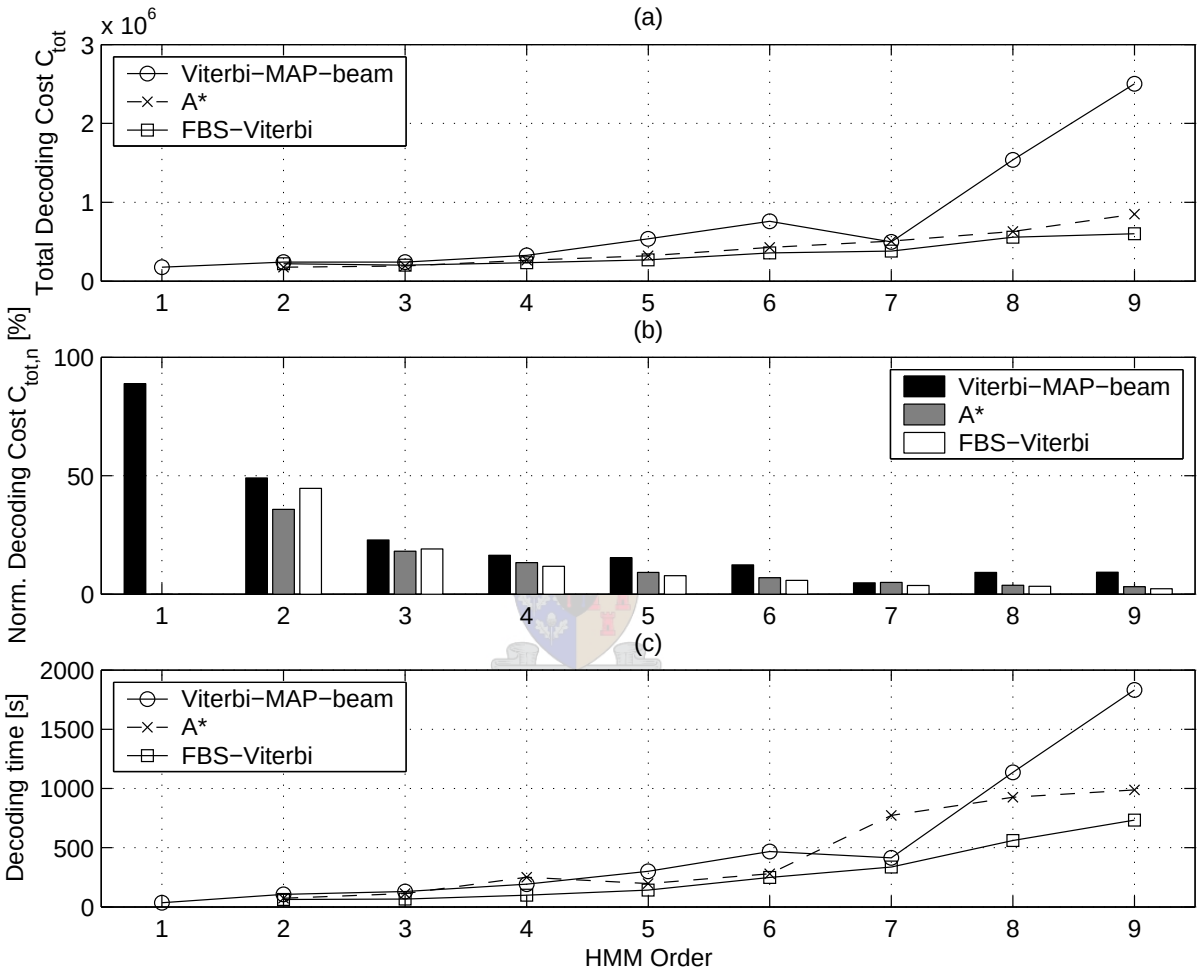
The sizes of the resultant high-order, left-context HMMs are shown in Table 6.1. Each of the high-order, left-context HMMs are decoded with respectively the standard Viterbi-MAP-beam decoder, the FBS-Viterbi-beam decoder and the A* search decoder. For the FBS-based decoders we also determine the minimum computational expense for each order of derived HMM i.e. $(R - K) = 1, 2, \ldots, R - 1$.

Results

Figs. 6.5(a), 6.6(a) and 6.7(a) respectively illustrate the total decoding cost of each of the Viterbi-MAP-beam, A* and FBS-based Viterbi-beam decoder, for $N = 10$, $N = 40$ and $N = 50$. Figs. 6.5(b), 6.6(b) and 6.7(b) illustrate the normalised total decoding cost while Figs. 6.5(c), 6.6(c) and 6.7(c) show the implementation dependent total decoding time. For reference, the total computational expense of decoding high-order HMMs with $N = 10$, $N = 40$ and $N = 50$ are also respectively tabulated in Tables C.4, C.5 and C.6.

Figs. 6.8(a), (b) and (c) respectively illustrate the improvement in total decoding cost of the FBS-based decoders relative to the Viterbi-MAP-beam decoder, for decoding $N = 10$, $N = 40$, and $N = 50$ high-order HMMs. Figs. 6.9(a), (b) and (c) respectively illustrate the improvement in total decoding cost of the FBS-based decoders relative to

Figure 6.5: *A comparison of (a) total decoding cost, (b) the normalised total decoding cost, and (c) the total decoding time of the FBS-Viterbi-beam, A* and Viterbi-MAP-beam decoders, during the forward decoding of high-order HMMs with $N = 10$ emitting-state and DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*

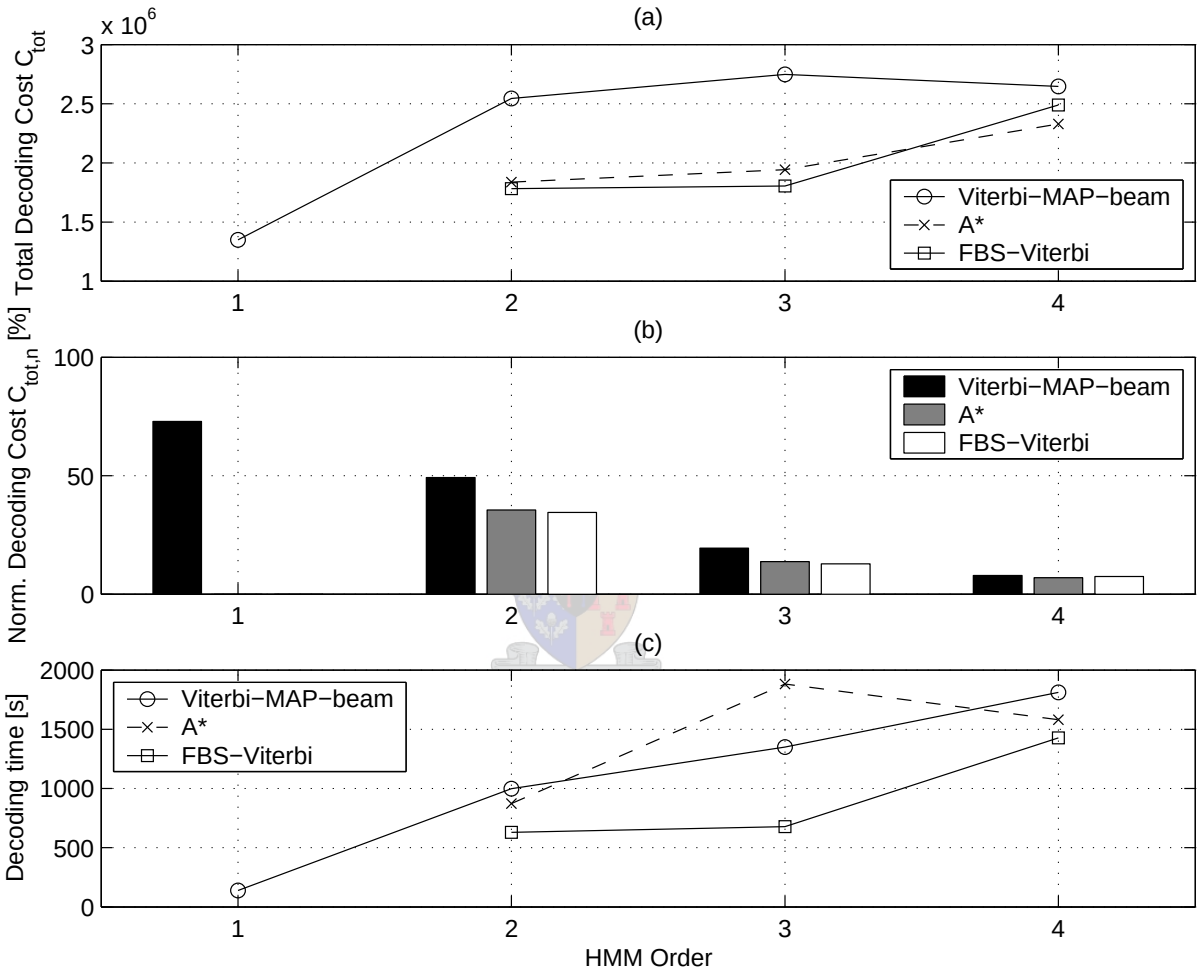


the Viterbi-MAP-beam decoder, for decoding $N = 10$, $N = 40$, and $N = 50$ high-order HMMs.

Interpretation

Figs. 6.5 and 6.6 indicate that for relatively small values of N and with fairly simple state output pdfs, the FBS-based decoders are computationally more efficient than the Viterbi-MAP-beam decoder. When decoding the seventh-order HMM, the Viterbi-MAP-beam decoder has nearly the same computational expense as the FBS-based decoders. However, this is the order for which the Viterbi-MAP-beam decoder is set to the smallest

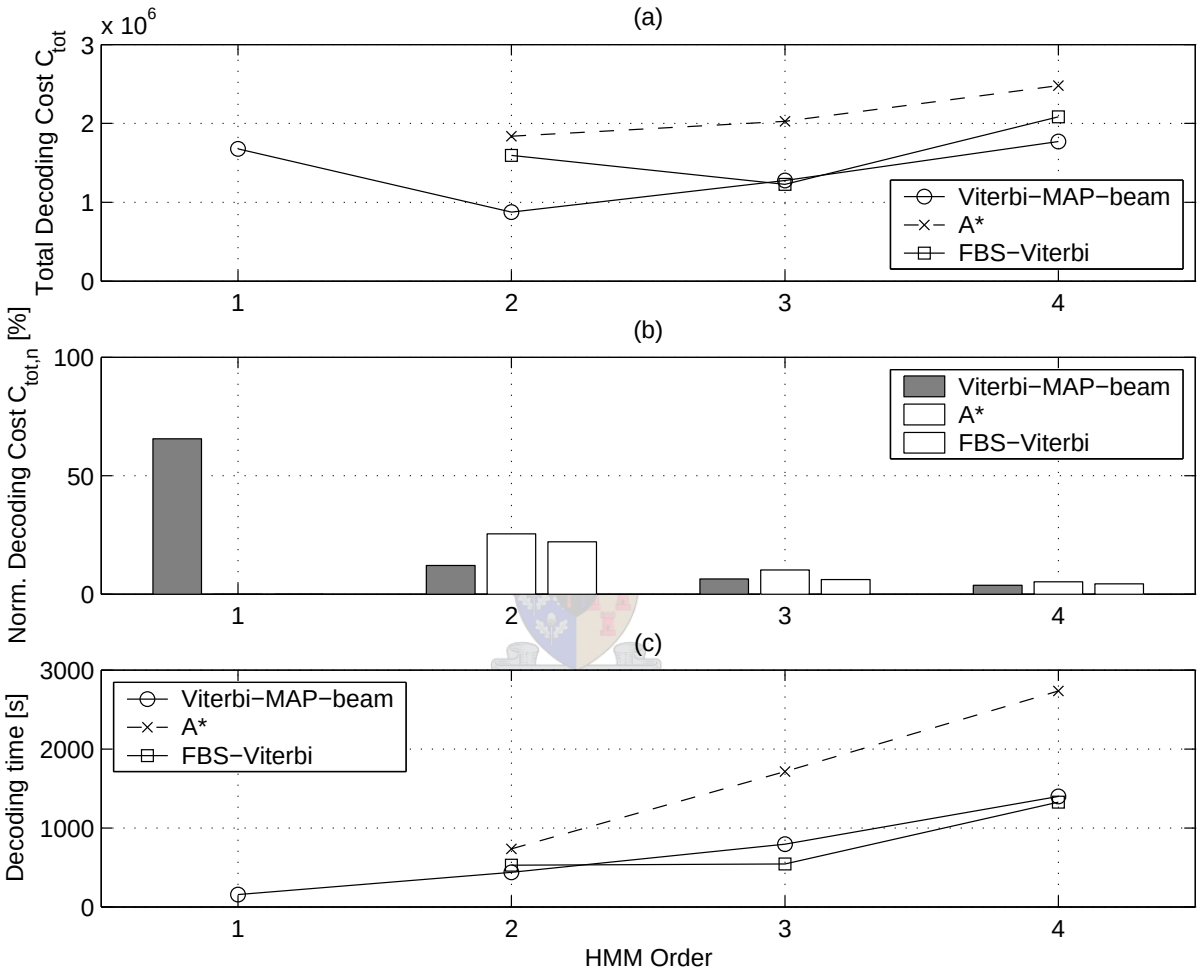
Figure 6.6: *A comparison of (a) total decoding cost, (b) the normalised total decoding cost, and (c) the total decoding time of the FBS-Viterbi-beam, A* and Viterbi-MAP-beam decoders, during the forward decoding of high-order HMMs with $N = 40$ emitting-state and DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*



beam-width of $B = 22$, in order to correctly decode all segments. For eighth- and ninth-order HMMs, the computational expense of the Viterbi-MAP-beam decoder increases significantly. This could be caused by poorly estimated transition probabilities, which is caused by insufficient training data for such high-order transition probabilities. These poorly estimated transition probabilities could cause the Viterbi-MAP-beam decoder to require a much wider beam-width in order to correctly decode all segments. Thus the Viterbi-MAP-beam results for the eight- and ninth-order HMMs could be overly pessimistic.

In Fig. 6.5 we can see that at low orders, the total decoding cost of the A* decoder is

Figure 6.7: *A comparison of (a) the total decoding cost, (b) the normalised total decoding cost, and (c) the total decoding time of the FBS-Viterbi-beam, A* and Viterbi-MAP-beam decoders, during the forward decoding of high-order HMMs with $N = 50$ emitting-state and DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*



less than the total decoding cost of the FBS-Viterbi-beam decoder. However, when the HMMs with orders larger than fourth-order are decoded, the FBS-Viterbi-beam decoder is computationally less expensive than the A* decoder. This behaviour is not repeated for $N = 40$, as can be seen in Fig. 6.6.

Fig. 6.8 shows the relative improvement in total decoding cost for the different size, high-order HMMs. For $N = 10$ we can see that the relative improvement in total decoding cost of the FBS-Viterbi-beam decoder seems to increase with the order of the HMMs. As previously mentioned, this could be caused by an overly pessimistic view of the minimum decoding expense of the Viterbi-MAP-beam decoder at high orders.

Figure 6.8: *The improvement in the normalised total decoding cost of the FBS-Viterbi-beam and A* decoders, relative to the Viterbi-MAP-beam decoder, during the forward decoding of high-order HMMs with (a) $N = 10$, (b) $N = 40$, and (c) $N = 50$ emitting states and DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*

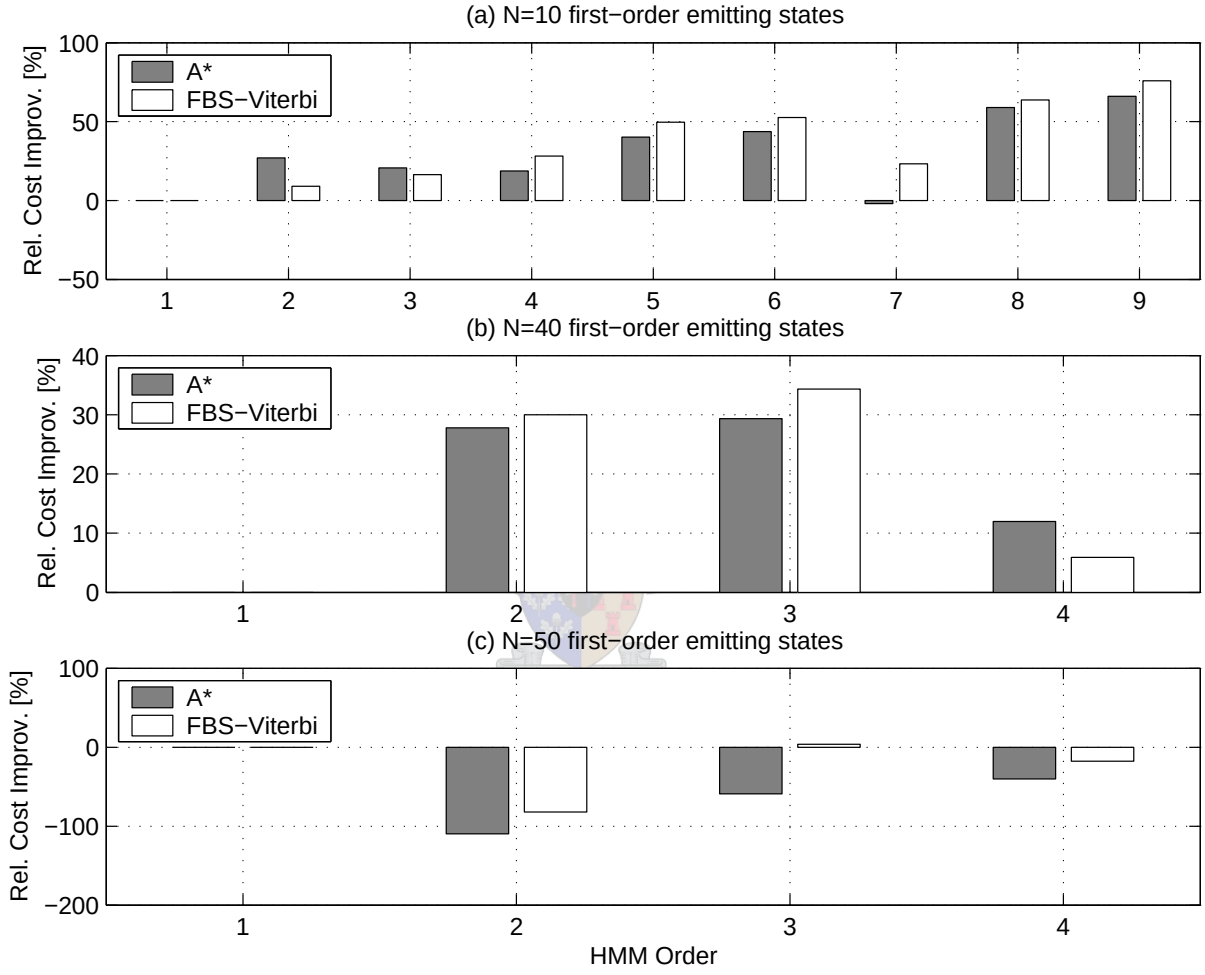
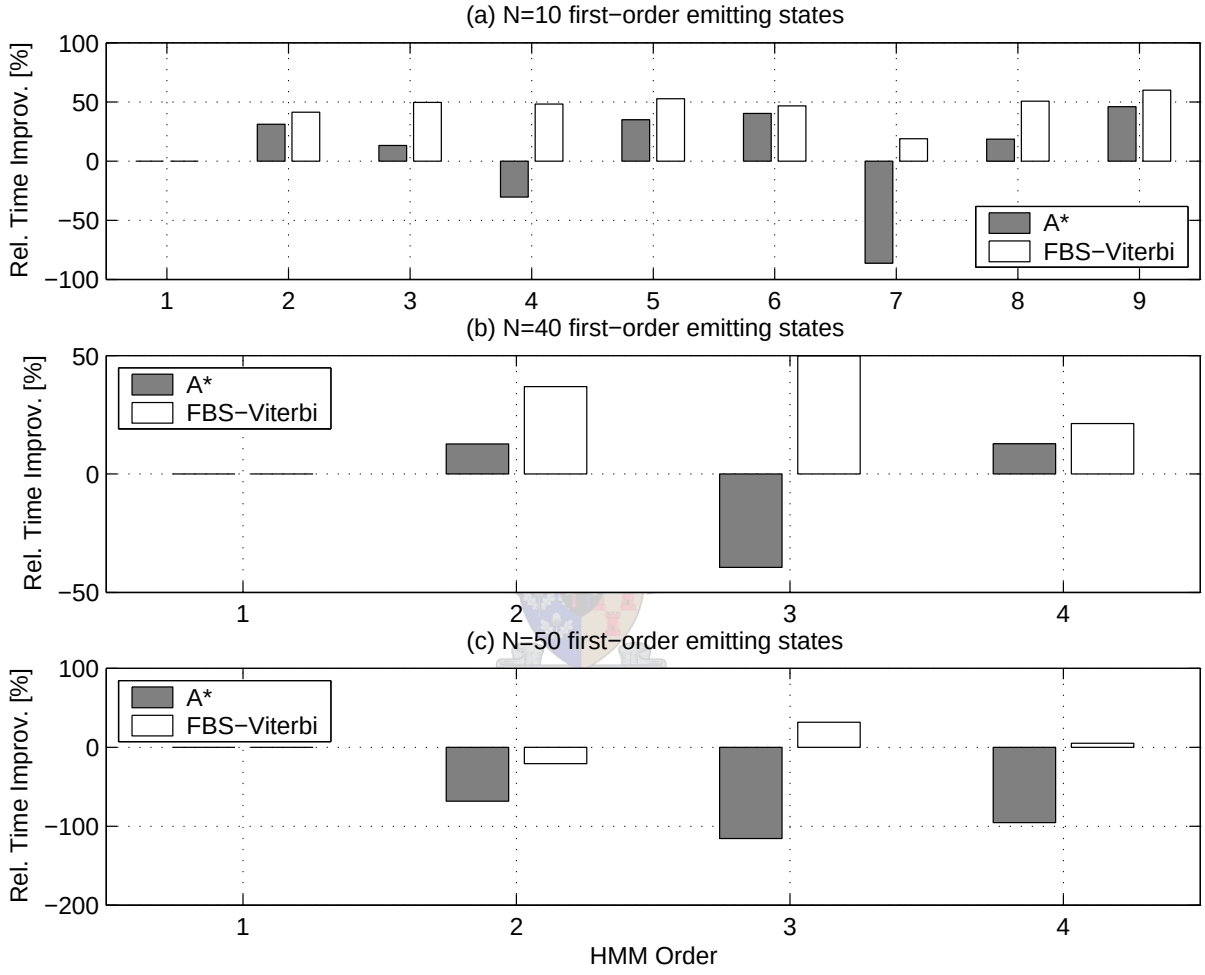


Fig. 6.9 shows the relative improvement in total decoding time for the different size, high-order HMMs. For $N = 10$ the FBS-Viterbi-beam decoder is always faster than the rest of the decoders. The A* decoder is faster than the Viterbi-MAP-beam decoder, except for fourth-order and seventh-order HMMs.

The results for $N = 40$ is the same as was reported in the previous experiment (Section 6.3.1). When the size of the HMM is further increased to $N = 50$, *while still using simple DC-Gaussian state output pdfs*, Fig. 6.7 indicates that the Viterbi-MAP-beam decoder is now computationally less expensive than the FBS-based decoders. This can also be readily seen in Fig. 6.8(c) and Fig. 6.9(c).

Figure 6.9: *The improvement in the total decoding time of the FBS-Viterbi-beam and A* decoders, relative to the Viterbi-MAP-beam decoder, during the forward decoding of high-order HMMs with (a) $N = 10$, (b) $N = 40$, and (c) $N = 50$ emitting states and DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*



Although the FBS-based decoders are computationally more efficient for relatively small HMMs ($N = 10$ and $N = 40$), when the state output pdfs are kept the same, the size of the HMM influences the efficiency of the FBS-based decoders. We suspect that as the number of distinct emitting states N increases, the overlap of the state output pdfs in the observation space increases. This decreases the ability of the state output pdfs to discriminate between states, given an observation sequence. Since the model is now less certain which models are more probable at a specific time, the heuristic function is less able to guide the decoding search, thus causing the FBS-based decoders to be less efficient. This matter of the state output pdfs will be further investigated in the next

section.

6.3.3 The influence of the complexity of the output pdfs

Motivation

In the previous experiment we determined that the state output pdfs do influence the computational efficiency of the FBS-based decoders. The purpose of this experiment is to determine whether increasing the complexity of the state output pdfs, and therefore the ability of the model to discriminate between states, will result in more computationally efficient FBS-based decoding.

Experimental setup

Since the Viterbi-MAP-beam decoder was computationally more efficient when decoding HMMs with $N = 50$ first-order emitting states and DC-Gaussian state output pdfs, we will investigate varying state output pdfs using a high-order HMM with $N = 50$ first-order emitting states. The density functions of the $N = 50$ emitting-state, first-order HMM are initialised by first dividing the training data into N regions utilising vector quantisation. The means of the N regions are then used to initialise diagonal covariance Gaussian output density functions.

We then proceed to train the following three types of pdfs:

- Single diagonal-covariance Gaussian pdfs.
- 8 mixture, diagonal-covariance Gaussian Mixture Models (GMMs).
- 16 mixture, diagonal-covariance Gaussian Mixture Models (GMMs).

The GMMs are trained by employing the method of mixture-splitting on the single diagonal-covariance Gaussians. During the training of the first-order HMM, the mixture component with the largest weight is split along its principal axis. The training of the first-order HMM continues until the desired number of mixtures are reached.

The transition probabilities are initialised to be equal. The FIT algorithm trains the R-th order HMM by starting with a fully-connected, first-order HMM and increasing the order incrementally. Between each increase in order the HMM parameters are trained using the Viterbi-re-estimation algorithm. State transition probabilities which drop below a threshold of 10^{-5} are removed from the HMM. The sizes of the resultant high-order, left-context HMMs are shown in Table 6.2.

Each of the high-order, left-context HMMs were decoded with respectively the standard Viterbi-MAP-beam decoder, the FBS-Viterbi-beam decoder and the A* search decoder. For the FBS-based decoders we also determine the minimum computational expense for each order of derived HMM i.e. $(R - K) = 1, 2, \ldots, R - 1$.

Table 6.2: *The number of transitions and states of high-order, left-context HMMs with $N = 50$ first-order emitting states. The HMMs respectively use DC-Gaussian, 8-mixture DC-Gaussian and 16-mixture DC-Gaussian state output pdfs.*

HMM order R	DC-Gaussian		GMM 8 DC-Gaussian		GMM 16 DC-Gaussian	
	# states	C_{tot}	**# states**	C_{tot}	**# states**	C_{tot}
1	52	1,277	52	1,293	52	1,300
2	934	3,615	750	3,096	771	3,213
3	2,882	9,904	2,676	9,204	2,794	9,523
4	8,700	23,786	8,421	22,634	8,676	23,169
5	22,246	52,153	21,600	50,051	22,078	50,714

Results

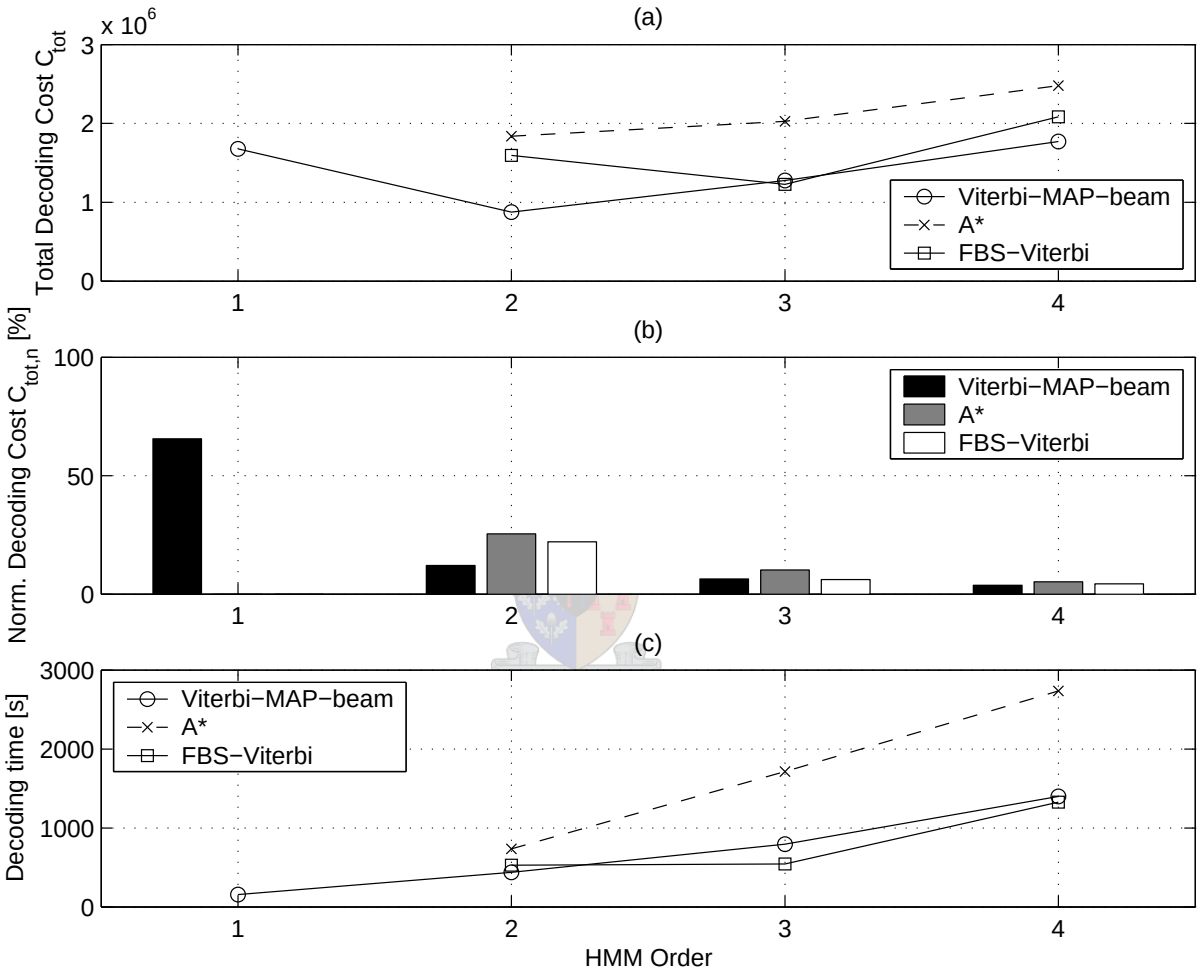
Figs. 6.10(a), 6.11(a) and 6.12(a) respectively illustrate the total decoding cost (the total number of transitions evaluated) of each of the Viterbi-MAP-beam, A* and FBS-based Viterbi-beam decoder, for the DC-Gaussian, 8-mixture DC-Gaussian and 16-mixture DC-Gaussian state output pdfs. Figs. 6.10(b), 6.11(b) and 6.11(b) illustrate the normalised total decoding cost while Figs. 6.10(c), 6.11(c) and 6.12(c) show the implementation dependent total decoding time. For reference, the computational expense of decoding high-order HMMs with $N = 50$ and respectively DC-Gaussian, 8-mixture DC-Gaussian and 16-mixture DC-Gaussian state output pdfs are tabulated in Table C.7.

Figs. 6.13(a), (b) and (c) illustrate the improvement in normalised total decoding cost of the FBS-based decoders relative to the Viterbi-MAP-beam decoder, for decoding high-order HMMs with $N = 50$ and respectively DC-Gaussian, 8-mixture DC-Gaussian and 16-mixture DC-Gaussian state output pdfs. Figs. 6.14(a), (b) and (c) illustrate the improvement in total decoding time of the FBS-based decoders relative to the Viterbi-MAP-beam decoder, for decoding high-order HMMs with $N = 50$ and respectively DC-Gaussian, 8-mixture DC-Gaussian and 16-mixture DC-Gaussian state output pdfs.

Interpretation

The results for decoding high-order HMMs with $N = 50$ and DC-Gaussian state pdfs are the same as was reported in the previous experiment (Section 6.3.2). When the complexity of the state output pdfs is increased to 8-mixture DC-Gaussian pdfs, we can see in Fig. 6.11 that the Viterbi-MAP-beam decoder is still computationally more efficient than the FBS-based decoders. However, the difference in computational efficiency is less than when simple DC-Gaussian state pdfs are used. This can also be seen in Fig. 6.13, which shows the relative improvement in the normalised total decoding cost of the FBS-based decoders vs. the Viterbi-MAP-beam decoder. When the high-order HMMs use

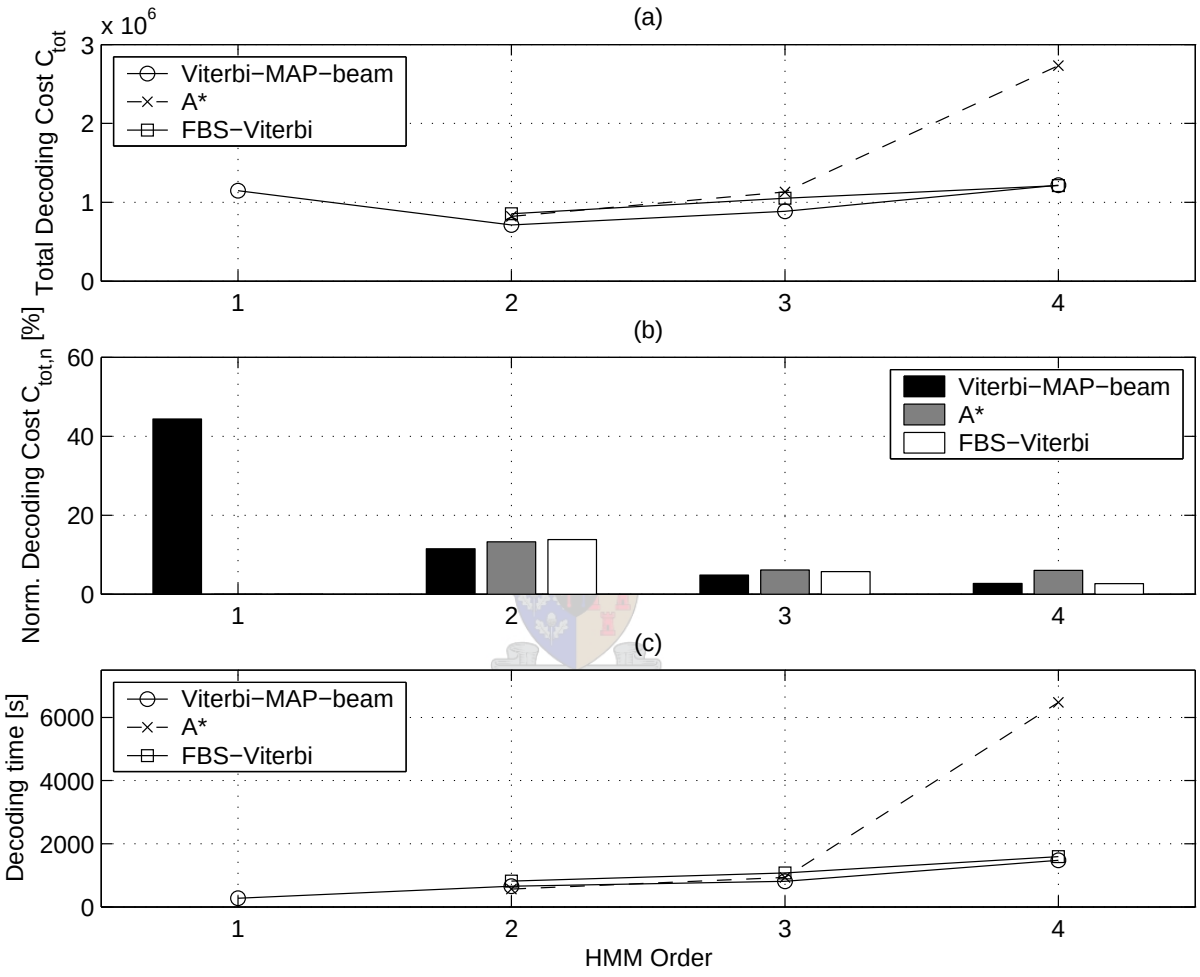
Figure 6.10: *A comparison of (a) the total decoding cost, (b) the normalised total decoding cost, and (c) the total decoding time of the FBS-Viterbi-beam, A* and Viterbi-MAP-beam decoders, during the forward decoding of high-order HMMs with $N = 50$ emitting-states and DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*



simple DC-Gaussian state pdfs, the relative degradation is more than 50% for the A* decoder. When the high-order HMMs use 8-mixture DC-Gaussian state pdfs, the relative degradation decreases to be in the range $0 - 30\%$.

If the complexity of the state output pdfs is further increased to 16-mixture DC-Gaussian pdfs, Figs. 6.12(a) and (b) once again indicate that the FBS-based decoders are computationally more efficient than the Viterbi-MAP-beam decoder. In Fig. 6.13(c) we can see that the relative improvement in the normalised total decoding cost of the FBS-based decoders vs. the Viterbi-MAP-beam decoder, is more than 50% for the FBS-based decoders. If we examine the implementation dependent decoding time, we can also

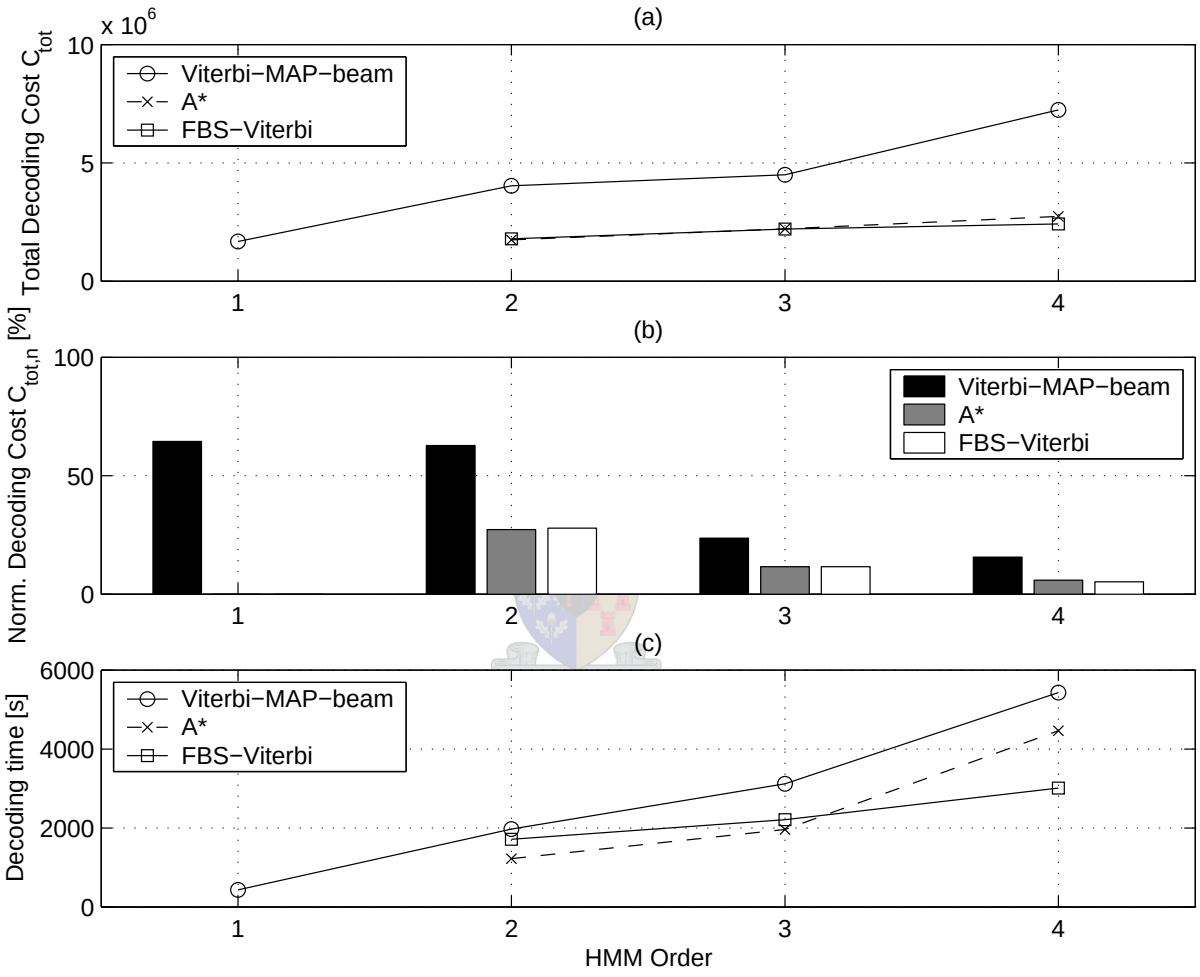
Figure 6.11: *A comparison of (a) the total decoding cost, (b) the normalised total decoding cost, and (c) the total decoding time of the FBS-Viterbi-beam, A* and Viterbi-MAP-beam decoders, during the forward decoding of high-order HMMs with $N = 50$ emitting-states and 8-mixture DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*



see in Fig. 6.14 that increasing the complexity of the state output pdfs to 16-mixture DC-Gaussian causes the FBS-based decoders to be at least 10% faster than the Viterbi-MAP-beam decoder. The A* decoder is at least 30% faster than the Viterbi-MAP-beam decoder, while for third- and fourth-order HMMs the FBS-Viterbi-beam decoder is more than 30% faster than the Viterbi-MAP-beam decoder.

These results indicate that more detailed state output pdfs allow the heuristic function to once again efficiently guide the decoding of high-order HMMs. Since more detailed state output pdfs generally result in higher recognition accuracies, we are not overly concerned that the FBS-based decoders are less computationally efficient than the Viterbi-MAP-

Figure 6.12: *A comparison of (a) the total decoding cost, (b) the normalised total decoding cost, and (c) the total decoding time of the FBS-Viterbi-beam, A* and Viterbi-MAP-beam decoders, during the forward decoding of high-order HMMs with* $N = 50$ *emitting-states and* 16*-mixture DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*



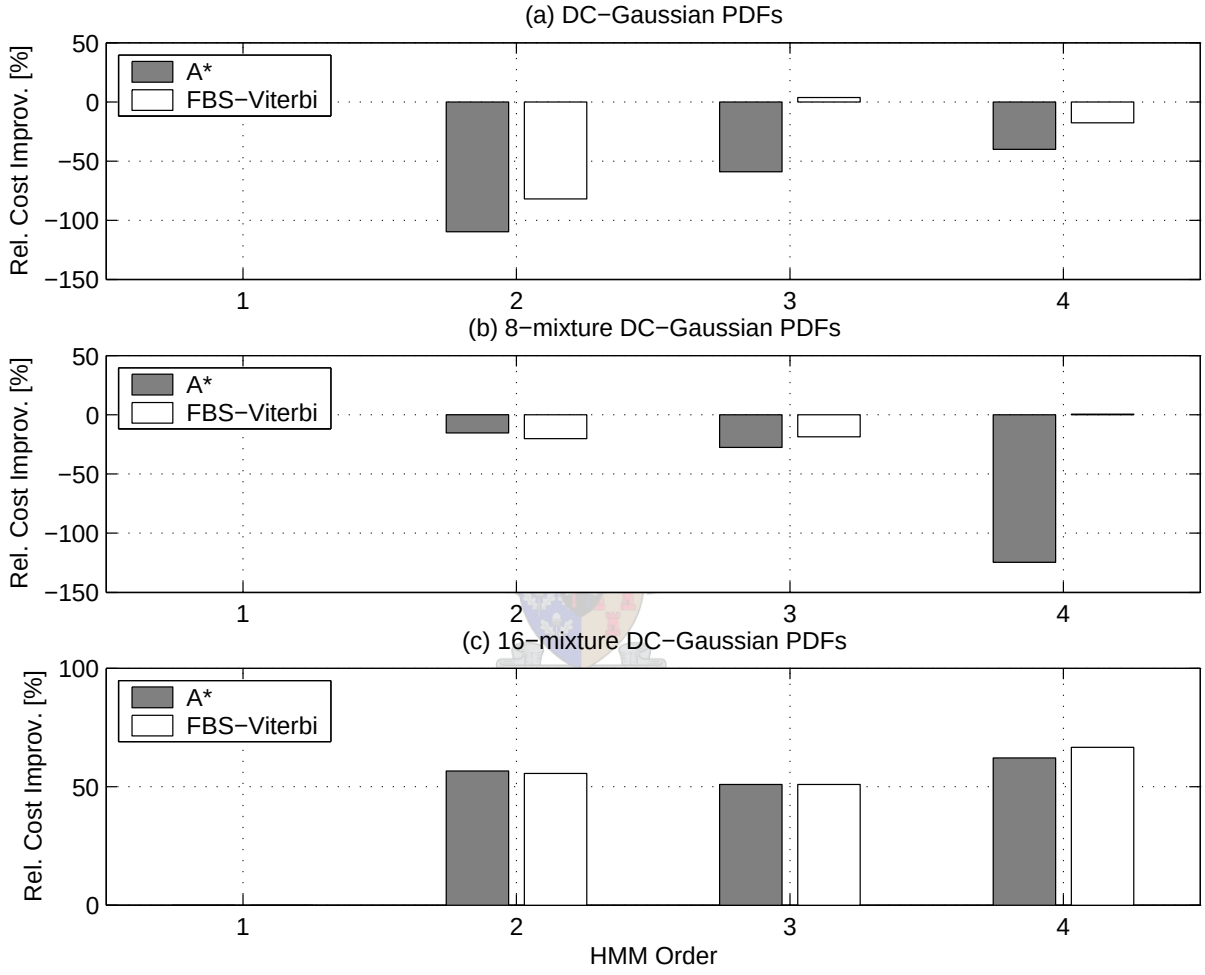
beam decoder, when simple pdfs are used.

6.3.4 What is the optimal choice of derived HMM order $(R-K)$?

Motivation

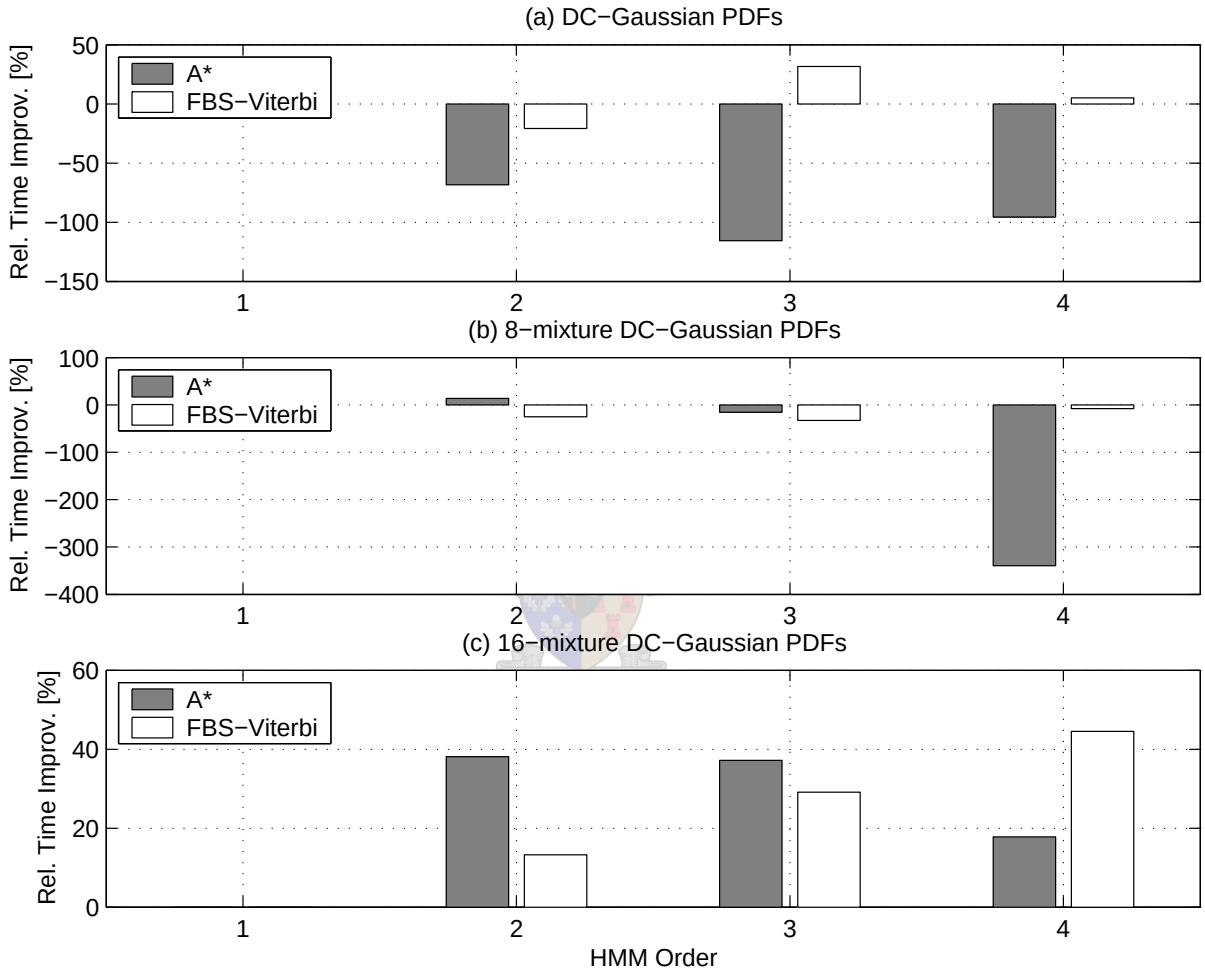
The purpose of this analysis is to examine the order of the derived, lower-order HMM that will result in the most computationally efficient decoding. This will be determined by analysing the total decoding cost C_{tot} to determine what percentage of the total cost is attributed to respectively the search cost C_{s}, the heuristic cost C_{h} and the conversion

Figure 6.13: *The improvement in the normalised total decoding cost of the FBS-Viterbi-beam and A* decoders, relative to the Viterbi-MAP-beam decoder, during the forward decoding of high-order HMMs with $N = 50$ emitting-states and (a) DC-Gaussian, (b) 8-mixture DC-Gaussian, and (c) 16-mixture DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*



cost C_c. We want to determine the conversion cost, since we are also interested in the cost of converting the right-context, best-path backward probability into the left-context, best-path backward probability, using the "forward" pointers that are determined when backward decoding the right-context HMM (as discussed in Section 5.1). Since the experiment with $N = 10$ and DC-Gaussian state output pdfs includes HMMs with orders up to ninth-order, we will use the results of that experiment for our analysis.

Figure 6.14: *The improvement in the total decoding time of the FBS-Viterbi-beam and A* decoders, relative to the Viterbi-MAP-beam decoder, during the forward decoding of high-order HMMs with $N = 50$ emitting-states and (a) DC-Gaussian, (b) 8-mixture DC-Gaussian, and (c) 16-mixture DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*

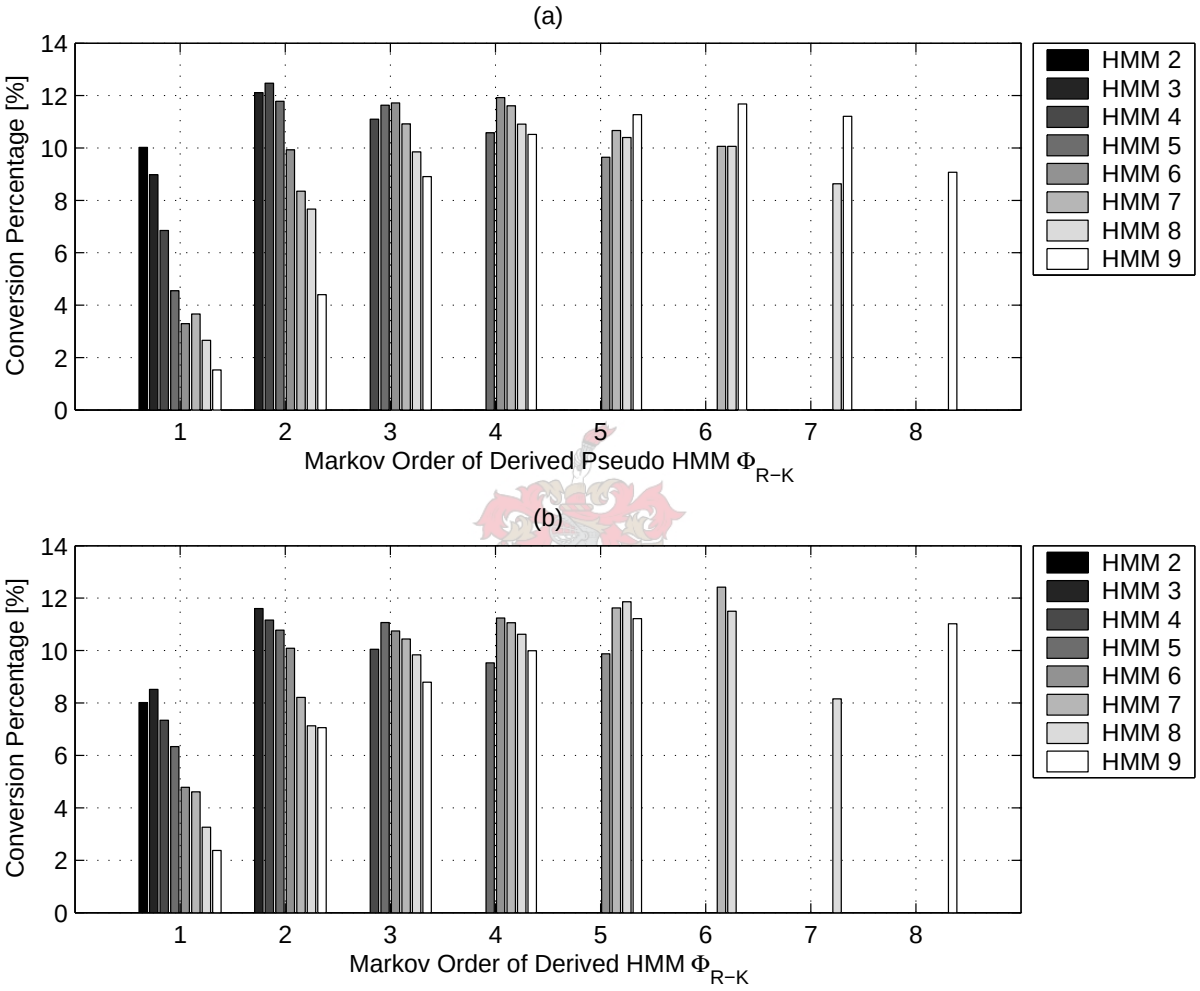


Experimental setup

During the FBS-based decoding experiments of high-order HMMs (that were presented in the previous experiments), we collected the following information for each order R of HMM and for each of the 1410 segments:

- The total number of transitions evaluated (the total decoding cost C_{tot}).
- The number of transitions evaluated by the forward search algorithm (the search cost C_{s}).
- The number of transitions evaluated by the heuristic calculation (the heuristic cost

Figure 6.15: *The normalised heuristic conversion cost (the percentage of the total decoding cost attributed to converting the heuristic) of (a) the A* decoder, and (b) the FBS-Viterbi-beam decoder, when using information from varying low-order derived HMMs that are derived from $N = 10$ emitting state high-order, left-context HMMs.*



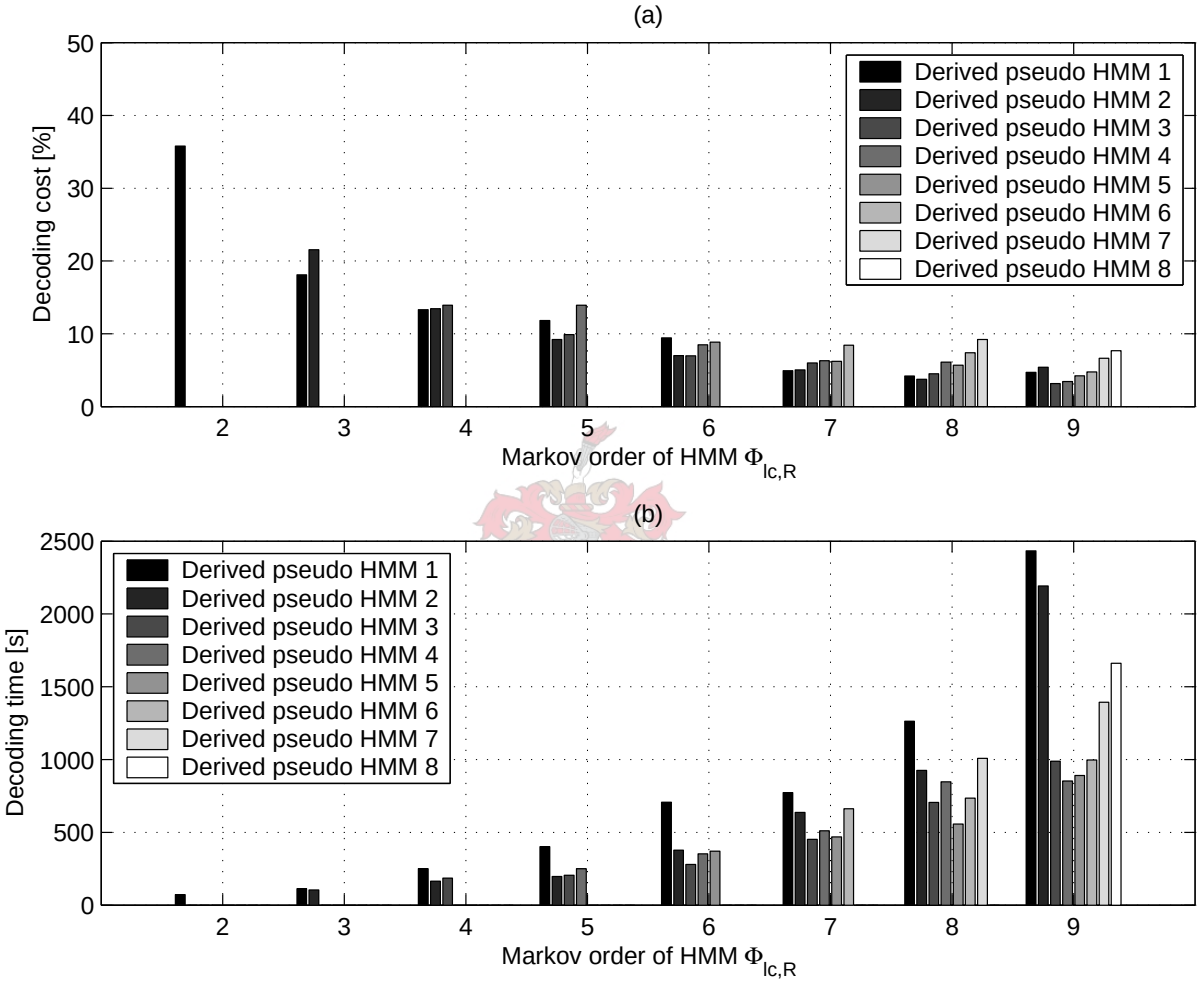
C_{h})

- The number of transitions evaluated during the conversion of the right-context, best-path probabilities to left-context, best-path probabilities (the conversion cost C_{c}).
- The total decoding time.

Results

We do not show the complete analysis of the total decoding cost for the second-order to the ninth-order here. For reference, the complete analysis of the total decoding cost of

Figure 6.16: *(a) The normalised total decoding cost, and (b) the toal decoding time of the A* decoder, during the forward decoding of high-order, left-context HMMs with $N = 10$ emitting-states and varying order derived pseudo HMMs, with beams set wide enough to decode all segments correctly.*

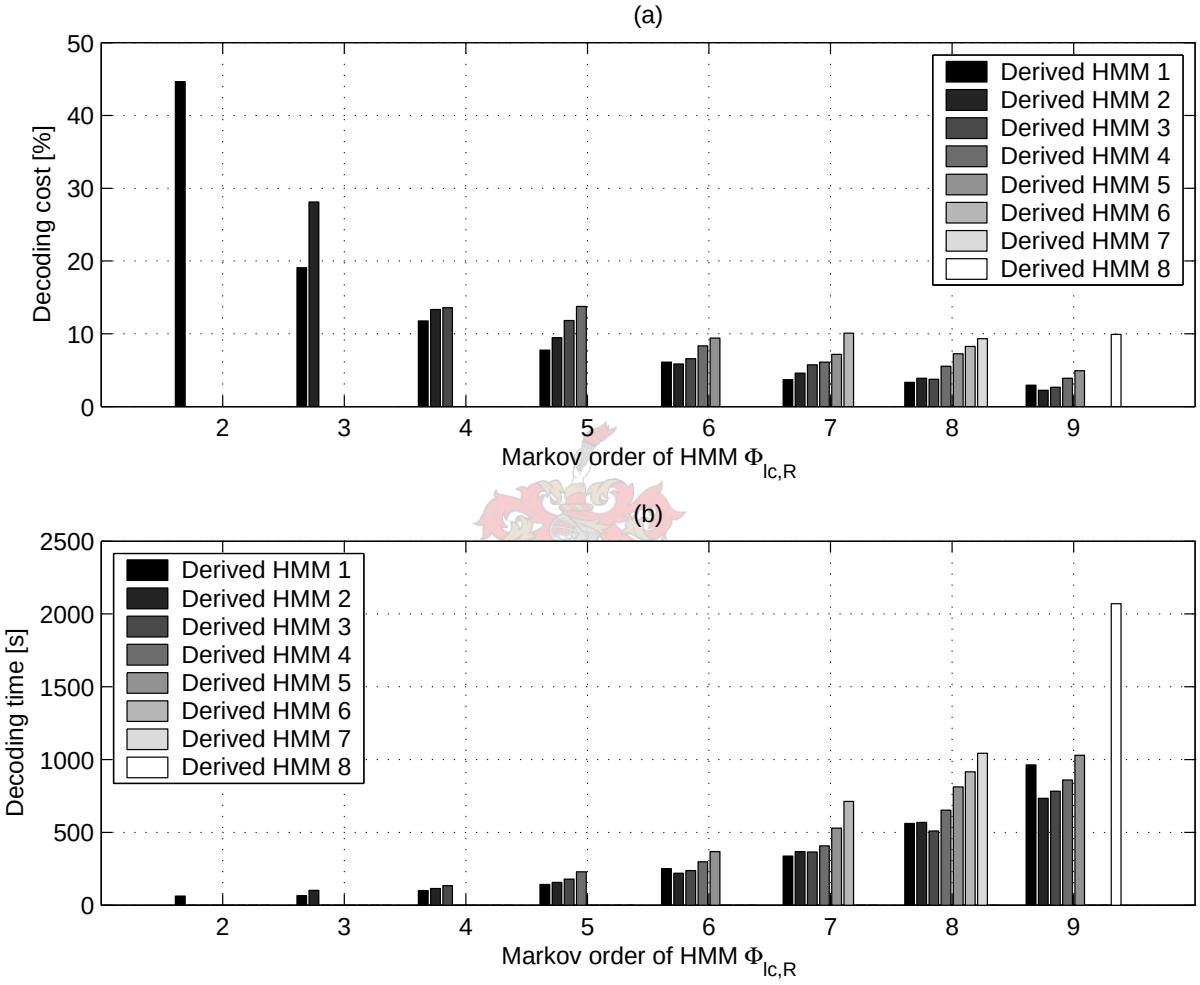


the A* decoder and FBS-Viterbi-beam decoder are respectively tabulated in Table C.9 and Table C.8. We note that during the FBS-Viterbi-beam decoding of the ninth-order HMM, results could not be obtained when the order of the derived HMM was $(R-K) = 6$ and $(R-K) = 7$.

In Fig. 6.15(a) we show the normalised conversion cost (the percentage of the total decoding cost attributed to converting the right-context heuristic to a left-context heuristic) when using the A* decoder. In Fig. 6.15(b) we show the normalised conversion cost when using the FBS-Viterbi-beam decoder.

Figs. 6.16(a) and (b) respectively illustrate the total decoding cost and total decoding time, when using the A* decoder to decode high-order HMMs, when the order of the

Figure 6.17: *(a) The normalised total decoding cost, and (b) the toal decoding time of the FBS-Viterbi decoder, during the forward decoding of high-order, left-context HMMs with $N = 10$ emitting-states and varying order derived HMMs, with beams set wide enough to decode all segments correctly.*



derived pseudo HMM is varied from $(R - K) = 1, \ldots, R - 1$.

Figs. 6.17(a) and (b) respectively illustrate the total decoding cost and total decoding time, when using the FBS-Viterbi-beam decoder to decode high-order HMMs, when the order of the derived HMM is varied from $(R - K) = 1, \ldots, R - 1$.

Interpretation

In Fig. 6.15 we see that the heuristic conversion cost is on average approximately 10% of the total decoding cost and never more than 13% of the total decoding cost. This result supports our statement in Section 4.5.1 that the method of using the "forward" pointers to convert the right-context, best-path backward probability is computationally

efficient. It is interesting to note that the heuristic conversion cost is generally lower for the FBS-Viterbi-beam decoder than for the A* decoder.

If we examine the total decoding cost and total decoding time for the A* decoding in Fig. 6.16, we see that the minimum total decoding cost is obtained when the derived, low-order pseudo HMM has an order of approximately half the order of the left-context, high-order HMM from which it is derived (i.e. $R - K \approx \frac{1}{2}R$). When the order of the derived pseudo HMM is much smaller than the high-order HMM, we find that most of the total decoding cost can be attributed to the search cost. Since the A* search algorithm requires additional bookkeeping, this causes the total decoding time to increase sharply.

If we examine the total decoding cost and total decoding time for the FBS-Viterbi-beam decoding in Fig. 6.17, we see that the minimum total decoding cost is generally obtained when the derived, low-order HMM is first- or second-order. We can also see that there is a much stronger correlation between the total decoding cost and the total decoding time.

6.3.5 The computational consistency of the FBS-based decoders

Motivation

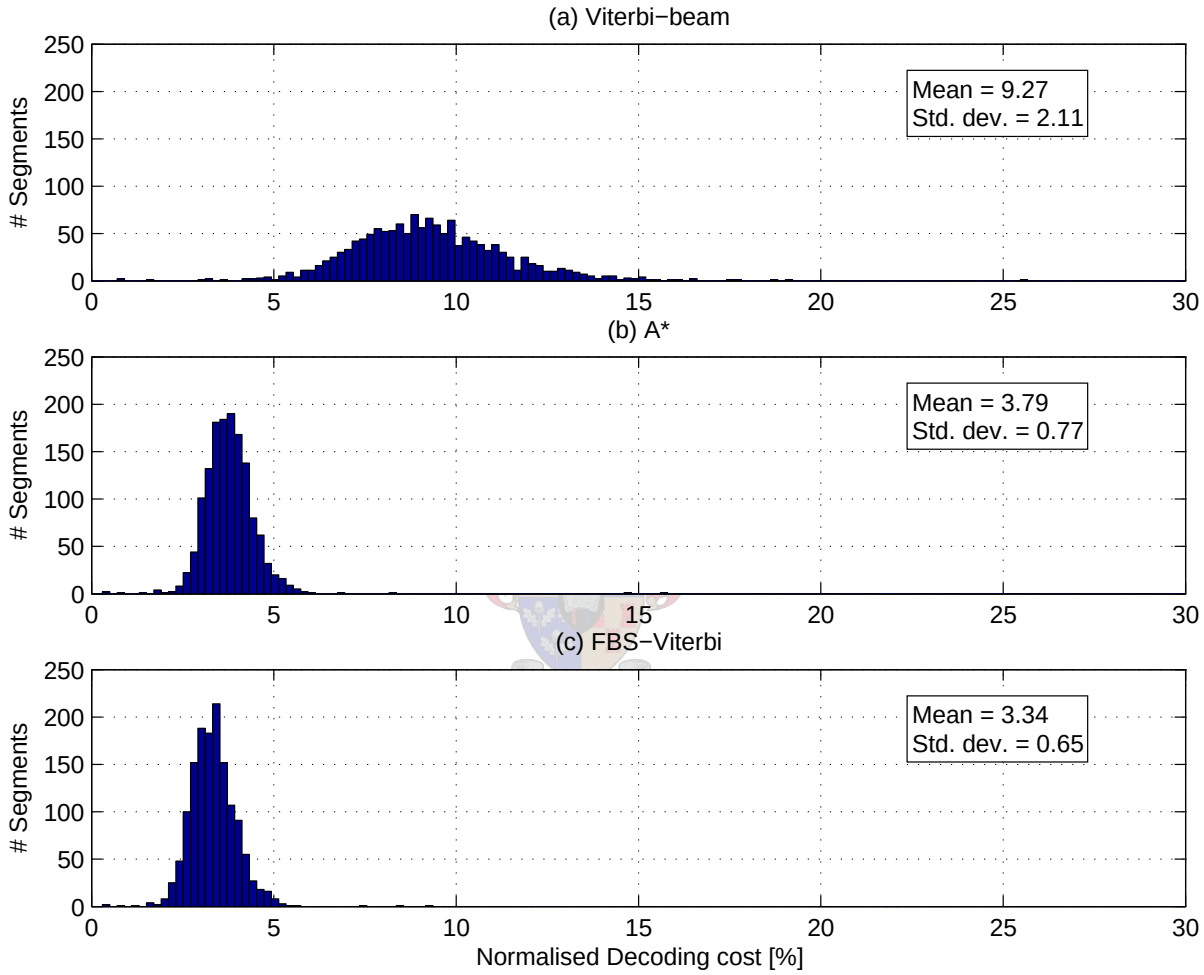
We wish to analyse the decoding results of the FBS-based decoders in order to determine whether they are more computationally consistent than the Viterbi-MAP-beam decoder, with respect to the number of evaluated transitions and the total decoding time. We wish to determine if the computational expense of the new decoders are more predictable than the computational expense of the Viterbi-MAP-beam decoder. We answer this question by performing a statistical analysis of the results obtained when the $N = 10$ first-order emitting-state, high-order HMMs were decoded.

Experimental setup

When analysing the results we determine the minimum, maximum, average and the standard deviation of the number of transitions evaluated for each of the standard Viterbi-MAP-beam decoder, the FBS-Viterbi-beam decoder and the A* search decoder. For the FBS-Viterbi-beam decoder and the A* search decoder we only examine the low-order derived HMM that was computationally the most efficient. For the number of transitions evaluated, we compare the normalised total decoding cost, which is an implementation independent performance measure. To include the A* overhead associated with the ordering of the lists, we also compare the normalised total decoding time (which is an implementation dependent performance measure). The total decoding time is normalised with the maximum decoding time of a single segment, when using the Viterbi-MAP-beam decoder.

Results

Figure 6.18: *The histogram of the normalised total decoding cost of 1410 segments when decoding an eight-order HMM with the (a) Viterbi-MAP-beam, (b) A*, and (c) FBS-Viterbi-beam decoder, with beams set wide enough to decode all segments correctly.*

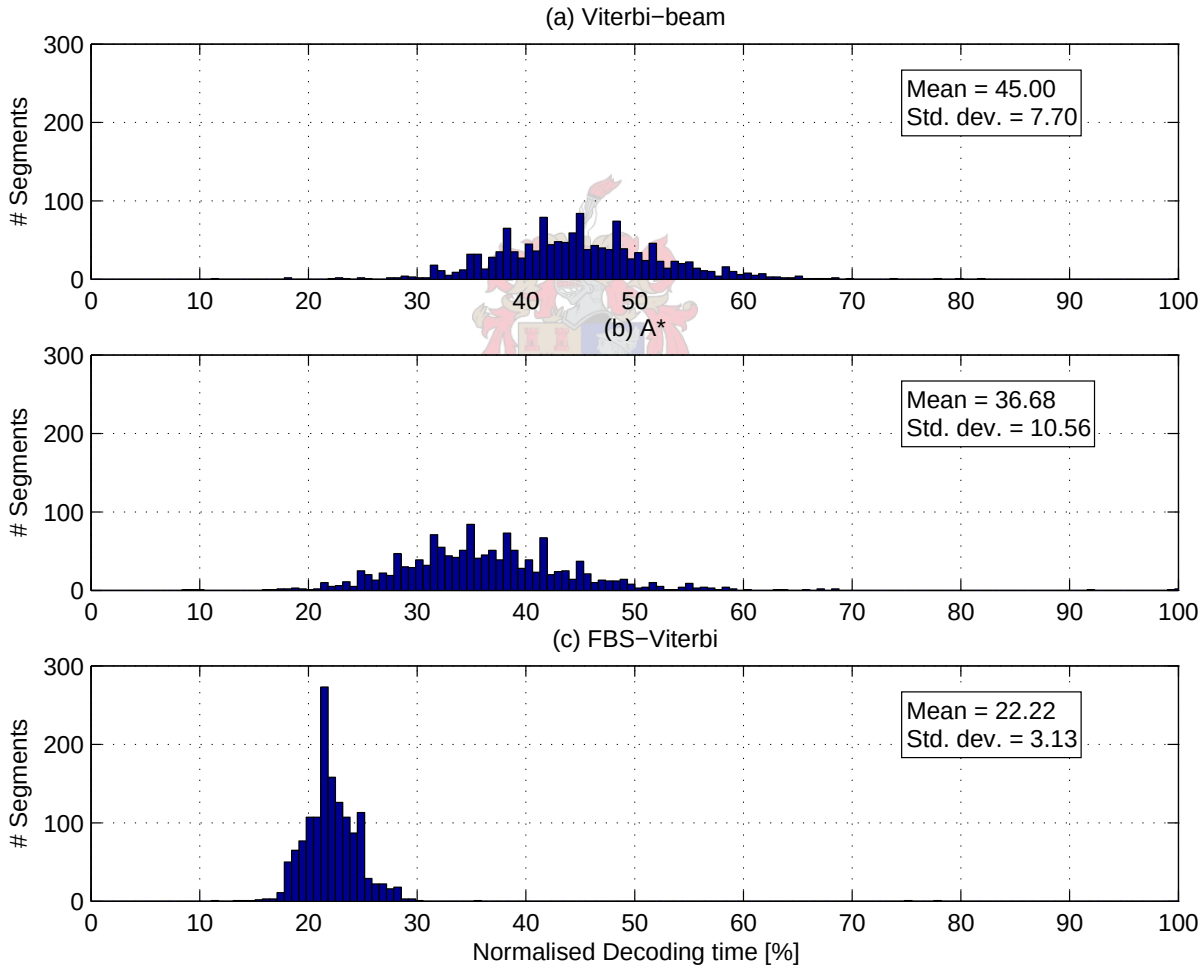


The complete analysis for the three different decoders is tabulated in Table C.10. The analysis shown in Table C.10 shows that not only do the FBS-based decoders have a lower average decoding cost, the standard deviation of the decoding cost is also significantly smaller than that of the Viterbi-MAP-beam decoder. We do not show all the results here, but have chosen to only present the analysis of the seventh- and eight-order HMM. We have specifically chosen the seventh-order HMM, since this is the case where the computational expense of the Viterbi-MAP-beam decoder is nearly the same as that of the FBS-based decoders (see Fig. 6.5). The analysis of the eight-order HMM illustrates the case where both FBS-decoders are computationally more efficient than the Viterbi-MAP-beam decoder, but the A* decoder is slower. These two cases clearly illustrate the

computational consistency of the FBS-based decoders, specifically the FBS-Viterbi-beam decoder.

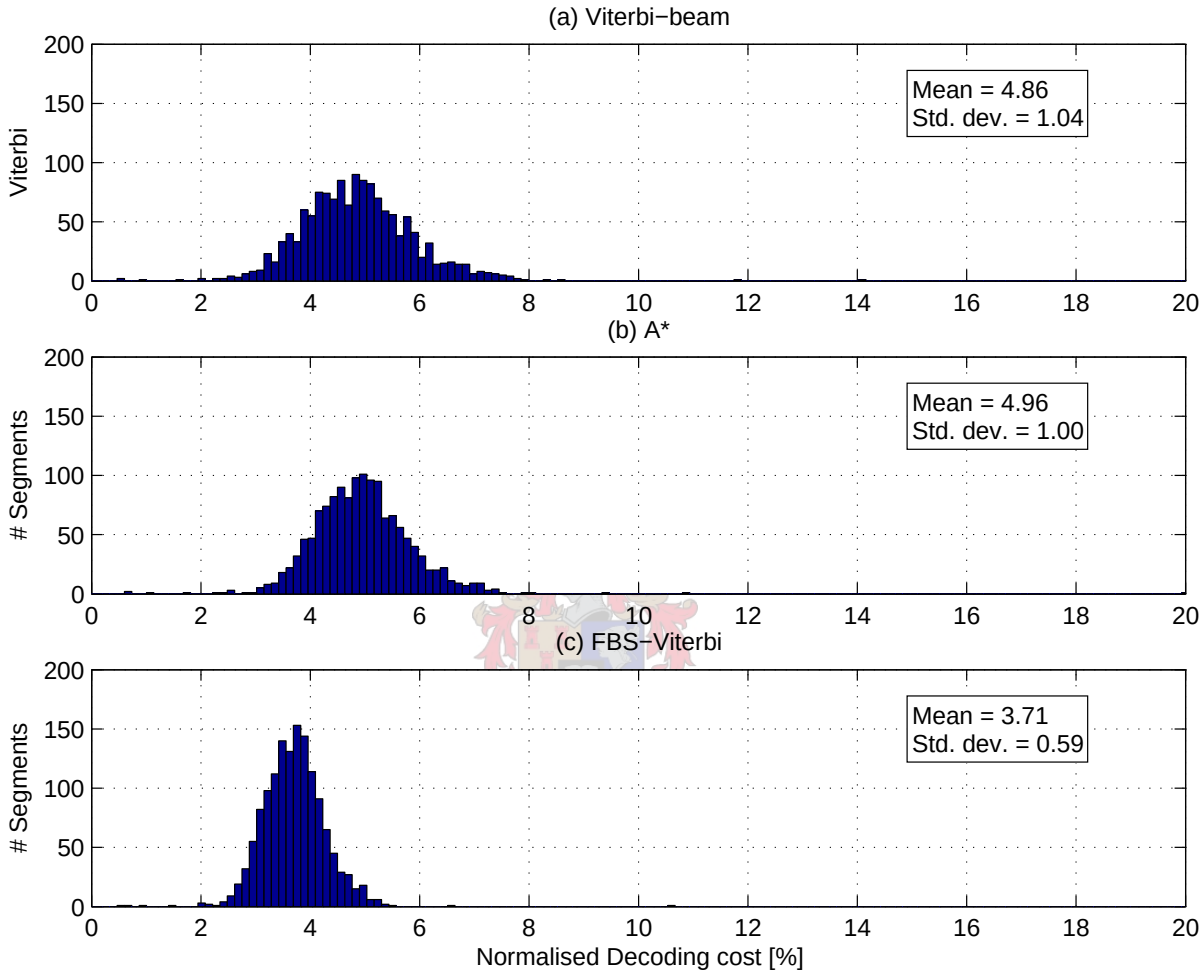
In Figs. 6.18 and 6.19 we respectively show the histogram of the normalised total decoding cost and the normalised total decoding time of the 1410 segments, given the eight-order HMM, when using the Viterbi-MAP-beam, A* and FBS-Viterbi-beam decoders.

Figure 6.19: *The histogram of the normalised total decoding time of 1410 segments when decoding an eight-order HMM with the (a) Viterbi-MAP-beam, (b) A*, and (c) FBS-Viterbi-beam decoder, with beams set wide enough to decode all segments correctly.*



In Fig. 6.20 and Fig. 6.21 we respectively show the histogram of the total decoding cost and the normalised total decoding time of the 1410 segments, given the seventh-order HMM, when using the Viterbi-MAP-beam, A* and FBS-Viterbi-beam decoders.

Figure 6.20: *The histogram of the normalised total decoding cost of the 1410 segments when decoding a seventh-order HMM with the (a) Viterbi-MAP-beam, (b) A*, and (c) FBS-Viterbi-beam decoder, with beams set wide enough to decode all segments correctly.*

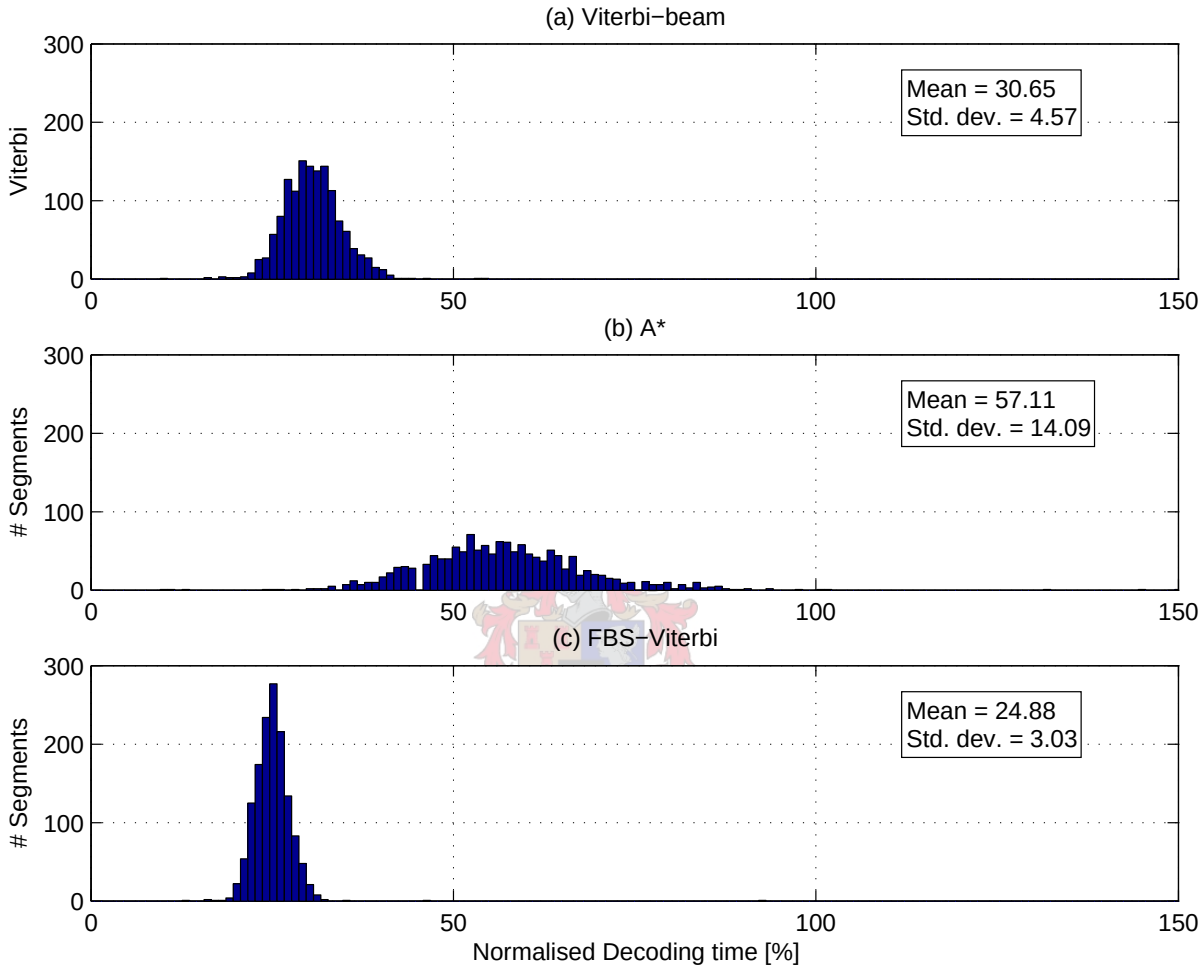


Interpretation

The general trend of the analysis shows that the standard deviation of the decoding cost of the FBS-Viterbi-beam decoder is less than half of the decoding cost of the Viterbi-MAP-beam decoder, for most of the HMM orders. The standard deviation of the decoding cost of the FBS-Viterbi-beam decoder is generally smaller than that of the A* decoder. This is clearly illustrated by the decoding cost distributions shown in Fig. 6.18 and Fig. 6.20. Even for the case of the seventh-order HMM, when the computational expense of the decoders are nearly the same, we can clearly see that the FBS-Viterbi-beam decoder is more consistent, with respect to the number of evaluated transitions, than the Viterbi-MAP-beam decoder.

When we examine the total decoding time, we again see that the FBS-Viterbi-beam

Figure 6.21: *The histogram of the normalised total decoding time of the 1410 segments when decoding a seventh-order HMM with the (a) Viterbi-MAP-beam, (b) A*, and (c) FBS-Viterbi-beam decoder, with beams set wide enough to decode all segments correctly.*



decoder is more consistent, since the standard deviation of the total decoding time is also generally half that of the Viterbi-MAP-beam decoder. However, the standard deviation of the total decoding time of the A* decoder is sometimes much larger than that of the Viterbi-MAP-beam decoder. This is illustrated by the distribution of the total decoding time shown in Fig. 6.19 and Fig. 6.21. However, we can clearly see the narrow standard deviation of the total decoding time of the FBS-Viterbi-beam decoder. We can thus conclude that the FBS-Viterbi-beam decoder is not only computationally more efficient than the Viterbi-MAP-beam decoder, it is also computationally more consistent.

6.4 Summary

In this chapter we first verified our assumption that derived HMMs (both pseudo HMM and true HMMs) exhibit similar backward decoding behaviour, with respect to the number of evaluated transitions, as the HMM decoding results presented in previous chapters. It was important to verify this assumption as it allows us to use the derived HMMs in our FBS-based decoders, confident in the fact that decoding the derived HMMs will not add unnecessary computational expense to the decoding of the high-order HMMs.

We then continued with our evaluation of the proposed FBS-based decoders by measuring the computational expense of decoding high-order, left-context HMMs. We compared these results to the baseline Viterbi-MAP-beam decoding results presented in chapter 2. This was done for varying sizes of high-order HMMs as well as state output pdfs that range from simple DC-Gaussian pdfs to more detailed 16-mixture DC-Gaussian GMMs.

These results have shown that including information obtained from decoding low-order HMMs significantly reduces the computational expense of decoding high-order HMMs. The decoding results obtained using the FBS-Viterbi-beam decoder indicate that state pruning based on the *complete* observation also results in significantly smaller search graphs being explored than state pruning based on *partial* observations. However, the advantage of using low-order HMMs to guide the search is dependent on using state output pdfs that are detailed enough to allow the heuristic to accurately predict the probability of the best path from a given state to the final state. These results also show that the FBS-Viterbi-beam decoder is computationally more efficient than the A* decoder. We note that if the FBS-Viterbi-beam decoder is used in conjuction with a derived first-order HMM, we can simply use a derived, left-context HMM instead of using a right-context HMM, since we have shown that backward decoding left- and right-context, first-order HMMs are computationally equivalent. By using a left-context HMM to calculate the heuristic, we can avoid the added heuristic conversion cost and further reduce the total decoding cost by approximately 10%.

Lastly, the distribution of the total decoding cost and total decoding time was analysed and we showed that the FBS-based decoders, specifically the FBS-Viterbi-beam decoder, is more computationally consistent than the Viterbi-MAP-beam decoder. Thus, not only is the FBS-Viterbi-beam decoder computationally more efficient, it is also computationally more consistent than the Viterbi-MAP-beam decoder.

Chapter 7

Conclusions

The preceding chapters presented the theoretical development and practical verification of Forward-Backward search-based decoding of high-order HMMs. In this last part of the dissertation, the specific results will again be related to the objective of this research. The relevance of these results to the broader research community will be estimated, and avenues for further research will be identified.

7.1 Concluding perspective

The objective of this research was to develop a more time-efficient method of decoding high-order HMMs. The Forward-Backward search paradigm was identified as a method that could incorporate information obtained from low-order HMMs, in order to reduce the decoding time. We proposed, implemented and evaluated two decoders based on the Forward-Backward search paradigm. The first decoder is based on time-synchronous Viterbi-beam decoding and the second is based on time-asynchronous A* search. The experimental results presented in the previous chapter show that both these proposed decoders result in more time-efficient decoding of the fully-connected, high-order HMMs that were investigated, thus fulfilling the objective of this research.

Three interesting facts were uncovered during the course of this research. The first is that conventional forward Viterbi-beam decoding of high-order HMMs is not as computationally expensive as is commonly thought. The results presented in chapter 2 indicate that the computational expense of Viterbi decoding high-order HMMs does increase exponentially with the order R of the HMM. The results in chapters 2 and 6 would seem to indicate that the computational expense of Viterbi-beam decoding also increases exponentially with the order R of the HMM, but that the rate of increase is significantly lower than that of Viterbi decoding (keeping in mind that the decoding experiments were all performed on a single language pair and that only a specific topology was investigated). Although the computational expense sharply increases when decoding high-order HMMs

with large R, we suspect that this is a result of insufficient training data, thus resulting in poorly estimated high-order transition probabilities. The Viterbi-beam decoder does sacrifice admissibility, but the resultant reduction in decoding expense more than compensates for this lack of admissibility. Furthermore, with a proper choice of beam-width, the Viterbi-beam decoder finds the optimal sequence most of the time and the non-optimal sequences are still quite close to the optimal state sequence.

The second interesting fact (uncovered in chapter 3) is that forward and backward decoding of conventional, high-order left-context HMMs are not computationally equivalent. Although forward and backward decoding results in the same optimal state sequence, backward decoding is significantly more expensive than forward decoding of left-context HMMs. We asked the question whether decoding, when time-reversing the observation sequence, is fundamentally more expensive than decoding when the observation sequence is kept the same. We showed that time-reversed decoding is *not* fundamentally more expensive, but that the extra computation is caused by the definition of the left-context state transition probabilities. When backward decoding left-context HMMs, this conditioning of previous states causes a time-asynchronicity between the observations and states under consideration. By developing the right-context HMM, where the state transition probabilities are conditioned on subsequent states, we showed that the backward decoding of a mathematically equivalent high-order, right-context HMM is as expensive as the forward decoding of the high-order, left-context HMM. This leads us to conclude that left- and right-context HMMs are simply two different ways of viewing the same underlying hidden Markov process.

The third interesting fact is that the use of information obtained from low-order HMMs significantly reduces the computational expense of decoding high-order HMMs. In the results presented in chapter 6, both the A* decoder and the FBS-Viterbi-beam decoder outperformed the standard Viterbi-beam decoder. The results indicate that, when the number of first-order emitting states (N) is large *and* less detailed state output pdfs are used, the Viterbi-beam decoder is more time-efficient than the proposed FBS-based decoders. However, when more detailed state output pdfs are used, the proposed FBS-based decoders are once again more time-efficient than the Viterbi-beam decoder. This does not cause us much concern, since more detailed state output pdfs generally result in better pattern recognition performance.

The comparison of the performance of the two new decoders indicates that the FBS-Viterbi-beam decoder is more time-efficient than the A* decoder. The FBS-Viterbi-beam decoder is not only simpler to implement, it also requires less memory since it does not need to perform the additional bookkeeping the A* decoder needs to perform. Furthermore, in order for the A* decoder to be admissible, the derived low-order HMMs are not true HMMs. Since the transition probabilities of the pseudo HMMs do not sum to unity,

decoding the pseudo HMM will result in a larger search graph than decoding the true HMM, when a constant beam-width is used. The advantage of the pseudo HMM is that all the transition "probabilities" will be well estimated, since they are derived from the largest high-order HMM transition probabilities.

7.2 Comparison to prior work

The concept of transition probabilities defined on subsequent states also occurs in the field of Markov processes, where it is called a reversible Markov process. A Markov process is said to show detailed balance if the transition rates between each pair of states i and j in the state space obey

$$P_{ij}\pi_i = P_{ji}\pi_j \tag{7.1}$$

where P is the Markov transition matrix (transition probability), i.e. $P_{ij} = P(s_t = j|s_{t-1} = i)$; and π_i and π_j are the equilibrium probabilities of being in states i and j, respectively [35].

Thus, by more closely examining the properties of reversible Markov processes, we might be able to predict the circumstances under which decoding the right-context HMM will be computationally equivalent to decoding the left-context HMM.

Our decoding of high-order HMMs using information obtained from low-order HMMs is based on the Forward-Backward search paradigm. The FBS paradigm was developed by Austin et al. [1] in order to reduce the decoding time of more complex continuous speech recognition algorithms. Although the speech recognition system used language models, the HMMs were limited to first-order HMMs. Since they were interested in performing real-time speech recognition, the forward search was the less complex search, while the backward search was done using more complex algorithms or acoustic and language models. It seems that the HMMs were simply processed in reverse, without changing the state transition or language model probabilities so that the probabilities are conditioned on the subsequent states or words [30]. We suspect that if Austin et al. performed the backward search first, and then performed a more detailed forward search, they might also have discovered the discrepancy between forward and backward decoding.

Sixtus and Ortmans [43] adapted the Forward-Backward search paradigm to develop a forward-backward word graph pruning algorithm. They showed that the forward-backward pruning method resulted in smaller word graphs than the forward pruning method. The results presented in this dissertation show that forward-backward pruning of states also results in the exploration of a much smaller search graph than forward pruning of states.

7.3 Outstanding issues and further topics of research

The following are still issues that require further investigation:

- We have limited our evaluation of our decoders to topologies that start as a fully-connected topology in the first order, which is an ergodic topology. The use of the newly proposed decoders should be examined with topologies such as left-to-right, ring, random-walk, etc. The biggest challenge is to show that the right-context HMM can still be backward decoded as efficiently as the left-context HMM can be forward decoded for these other topologies. From the field of Markov processes it seems that the solution lies in determining which HMM topologies show detailed balance, and thus are reversible.

- We have only examined one language pair of one speech corpus. The decoders need to be evaluated on other speech corpora and tasks other than speech processing.

- We have limited our evaluation to observations lengths of $T = 2000$. Longer observation features require more memory to store the heuristic function and the observation likelihood cache. Apart from the increase in memory requirements, we do not see any reason why the observation length should influence the computational expense of the decoder, but it should still be investigated.

- It seems that the ability of the state output pdfs to discriminate between states influences the computational expense of the FBS-based decoders. The influence of the pdfs should be thoroughly investigated. This could be done by generating random HMMs with state output pdfs which are initialised to have a specific overlap in the observation space. These random HMMs can be used to generate artificial data, which must then be used to quantify the influence of overlap between state output pdfs on the computational expense of the FBS-based decoders.

- We mentioned that the Viterbi-beam decoding of high-order HMMs might have suffered from transition probabilities which were estimated with insufficient training data. Since the transition probabilities of the derived, low-order HMMs were well estimated, and our proposed decoders were guided by the decoding of these well-estimated HMMs, it is possible that our proposed decoders might have an advantage they would not have had if the high-order transition probabilities were estimated with sufficient training data. This can be investigated via a (much) larger experiment incorporating the full CallFriend database. This should ensure that all high-order HMMs have well estimated parameters.

As previously mentioned, very little has been published on the efficient decoding of high-order HMMs. When high-order HMMs are used, it seems that the decoding is most

often done using the Viterbi-beam algorithm. We suspect that the broader research community regards the Viterbi-beam algorithm as the most efficient method of decoding HMMs. We hope that the research presented in this dissertation will result in renewed investigation into decoding algorithms that are applicable to high-order HMMs.

Bibliography

[1] AUSTIN, S., SCHWARTZ, R., and PLACEWAY, P., "The Forward-Backward Search Algorithm." in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, (Toronto, Canada), pp. 697–700, 1991.

[2] AYCARD, O., MARI, J.-F., and WASHINGTON, R., "Learning to automatically detect features for mobile robots using second-order Hidden Markov Models." *Int. Journal of Advanced Robotic Systems*, December 2004, Vol. 1, No. 4, pp. 231–245.

[3] BAHL, L. *et al.*, "Large Vocabulary Natural Language Continuous Speech Recognition." in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, (Glasgow, Scotland), pp. 465–467, 1989.

[4] BAHL, L., JELINEK, F., and MERCER, R., "A Maximum Likelihood Approach to Continuous Speech Recognition." in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 179–190, 1983.

[5] BAKER, J., "Stochastic modelling for automatic speech understanding." in *Speech Recognition* (REDDY, D. (Ed.)), pp. 521–524, New York: Academic Press, 1975.

[6] BELLMAN, R., *Dynamic Programming*. Princeton: New Jersey: Princeton University Press, 1957.

[7] BENGIO, Y., "Markovian Models for Sequential Data." *Neural Computing Surveys*, 1999, Vol. 2, pp. 129–162.

[8] CANAVAN, A. and ZIPPERLEN, G., *CALLFRIEND American English-Non-Southern Dialect*. Linguistic Data Consortium, Philadelphia, 1996.

[9] DELLER, J. R. J. *et al.*, *Discrete-Time Processing of Speech Signals*. Upper Saddle River, New Jersey: Prentice-Hall, 1993.

[10] DEVIJVER, D. and KITTLER, J., *Pattern Recognition, a statistical approach*. Englewood Cliffs, New Jersey: Prentice-Hall International, 1982.

[11] DU PREEZ, J., *Efficient High-Order Hidden Markov Modelling*. PhD thesis, University of Stellenbosch, November 1997.

[12] DU PREEZ, J., "Efficient training of high-order hidden Markov models, using first-order representations." *Computer Speech & Language*, 1998, Vol. 12, No. 1, No. 1, pp. 23–39.

[13] ENG, C., THIBESSARD, A., HERGALANT, S., MARI, J.-F., and LEBLOND, P., "Data Mining Using Hidden Markov Models (HMM2) to Detect Heterogeneities into Bacteria Genomes." *Journes Ouvertes Biologie, Informatique et Mathmatiques - JOBIM 2005*, July 2005.

[14] FORNEY JR., G. D., "The Viterbi Algorithm: A Personal History." 2005.

[15] HE, Y., "Extended Viterbi algorithm for second-order hidden Markov process." in *Proc. of the IEEE 9th Intl. Conf. on Pattern Recognition*, (Rome, Italy), pp. 718–720, 1988.

[16] HUANG, X., ACERO, A., and HON, H., *Spoken language processing: A guide to theory, algorithm and system development*. Upper Saddle River, New Jersey: Prentice-Hall, 2001.

[17] JELINEK, F., "Continuous speech recognition by statistical methods." in *Proc. of the IEEE*, pp. 532–556, 1976.

[18] JELINEK, F., *Statistical Methods for Speech Recognition*. Massachusetts: The Massachusetts Institute of Technology Press, 1997.

[19] JURAFSKY, D. and MARTIN, J., *Speech and Language Processing - An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Upper Saddle River, New Jersey 07458: Prentice Hall, 2000.

[20] KITTLER, J. and YOUNG, P., "A new approach to feature selection based on the Karhunen-Loeve expansion." *Pattern Recognition*, 1973, Vol. 5, pp. 335–352.

[21] KRIOUILE, A., J.-F., M., and J.-P., H., "Some improvements in speech recognition based on HMM." in *Proc. of the IEEE Int. Conf. on Acoustics*, (Albuquerque, USA), pp. 545–548, 1990.

[22] KROG, A., BROWN, M., MIAN, I., SJOLANDER, K., and HAUSSLER, D., "Hidden markov models in computational biology: Applications to protein modeling." *Journal Molecular Biology*, 1994, Vol. 235, pp. 1501–1531.

[23] MARI, J., FOHR, D., and JUNQUA, J., "A second-order HMM for high-performance word and phoneme-based continuous speech recognition." in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, (Atlanta, USA), pp. 435–438, 1996.

[24] MARI, J., HATON, J.-P., and KRIOUILE, A., "Automatic word recognition based on second-order Markov models." *IEEE Trans. on Speech and Audio Processing*, 1997, Vol. 5, No. 3, No. 3, pp. 22–25.

[25] MOON, T. K., "The Expectation-Maximation Algorithm." *IEEE Signal Processing Magazine*, November 1996, pp. 47–60.

[26] NAG, R., WONG, K., and FALLSIDE, F., "Script recognition using hidden Markov models." in *Proc. Int. Cont. on Acoustics, Speech, and Signal Processing*, pp. 2017–2074, 1986.

[27] NEL, E.-M., DU PREEZ, J., and HERBST, B., "Estimating the Pen Trajectories of Static Signatures Using Hidden Markov Models." *IEEE Trans. Pattern Anal. Mach. Intell.*, 2005, Vol. 27, No. 11, No. 11, pp. 1733–1746.

[28] NEY, H., MERGEL, D., NOLL, A., and PAESELER, A., "Data Driven Search Organisation for Continuous Speech Recognition." *IEEE Trans. on Signal Processing*, February 1992, Vol. 40, No. 2, pp. 272–281.

[29] NGUYEN, L. and SCHWARTZ, R., "Efficient 2-Pass N-Best Decoder." in *Proc. of Eurospeech Conf.*, (Rhodes, Greece), pp. 167–170, September 1997.

[30] NGUYEN, L. and SCHWARTZ, R., "Single-tree method for grammar-directed search." in *Proc. IEEE Int. Conf. on the Acoustics, Speech, and Signal Processing*, (Washington, DC, USA), pp. 613–616, IEEE Computer Society, 1999.

[31] NGUYEN, L., SCHWARTZ, R., KUBALA, F., and PLACEWAY, P., "Search algorithms for software-only real-time recognition with very large vocabularies." in *Proc. of the Workshop on Human Language Technology*, (Morristown, NJ, USA), pp. 91–95, Association for Computational Linguistics, 1993.

[32] NGUYEN, L., SCHWARTZ, R., ZHAO, Y., and ZAVALIAGKOS, G., "Is N-Best Dead?." in *Proc. of the ARPA Workshop on Human Language Technology*, (Merril Lynch Conference Centre), pp. 411–414, 1994.

[33] NILSSON, N., *Principles of Artificial Intelligence*. Springer-Verlag, 1982.

[34] ODELL, J., VALTCHEF, V., WOODLAND, P., and YOUNG, S., "A One Pass Decoder Design For Large Vocabulary Recognition." in *Proc. of the ARPA Workshop on Human Language Technology*, (Merril Lynch Conference Centre), pp. 405–410, 1994.

[35] PAPOULIS, A. and PILLAI, S., *Probability, Random Variables and Stochastic Processes*. New York, USA: McGraw-Hill, 2002.

[36] PAUL, D., "Algorithms for an Optimal A* Search and Linearizing the Search in the Stack Decoder." in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 1, (Toronto, Ontario), pp. 693–696, 1991.

[37] PAUL, D., "An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model." in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 1, (San Francisco, California), pp. 25–28, 1992.

[38] RABINER, L., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proc. of the IEEE*, February 1989, Vol. 77, No. 2, pp. 257–286.

[39] RABINER, L. and JUANG, B., "An Introduction to Hidden Markov Models." *IEEE ASSP Magazine*, January 1986, Vol. 1, No. 3, pp. 4–16.

[40] RABINER, L. and JUANG, B., *Fundamentals of Speech Recognition*. Englewood Cliffs, New Jersey: Prentice-Hall, 1993.

[41] SCHWARTZ, R. and AUSTIN, S., "Efficient, high-performance algorithms for N-Best search." in *Proc. of the Workshop on Speech and Natural Language*, (Morristown, NJ, USA), pp. 6–11, Association for Computational Linguistics, 1990.

[42] SCHWARTZ, R. and CHOW, Y.-L., "The N-best algorithms: an efficient and exact procedure for finding the N most likely sentence hypotheses." in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 81–84, April 1990.

[43] SIXTUS, A. and ORTMANNS, S., "High quality word graphs using forward-backward pruning." in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, (Washington, DC, USA), pp. 593–596, IEEE Computer Society, 1999.

[44] T.H., C., LEISERSON, C., RIVEST, R., , and STEIN, C., *Introduction to Algorithms*. 2nd edition. The MIT Press, 2001.

[45] VITERBI, A., "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." *IEEE Trans. on Information Theory*, 1967, Vol. 13, No. 2, No. 2, pp. 260–269.

[46] VITERBI, A., "A Personal History of the Viterbi Algorithm." *IEEE Signal Processing Magazine*, 2006, Vol. 23, No. 4, No. 4, pp. 120–142.

Appendix A

Equivalence of Right-context HMM

A.1 Evaluation Problem Equivalence

Given the observation sequence $\mathbf{X}_1^T$ and a right-context HMM model Φ_{rc} we can compute the probability that the model generated the observations $P(\mathbf{X}_1^T|\Phi_{\text{rc}})$ as:

$$\begin{aligned} P(\mathbf{X}_1^T|\Phi_{\text{rc}}) &= \sum_{\text{all } \mathbf{s}_1^T} P(\mathbf{X}_1^T, \mathbf{S}_1^T|\Phi_{\text{rc}}) \\ &= \sum_{\text{all } \mathbf{s}_1^T} P(\mathbf{S}_1^T|\Phi_{\text{rc}}) P(\mathbf{X}_1^T|\mathbf{S}_1^T, \Phi_{\text{rc}}) \end{aligned}$$

The proof consists of two parts: the first part is to show that, for a particular state sequence $\mathbf{S}_1^T = (s_1, s_2, \ldots, s_T)$, the state sequence probability of the right-context HMM $P(\mathbf{S}_1^T|\Phi_{\text{rc}})$ is equal to the state sequence probability of the equivalent left-context HMM $P(\mathbf{S}_1^T|\Phi_{\text{lc}})$. The second part of the proof is to show that the for the same state sequence $\mathbf{S}_1^T$ the joint output observation probability of the right-context HMM along the state sequence, $P(\mathbf{X}_1^T|\mathbf{S}_1^T, \Phi_{\text{rc}})$, is equal to the joint output observation probability of the left-context HMM, $P(\mathbf{X}_1^T|\mathbf{S}_1^T, \Phi_{\text{rc}})$, along the same state sequence.

For a particular sequence $\mathbf{S}_1^T = (s_1, s_2, \ldots, s_T)$ the state sequence probability $P(\mathbf{S}_1^T|\Phi_{\text{rc}})$ can be rewritten by applying the right-context Markov assumption:

$$\begin{aligned} P(\mathbf{S}_1^T|\Phi_{\text{rc}}) &= P(s_T|\Phi_{\text{rc}}) P(s_{T-1}|s_T, \Phi_{\text{rc}}) \ldots P(s_{T-R+1}|s_{T-R+2}, \ldots, s_T, \Phi_{\text{rc}}) \\ &\quad \times \prod_{t=T-R}^{1} P(s_t|s_{t+1}, \ldots, s_{t+R}, \Phi_{\text{rc}}) \\ &= \overleftarrow{a}_{s_T(N+1)} \overleftarrow{a}_{s_{(T-1)}s_T} \ldots \overleftarrow{a}_{s_{(T-R+1)}s_{(T-R+2)}\ldots s_T} \prod_{t=T-R}^{1} \overleftarrow{a}_{s_t s_{(t+1)}\ldots s_{(t+R)}} \\ &= \left[\prod_{t=1}^{T-R} \overleftarrow{a}_{s_t s_{(t+1)}\ldots s_{(t+R)}} \right] \overleftarrow{a}_{s_{(T-R+1)}s_{(T-R+2)}\ldots s_T} \ldots \overleftarrow{a}_{s_{(T-1)}s_T} \overleftarrow{a}_{s_T(N+1)} \end{aligned}$$

which we can rewrite using Bayes' Theorem and the joint state probabilities to be:

$$P(\mathbf{S}_1^T|\Phi_{\text{rc}})$$

$$
\begin{aligned}
&= \frac{P(s_1, s_2, \ldots, s_{1+R}|\Phi_{\text{rc}})}{P(s_2, s_3, \ldots, s_{1+R}|\Phi_{\text{rc}})} \frac{P(s_2, s_3, \ldots, s_{2+R}|\Phi_{\text{rc}})}{P(s_3, s_4, \ldots, s_{2+R}|\Phi_{\text{rc}})} \cdots \times \\
&= \frac{P(s_{T-R}, s_{T-R+1}, \ldots, s_T|\Phi_{\text{rc}})}{P(s_{T-R+1}, s_{T-R+2}, \ldots, s_T|\Phi_{\text{rc}})} \frac{P(s_{T-R+1}, s_{T-R+2}, \ldots, s_T|\Phi_{\text{rc}})}{P(s_{T-R+2}, s_{T-R+3}, \ldots, s_T|\Phi_{\text{rc}})} \times \\
&\quad \frac{P(s_{T-R+2}, s_{T-R+3}, \ldots, s_T|\Phi_{\text{rc}})}{P(s_{T-R+3}, s_{T-R+4}, \ldots, s_T|\Phi_{\text{rc}})} \cdots \frac{P(s_{T-2}, s_{T-1}, s_T|\Phi_{\text{rc}})}{P(s_{T-1}, s_T|\Phi_{\text{rc}})} \times \\
&\quad \frac{P(s_{T-1}, s_T|\Phi_{\text{rc}})}{P(s_T|\Phi_{\text{rc}})} P(s_T|\Phi_{\text{rc}}) \\
&= \frac{P(s_1, s_2, \ldots, s_{1+R}|\Phi_{\text{rc}})}{P(s_2, s_3, \ldots, s_{1+R}|\Phi_{\text{rc}})} \frac{P(s_2, s_3, \ldots, s_{2+R}|\Phi_{\text{rc}})}{P(s_3, s_4, \ldots, s_{2+R}|\Phi_{\text{rc}})} \cdots \times \\
&\quad \frac{P(s_{T-R}, s_{T-R+1}, \ldots, s_{T-1}|\Phi_{\text{rc}})}{P(s_{T-R+1}, s_{T-R+2}, \ldots, s_{T-1}|\Phi_{\text{rc}})} P(s_{T-R}, s_{T-R+1}, \ldots, s_T|\Phi_{\text{rc}}) \\
&= P(s_1, s_2, \ldots, s_{1+R}|\Phi_{\text{rc}}) \frac{P(s_2, s_3, \ldots, s_{2+R}|\Phi_{\text{rc}})}{P(s_2, s_3, \ldots, s_{1+R}|\Phi_{\text{rc}})} \frac{P(s_3, s_4, \ldots, s_{3+R}|\Phi_{\text{rc}})}{P(s_3, s_4, \ldots, s_{2+R}|\Phi_{\text{rc}})} \\
&\quad \times \cdots \frac{P(s_{T-R-1}, s_{T-R}, \ldots, s_{T-1}|\Phi_{\text{rc}})}{P(s_{T-R-1}, s_{T-R}, \ldots, s_{T-2}|\Phi_{\text{rc}})} \frac{P(s_{T-R}, s_{T-R+1}, \ldots, s_T|\Phi_{\text{rc}})}{P(s_{T-R}, s_{T-R+1}, \ldots, s_{T-1}|\Phi_{\text{rc}})}
\end{aligned}
$$

If the requirement is made that the joint state probabilities of the right-context HMM is equal to the joint state probabilities of its equivalent left-context HMM, expressed as

$$P(s_t, s_{t+1}, \ldots, s_{t+R}|\Phi_{\text{rc}}) = P(s_t, s_{t+1}, \ldots, s_{t+R}|\Phi_{\text{lc}}) \tag{A.1}$$

then the state sequence probability of the right-context HMM can be written in terms of the parameters of its equivalent left-context HMM as:

$$
\begin{aligned}
&P(\mathbf{S}_1^T|\Phi_{\text{rc}}) \\
&= P(s_1, s_2, \ldots, s_{1+R}|\Phi_{\text{lc}}) \frac{P(s_2, s_3, \ldots, s_{2+R}|\Phi_{\text{lc}})}{P(s_2, s_3, \ldots, s_{1+R}|\Phi_{\text{lc}})} \frac{P(s_3, s_4, \ldots, s_{3+R}|\Phi_{\text{lc}})}{P(s_3, s_4, \ldots, s_{2+R}|\Phi_{\text{lc}})} \\
&\quad \cdots \times \frac{P(s_{T-R-1}, s_{T-R}, \ldots, s_{T-1}|\Phi_{\text{lc}})}{P(s_{T-R-1}, s_{T-R}, \ldots, s_{T-2}|\Phi_{\text{lc}})} \frac{P(s_{T-R}, s_{T-R+1}, \ldots, s_T|\Phi_{\text{lc}})}{P(s_{T-R}, s_{T-R+1}, \ldots, s_{T-1}|\Phi_{\text{lc}})} \\
&= P(s_1|\Phi_{\text{lc}}) P(s_2|s_1, \Phi_{\text{lc}}) P(s_3|s_2, s_1, \Phi_{\text{lc}}) \cdots P(s_{1+R}|s_R, \ldots, s_2, s_1, \Phi_{\text{lc}}) \times \\
&\quad P(s_{2+R}|s_{1+R}, s_R, \ldots, s_2, \Phi_{\text{lc}}) \ldots P(s_T|s_{T-1}, \ldots, s_{T-R+1}, s_{T-R}, \Phi_{\text{lc}}) \\
&= P(s_1|\Phi_{\text{lc}}) P(s_2|s_1, \Phi_{\text{lc}}) P(s_3|s_2, s_1, \Phi_{\text{lc}}) \cdots P(s_{1+R}|s_R, \ldots, s_2, s_1, \Phi_{\text{lc}}) \times \\
&\quad \prod_{t=2+R}^{T} P(s_{t+R}|s_{t+R-1}, s_{t+R-2}, \ldots, s_{t-1}, s_t, \Phi_{\text{lc}}) \\
&= P(\mathbf{S}_1^T|\Phi_{\text{lc}})
\end{aligned}
$$

For the same state sequence $\mathbf{S}_1^T$ the joint output observation probability of the right-context HMM along the state sequence, $P(\mathbf{X}_1^T|\mathbf{S}_1^T, \Phi_{\text{rc}})$, can be rewritten by applying the output independence assumption as:

$$P(\mathbf{X}_1^T|\mathbf{S}_1^T, \Phi_{\text{rc}}) = \prod_{t=1}^{T} P(\mathbf{x}_t|s_t, \Phi_{\text{rc}}) \tag{A.2}$$

If we make the further requirement that the right-context HMM and its equivalent left-context HMM use the same set of probability density functions i.e.

$$\begin{aligned} P(\mathbf{x}_t|s_t,\Phi_{\text{rc}}) &= b_{s_t}(\mathbf{x}_t|\Phi_{\text{rc}}) \\ = b_{s_t}(\mathbf{x}_t|\Phi_{\text{lc}}) &= P(\mathbf{x}_t|s_t,\Phi_{\text{lc}}) \end{aligned}$$

then the joint output observation probability of the right-context HMM along the state sequence $\mathbf{S}_1^T$ is equal to the joint output observation probability of the equivalent left-context HMM as shown by:

$$P(\mathbf{X}_1^T|\mathbf{S}_1^T,\Phi_{\text{rc}}) = \prod_{t=1}^{T} P(\mathbf{x}_t|s_t,\Phi_{\text{rc}}) = \prod_{t=1}^{T} P(\mathbf{x}_t|s_t,\Phi_{\text{lc}}) = P(\mathbf{X}_1^T|\mathbf{S}_1^T,\Phi_{\text{lc}}) \tag{A.3}$$

Thus it is proven that the probability of the right-context HMM Φ_{rc} generating the observation sequence $\mathbf{X}_1^T$ is equivalent to the probability of the left-context HMM Φ_{lc} generating the same observation sequence as shown by:

$$\begin{aligned} P(\mathbf{X}_1^T|\Phi_{\text{rc}}) &= \sum_{\text{all } \mathbf{s}_1^T} P(\mathbf{X}_1^T,\mathbf{S}_1^T|\Phi_{\text{rc}}) \\ &= \sum_{\text{all } \mathbf{s}_1^T} P(\mathbf{S}_1^T|\Phi_{\text{rc}})P(\mathbf{X}_1^T|\mathbf{S}_1^T,\Phi_{\text{rc}}) \\ &= \sum_{\text{all } \mathbf{s}_1^T} P(\mathbf{S}_1^T|\Phi_{\text{lc}})P(\mathbf{X}_1^T|\mathbf{S}_1^T,\Phi_{\text{lc}}) \\ &= \sum_{\text{all } \mathbf{s}_1^T} P(\mathbf{X}_1^T,\mathbf{S}_1^T|\Phi_{\text{lc}}) \\ &= P(\mathbf{X}_1^T|\Phi_{\text{lc}}) \end{aligned}$$

A.2 Decoding Problem Equivalence

We will now prove that the right-context HMM results in the same partial paths as the left-context HMM. This is done by proving that the back pointers of the forward Viterbi-beam algorithm when decoding the left-context HMM is equal to the back pointers of the forward Viterbi-beam algorithm when decoding the equivalent right-context HMM i.e.

$$\overrightarrow{\Psi}^{\delta}_{t}(i_2,i_3,\ldots,i_{R+1}) = \overleftarrow{\Psi}^{\delta}_{t}(i_2,i_3,\ldots,i_{R+1}) \tag{A.4}$$

The partial path equivalence will be proven by first proving through induction that:

$$\begin{aligned} &\overleftarrow{\delta}_{t}(i_2,i_3,\ldots,i_{R+1}) \\ = \; &\frac{\overrightarrow{\delta}_{t+R-1}(i_2,i_3,\ldots,i_{R+1})}{b_{i_3}(\mathbf{x}_{t+1})\ldots b_{i_{R+1}}(\mathbf{x}_{t+R-1})P(s_t=i_2,\ldots,s_{t+R-1}=i_{R+1})} \end{aligned} \tag{A.5}$$

When decoding a left-context HMM, we will refer to the best-path forward probability as the left-context, best-path forward probability and when decoding a right-context HMM we will refer to the right-context, best-path forward probability. The left-context best-path forward probabilities for $t \leq R$ can be written as:

$$\begin{aligned}
\overrightarrow{\delta}_R(i_1, i_2, \ldots, i_R) &= \overrightarrow{\delta}_{R-1}(i_1, i_2, \ldots, i_{R-1})\overrightarrow{a}_{0i_1i_2\cdots i_R} b_{i_R}(\mathbf{x}_R) \\
\overrightarrow{\delta}_{R-1}(i_1, i_2, \ldots, i_{R-1}) &= \overrightarrow{\delta}_{R-2}(i_1, i_2, \ldots, i_{R-2})\overrightarrow{a}_{0i_1i_2\cdots i_{R-1}} b_{i_{R-1}}(\mathbf{x}_{R-1}) \\
&\vdots \\
\overrightarrow{\delta}_2(i_1, i_2) &= \overrightarrow{\delta}_1(i_1)\overrightarrow{a}_{0i_1i_2} b_{i_2}(\mathbf{x}_2) \\
\overrightarrow{\delta}_1(i_1) &= \overrightarrow{a}_{0i_1} b_{i_1}(\mathbf{x}_1)
\end{aligned}$$

and thus the left-context best-path forward probability at time $t = R$ can be rewritten as:

$$\begin{aligned}
\overrightarrow{\delta}_R(i_1, i_2, \ldots, i_R) &= \overrightarrow{a}_{0i_1} b_{i_1}(\mathbf{x}_1)\overrightarrow{a}_{0i_1i_2} b_{i_2}(\mathbf{x}_2) \ldots \overrightarrow{a}_{0i_1i_2\cdots i_R} b_{i_R}(\mathbf{x}_R) \\
&= \overrightarrow{a}_{0i_1}\overrightarrow{a}_{0i_1i_2} \ldots \overrightarrow{a}_{0i_1i_2\cdots i_R} b_{i_1}(\mathbf{x}_1) b_{i_2}(\mathbf{x}_2) \ldots b_{i_R}(\mathbf{x}_R) \\
&= P(s_1 = 0, s_1 = i_1, s_2 = i_2, \ldots, s_R = i_R) \times \\
&\quad b_{i_1}(\mathbf{x}_1) b_{i_2}(\mathbf{x}_2) \ldots b_{i_R}(\mathbf{x}_R)
\end{aligned}$$

The right-context best-path forward probability at time $t = 1$ can be written as:

$$\overleftarrow{\delta}_1(i_1, i_2, \ldots, i_R) = \overleftarrow{a}_{0i_1i_2\cdots i_R} b_{i_1}(\mathbf{x}_1)$$

By applying Bayes' Theorem to the right-context transition probabilities the right-context best-path forward probability can be rewritten as:

$$\begin{aligned}
\overleftarrow{\delta}_1(i_1, i_2, \ldots, i_R) &= \frac{P(s_1 = 0, s_1 = i_1, s_2 = i_2, \ldots, s_R = i_R)}{P(s_1 = i_1, s_2 = i_2, \ldots, s_R = i_R)} b_{i_1}(\mathbf{x}_1) \\
&= \frac{P(s_1 = 0, s_1 = i_1, s_2 = i_2, \ldots, s_R = i_R)}{P(s_1 = i_1, s_2 = i_2, \ldots, s_R = i_R)} b_{i_1}(\mathbf{x}_1) \\
&\quad \times \frac{b_{i_2}(\mathbf{x}_2) \ldots b_{i_R}(\mathbf{x}_R)}{b_{i_2}(\mathbf{x}_2) \ldots b_{i_R}(\mathbf{x}_R)} \\
&= \frac{\overrightarrow{\delta}_R(i_1, i_2, \ldots, i_R)}{b_{i_2}(\mathbf{x}_2) \ldots b_{i_R}(\mathbf{x}_R) P(s_1 = i_1, \ldots, s_{t+R-1} = i_R)}
\end{aligned}$$

The left-context best-path forward probability at time $t = R + 1$ can be written as:

$$\overrightarrow{\delta}_{R+1}(i_2, i_3, \ldots, i_{R+1}) = \max_{i_1} \left[\overrightarrow{\delta}_R(i_1, i_2, \ldots, i_R)\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}} \right] b_{i_{R+1}}(\mathbf{x}_{R+1}) \tag{A.6}$$

thus $\overrightarrow{\delta}_{R+1}(i_2, i_3, \ldots, i_{R+1})$ can be rewritten as:

$$\begin{aligned}
&\overrightarrow{\delta}_{R+1}(i_2, i_3, \ldots, i_{R+1}) \\
&= \max_{i_1} \left[\overrightarrow{a}_{0i_1} b_{i_1}(\mathbf{x}_1)\overrightarrow{a}_{0i_1i_2} b_{i_2}(\mathbf{x}_2) \ldots \overrightarrow{a}_{0i_1i_2\cdots i_R} b_{i_R}(\mathbf{x}_R)\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}} \right] \times
\end{aligned}$$

$$
\begin{aligned}
& b_{i_{R+1}}(\mathbf{x}_{R+1}) \\
= & \max_{i_1} \left[\overrightarrow{a}_{0i_1} \overrightarrow{a}_{0i_1i_2} \ldots \overrightarrow{a}_{0i_1i_2\cdots i_R} b_{i_1}(\mathbf{x}_1) b_{i_2}(\mathbf{x}_2) \ldots b_{i_R}(\mathbf{x}_R) \overrightarrow{a}_{i_1i_2\cdots i_{R+1}} \right] \times \\
& b_{i_{R+1}}(\mathbf{x}_{R+1}) \\
= & \max_{i_1} \left[\begin{array}{l} P(s_{t-R+1}=0, s_{t-R+1}=i_1, s_{t-R+2}=i_2, \ldots, s_t=i_R|\Phi_{\text{lc}}) \times \\ b_{i_1}(\mathbf{x}_1) b_{i_2}(\mathbf{x}_2) \ldots b_{i_R}(\mathbf{x}_R) \overrightarrow{a}_{i_1i_2\cdots i_{R+1}} \end{array} \right] \times \\
& b_{i_{R+1}}(\mathbf{x}_{R+1}) \\
= & \max_{i_1} \left[P(s_{t-R+1}=0, s_{t-R+1}=i_1, \ldots, s_t=i_R|\Phi_{\text{lc}}) b_{i_1}(\mathbf{x}_1) \overrightarrow{a}_{i_1i_2\cdots i_{R+1}} \right] \times \\
& b_{i_2}(\mathbf{x}_2) \ldots b_{i_R}(\mathbf{x}_R) b_{i_{R+1}}(\mathbf{x}_{R+1}) \qquad \text{(A.7)}
\end{aligned}
$$

The right-context best-path forward probability at time $t = 2$ can be written as:

$$
\begin{aligned}
\overleftarrow{\delta}_2(i_2, i_3, \ldots, i_{R+1}) &= \max_{i_1} \left[\overleftarrow{\delta}_1(i_1, i_2, \ldots, i_R) \overleftarrow{a}_{i_1i_2\cdots i_{R+1}} \right] b_{i_2}(\mathbf{x}_2) \\
&= \max_{i_1} \left[\overleftarrow{a}_{0i_1i_2\cdots i_R} b_{i_1}(\mathbf{x}_1) \overleftarrow{a}_{i_1i_2\cdots i_{R+1}} \right] b_{i_2}(\mathbf{x}_2) \qquad \text{(A.8)}
\end{aligned}
$$

By applying Bayes' Theorem to the right-context transition probabilities denoted by $\overleftarrow{a}_{i_1i_2\cdots i_{R+1}}$ the best-path forward probability $\overleftarrow{\delta}_2(i_2, i_3, \ldots, i_{R+1})$ can be rewritten as:

$$
\begin{aligned}
& \overleftarrow{\delta}_2(i_2, i_3, \ldots, i_{R+1}) \\
= & \max_{i_1} \left[\begin{array}{l} \frac{P(s_{t-R+1}=0, s_{t-R+1}=i_1, \ldots, s_{t-1}=i_{R-1}, s_t=i_R)}{P(s_{t-R+1}=i_1, s_{t-R+2}=i_2 \ldots, s_t=i_R)} b_{i_1}(\mathbf{x}_1) \times \\ \frac{P(s_{t-R+1}=i_1, s_{t-R+2}=i_2, \ldots, s_t=i_R, s_{t+1}=i_{R+1})}{P(s_{t-R+2}=i_2, s_{t-R+3}=i_3 \ldots, s_{t+1}=i_{R+1})} \end{array} \right] b_{i_2}(\mathbf{x}_2) \\
= & \max_{i_1} \left[\begin{array}{l} P(s_{t-R+1}=0, s_{t-R+1}=i_1, \ldots, s_{t-1}=i_{R-1}, s_t=i_R) b_{i_1}(\mathbf{x}_1) \times \\ \frac{P(s_{t-R+1}=i_1, s_{t-R+2}=i_2, \ldots, s_{t+1}=i_{R+1})}{P(s_{t-R+1}=i_1, s_{t-R+2}=i_2 \ldots, s_t=i_R)} \end{array} \right] \times \\
& \frac{b_{i_2}(\mathbf{x}_2)}{P(s_{t-R+2}=i_2, s_{t-R+3}=i_3 \ldots, s_{t+1}=i_{R+1})} \\
= & \max_{i_1} \left[P(s_{t-R+1}=0, s_{t-R+1}=i_1, \ldots, s_t=i_R) b_{i_1}(\mathbf{x}_1) \overrightarrow{a}_{i_1i_2\cdots i_{R+1}} \right] \times \\
& \frac{b_{i_2}(\mathbf{x}_2)}{P(s_{t-R+2}=i_2, s_{t-R+3}=i_3 \ldots, s_{t+1}=i_{R+1})} \\
= & \frac{\overrightarrow{\delta}_{R+1}(i_2, i_3, \ldots, i_{R+1})}{b_{i_3}(\mathbf{x}_3) \ldots b_{i_{R+1}}(\mathbf{x}_{R+1}) P(s_t=i_2, s_{t+1}=i_3 \ldots, s_{t+R-1}=i_{R+1})}
\end{aligned}
$$

The left-context best-path forward probability for time $t = k + R$ can be written as:

$$
\overrightarrow{\delta}_{k+R}(i_2, i_3, \ldots, i_{R+1}) = \max_{i_1} \left[\overrightarrow{\delta}_{k+R-1}(i_1, i_2, \ldots, i_R) \overrightarrow{a}_{i_1i_2\cdots i_{R+1}} \right] b_{i_{R+1}}(\mathbf{x}_{k+R}) \qquad \text{(A.9)}
$$

The right-context best-path forward probability at time $t = k + 1$ can be written as:

$$
\overleftarrow{\delta}_{k+1}(i_2, i_3, \ldots, i_{R+1}) = \max_{i_1} \left[\overleftarrow{\delta}_k(i_1, i_2, \ldots, i_R) \overleftarrow{a}_{i_1i_2\cdots i_{R+1}} \right] b_{i_2}(\mathbf{x}_{k+1}) \qquad \text{(A.10)}
$$

Assuming that the right-context best-path forward probability at time $t = k$ can be written as in terms of the left-context best-path forward probability at time $t = k+R-1$ as follows:

$$\overleftarrow{\delta}_k(i_1, i_2, \ldots, i_R) = \frac{\overrightarrow{\delta}_{k+R-1}(i_1, i_2, \ldots, i_R)}{b_{i_2}(\mathbf{x}_{k+1}) \ldots b_{i_R}(\mathbf{x}_{k+R-1}) P(s_{k-R+2} = i_1, \ldots, s_{k+1} = i_R)} \tag{A.11}$$

the right-context best-path forward probability at time $t = k + 1$ can be rewritten as:

$$\begin{aligned}
& \overleftarrow{\delta}_{k+1}(i_2, i_3, \ldots, i_{R+1}) \\
= & \max_{i_1} \left[\begin{array}{l} \frac{\overrightarrow{\delta}_{k+R-1}(i_1, i_2, \ldots, i_R)}{b_{i_2}(\mathbf{x}_{k+1}) \ldots b_{i_R}(\mathbf{x}_{k+R-1}) P(s_{k-R+2}=i_1, s_{k-R+3}=i_2 \ldots, s_{k+1}=i_R)} \times \\ \overleftarrow{a}_{i_1 i_2 \cdots i_{R+1}} \end{array} \right] b_{i_2}(\mathbf{x}_{k+1}) \\
= & \max_{i_1} \left[\begin{array}{l} \frac{\overrightarrow{\delta}_{k+R-1}(i_1, i_2, \ldots, i_R)}{b_{i_2}(\mathbf{x}_{k+1}) \ldots b_{i_R}(\mathbf{x}_{k+R-1}) P(s_{k-R+2}=i_1, s_{k-R+3}=i_2 \ldots, s_{k+1}=i_R)} \times \\ \overleftarrow{a}_{i_1 i_2 \cdots i_{R+1}} \end{array} \right] b_{i_2}(\mathbf{x}_{k+1}) \\
= & \max_{i_1} \left[\begin{array}{l} \frac{\overrightarrow{\delta}_{k+R-1}(i_1, i_2, \ldots, i_R)}{P(s_{k-R+2}=i_1, s_{k-R+3}=i_2 \ldots, s_{k+1}=i_R)} \times \\ \frac{P(s_{k-R+2}=i_1, s_{k-R+3}=i_2 \ldots, s_{k+2}=i_{R+1})}{P(s_{k-R+3}=i_2, s_{k-R+3}=i_2 \ldots, s_{k+1}=i_{R+1})} \end{array} \right] \frac{b_{i_{R+1}}(\mathbf{x}_{k+R})}{b_{i_3}(\mathbf{x}_{k+2}) \ldots b_{i_R}(\mathbf{x}_{k+R-1}) b_{i_{R+1}}(\mathbf{x}_{k+R})} \\
= & \max_{i_1} \left[\overrightarrow{\delta}_{k+R-1}(i_1, i_2, \ldots, i_R) \frac{P(s_{k-R+2}=i_1, s_{k-R+3}=i_2 \ldots, s_{k+2}=i_{R+1})}{P(s_{k-R+2}=i_1, s_{k-R+3}=i_2 \ldots, s_{k+1}=i_R)} \right] \times \\
& \frac{b_{i_{R+1}}(\mathbf{x}_{k+R})}{b_{i_3}(\mathbf{x}_{k+2}) \ldots b_{i_{R+1}}(\mathbf{x}_{k+R}) P(s_{k-R+3} = i_2, s_{k-R+3} = i_2 \ldots, s_{k+1} = i_{R+1})} \\
= & \frac{\max_{i_1} \left[\overrightarrow{\delta}_{k+R-1}(i_1, i_2, \ldots, i_R) \overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}} \right] b_{i_{R+1}}(\mathbf{x}_{k+R})}{b_{i_3}(\mathbf{x}_{k+2}) \ldots b_{i_{R+1}}(\mathbf{x}_{k+R}) P(s_{k-R+3} = i_2, s_{k-R+3} = i_2 \ldots, s_{k+1} = i_{R+1})} \\
= & \frac{\overrightarrow{\delta}_{k+R}(i_2, i_3, \ldots, i_{R+1})}{b_{i_3}(\mathbf{x}_{k+2}) \ldots b_{i_{R+1}}(\mathbf{x}_{k+R}) P(s_{k-R+3} = i_2, s_{k-R+3} = i_2 \ldots, s_{k+1} = i_{R+1})}
\end{aligned} \tag{A.12}$$

Thus we have proven by induction that Eq. A.5 is true for $t = 1, 2, \ldots, T - R$.

For $t = T - R + 1, T - R + 2, \ldots, T$ the right-context, best-path forward probability is equal to

$$\begin{aligned}
& \overleftarrow{\delta}_t(i_{t-T+R+1}, i_{t-T+R+1}, \ldots, i_{R+1}) \\
= & \max_{i_{t-T+R}} \left[\overleftarrow{\delta}_{t-1}(i_{t-T+R}, \ldots, i_{R+1}) \overleftarrow{a}_{i_{t-T+R} \cdots i_{R+1}(N+1)} \right] b_{i_{t-T+R+1}}(\mathbf{x}_t) \\
= & \max_{i_{t-T+R}} \left[\begin{array}{l} \max_{i_{t-T+R-1}} \left[\overleftarrow{\delta}_{t-2}(i_{t-T+R-1}, \ldots, i_{R+1}) \overleftarrow{a}_{i_{t-T+R-1} \cdots i_{R+1}(N+1)} \right] \\ b_{i_{t-T+R}}(\mathbf{x}_{t-1}) \overleftarrow{\delta}_{t-1}(i_{t-T+R}, \ldots, i_{R+1}) \overleftarrow{a}_{i_{t-T+R} \cdots i_{R+1}(N+1)} \end{array} \right] b_{i_{t-T+R+1}}(\mathbf{x}_t) \\
= & \max_{i_2, \ldots, i_{t-T+R}} \left[\begin{array}{l} \overleftarrow{\delta}_{t-R+1}(i_2, \ldots, i_{R+1}) \overleftarrow{a}_{i_2 \cdots i_{R+1}(N+1)} b_{i_3}(\mathbf{x}_{t-R+2}) \ldots \\ \overleftarrow{a}_{i_{t-T+R-1} \cdots i_{R+1}(N+1)} b_{i_{t-T+R}}(\mathbf{x}_{t-1}) \overleftarrow{a}_{i_{t-T+R} \cdots i_{R+1}(N+1)} \end{array} \right] b_{i_{t-T+R+1}}(\mathbf{x}_t) \\
= & \max_{i_2, \ldots, i_{t-T+R}} \left[\begin{array}{l} \frac{\overrightarrow{\delta}_T(i_2, \ldots, i_{R+1})}{b_{i_3}(\mathbf{x}_{T-R+2}) \ldots b_{i_{R+1}}(\mathbf{x}_T) P(s_{T-R+1}=i_2, \ldots, s_T=i_{R+1}|\Phi)} \times \\ \overleftarrow{a}_{i_2 \cdots i_{R+1}(N+1)} b_{i_3}(\mathbf{x}_{t-R+2}) \ldots \\ \overleftarrow{a}_{i_{t-T+R-1} \cdots i_{R+1}(N+1)} b_{i_{t-T+R}}(\mathbf{x}_{t-1}) \overleftarrow{a}_{i_{t-T+R} \cdots i_{R+1}(N+1)} \end{array} \right] b_{i_{t-T+R+1}}(\mathbf{x}_t)
\end{aligned}$$

$$
\begin{aligned}
&= \max_{i_2,\ldots,i_{t-T+R}} \left[\begin{array}{l} \frac{\overrightarrow{\delta}_{T(i_2,\ldots,i_{R+1})}}{b_{i_{t-T+R+2}}(\mathbf{x}_{t+1})\ldots b_{i_{R+1}}(\mathbf{x}_T)P(s_{T-R+1}=i_2,\ldots,s_T=i_{R+1}|\Phi)} \times \\ \overleftarrow{a}_{i_2\cdots i_{R+1}(N+1)} \cdots \overleftarrow{a}_{i_{t-T+R-1}\cdots i_{R+1}(N+1)} \overleftarrow{a}_{i_{t-T+R}\cdots i_{R+1}(N+1)} \end{array} \right] \\
&= \max_{i_2,\ldots,i_{t-T+R}} \left[\begin{array}{l} \frac{\overrightarrow{\delta}_{T(i_2,\ldots,i_{R+1})}}{b_{i_{t-T+R+2}}(\mathbf{x}_{t+1})\ldots b_{i_{R+1}}(\mathbf{x}_T)P(s_{T-R+1}=i_2,\ldots,s_T=i_{R+1}|\Phi)} \times \\ \frac{P(s_{T-R+1}=i_2,\ldots,s_T=i_{R+1},s_T=N+1|\Phi)}{P(s_{t-T+R+1}=i_{t-T+R+1},\ldots,s_T=i_{R+1}s_T=N+1|\Phi)} \end{array} \right] \\
&= \max_{i_2,\ldots,i_{t-T+R}} \left[\frac{\overrightarrow{\delta}_{T(i_2,\ldots,i_{R+1})} \overrightarrow{a}_{i_2\cdots i_{R+1}(N+1)}}{b_{i_{t-T+R+2}}(\mathbf{x}_{t+1})\ldots b_{i_{R+1}}(\mathbf{x}_T)P(s_{t-T+R+1}=i_{t-T+R+1},\ldots,s_T=i_{R+1},s_T=N+1|\Phi)} \right] \\
&= \frac{\max_{i_2,\ldots,i_{t-T+R}} \left[\overrightarrow{\delta}_T(i_2,\ldots,i_{R+1}) \overrightarrow{a}_{i_2\cdots i_{R+1}(N+1)} \right]}{b_{i_{t-T+R+2}}(\mathbf{x}_{t+1})\ldots b_{i_{R+1}}(\mathbf{x}_T)P(s_{t-T+R+1}=i_{t-T+R+1},\ldots,s_T=i_{R+1},s_T=N+1|\Phi)}
\end{aligned}
$$

The left-context back pointer at time $t=1$ given the state sequence $\mathbf{S}_1^R$ is simply the initial state 0. The left-context back pointer at time t given the state sequence S_1^{t+R-1} can be written as:

$$
\overrightarrow{\Psi}^{\delta}_t(i_2,i_3,\ldots,i_{R+1}) = \arg\max_{i_1} \left[\overrightarrow{\delta}_{t+R-1}(i_1,i_2,\ldots,i_R) \overrightarrow{a}_{i_1 i_2\cdots i_{R+1}} \right] \tag{A.13}
$$

The right-context back pointer at time t given the state sequence $\mathbf{S}_1^{t+R-1}$ can be written as:

$$
\overleftarrow{\Psi}^{\delta}_t(i_2,i_3,\ldots,i_{R+1}) = \arg\max_{i_1} \left[\overleftarrow{\delta}_t(i_1,i_2,\ldots,i_R) \overleftarrow{a}_{i_1 i_2\cdots i_{R+1}} \right] \tag{A.14}
$$

By using Equation A.5 the right-context back pointer for $t=1,\ldots,T-R$ can be rewritten as:

$$
\begin{aligned}
&\overleftarrow{\Psi}^{\delta}_t(i_2,i_3,\ldots,i_{R+1}) \\
&= \arg\max_{i_1} \left[\frac{\overrightarrow{\delta}_{t+R-1}(i_1,i_2,\ldots,i_R) \overleftarrow{a}_{i_1 i_2\cdots i_{R+1}}}{b_{i_2}(\mathbf{x}_{t+1})\ldots b_{i_R}(\mathbf{x}_{t+R-1})P(s_{t-R+1}=i_1,\ldots,s_t=i_R)} \right] \\
&= \arg\max_{i_1} \left[\frac{\overrightarrow{\delta}_{t+R-1}(i_1,i_2,\ldots,i_R) \overleftarrow{a}_{i_1 i_2\cdots i_{R+1}}}{P(s_{t-R+1}=i_1,s_{t-R+2}=i_2\ldots,s_t=i_R)} \right] \\
&= \arg\max_{i_1} \left[\frac{\overrightarrow{\delta}_{t+R-1}(i_1,i_2,\ldots,i_R) \overrightarrow{a}_{i_1 i_2\cdots i_{R+1}}}{P(s_{t-R+2}=i_2,s_{t-R+3}=i_3\ldots,s_{t+1}=i_{R+1})} \right] \\
&= \arg\max_{i_1} \left[\overrightarrow{\delta}_{t+R-1}(i_1,i_2,\ldots,i_R) \overrightarrow{a}_{i_1 i_2\cdots i_{R+1}} \right] \\
&= \overrightarrow{\Psi}^{\delta}_t(i_2,i_3,\ldots,i_{R+1})
\end{aligned} \tag{A.15}
$$

The right-context back pointer for $t=T-R+1,\ldots,T-1,T$ can be written as:

$$
\begin{aligned}
&\overleftarrow{\Psi}^{\delta}_t(i_{t-T+R+1},\ldots,i_{R+1}) \\
&= \arg\max_{i_{t-T+R}} \left[\overleftarrow{\delta}_{t+R-1}(i_{t-T+R},\ldots,i_{R+1}) \overleftarrow{a}_{i_{t-T+R}\cdots i_{R+1}(N+1)} \right] \\
&= \arg\max_{i_{t-T+R}} \left[\begin{array}{l} \frac{\max_{i_2,\ldots,i_{t-T+R-1}} \left[\overrightarrow{\delta}_{T(i_2,\ldots,i_{R+1})} \overrightarrow{a}_{i_2\cdots i_{R+1}(N+1)} \right]}{b_{i_{t-T+R+1}}(\mathbf{x}_t)\ldots b_{i_{R+1}}(\mathbf{x}_T)P(s_{t-T+R}=i_{t-T+R},\ldots,s_T=i_{R+1},s_T=N+1|\Phi)} \times \\ \overleftarrow{a}_{i_{t-T+R}\cdots i_{R+1}(N+1)} \end{array} \right]
\end{aligned}
$$

$$
\begin{aligned}
&= \arg\max_{i_{t-T+R}} \left[\frac{\max_{i_2,\ldots,i_{t-T+R-1}} \left[\overrightarrow{\delta}_T(i_2,\ldots,i_{R+1})\, \overrightarrow{a}_{i_2\cdots i_{R+1}(N+1)} \right]}{b_{i_{t-T+R+1}}(\mathbf{x}_t)\ldots b_{i_{R+1}}(\mathbf{x}_T) P(s_{t-T+R+1}=i_{t-T+R+1},\ldots,s_T=i_{R+1},s_T=N+1|\Phi)} \right] \\
&= \arg\max_{i_{t-T+R}} \left[\max_{i_2,\ldots,i_{t-T+R-1}} \left[\overrightarrow{\delta}_T(i_2,\ldots,i_{R+1})\, \overrightarrow{a}_{i_2\cdots i_{R+1}(N+1)} \right] \right] \qquad \text{(A.16)}
\end{aligned}
$$

A.3 Parameter estimation

The parameters of the right-context HMM can be estimate by using the *Baum-Welch* algorithm to gather joint state occupancy counts. The only difference lies in the re-estimation equations for the state transition probabilities. The same state occupancy probability counts that are used to reestimate the left-context transition probabilities are used to estimate the right-context transition probabilities.

Appendix B

A* Admissibility

B.1 Proof of Admissibility of Heuristic Function

We will now show that the $(R-K)$-order, left-context best-path backward probability $\hat{\overrightarrow{\epsilon}}_t(i_{K+1},\ldots,i_R)$, computed using the $(R-K)$-order, left-context pseudo HMM $\hat{\Phi}_{\mathrm{R-K,lc}}$, is an admissible heuristic when using A* search to decode the HMM $\Phi_{\mathrm{R,lc}}$. This is done by proving that

$$\begin{aligned} h\left[n_t(i_1,i_2,\ldots,i_R)\right] &= \hat{\overrightarrow{\epsilon}}_t(i_{K+1},\ldots,i_R) \\ &\geq \overrightarrow{\epsilon}_t(i_1,i_2,\ldots,i_R) \\ &= h^*\left[n_t(i_1,i_2,\ldots,i_R)\right], \text{ with } i_1,\ldots,i_R=0,\ldots,N \end{aligned} \tag{B.1}$$

At time $t=T$, the best-path backward probability of the pseudo HMM $\hat{\Phi}_{\mathrm{R-K,lc}}$ is

$$\hat{\overrightarrow{\epsilon}}_T(i_{K+1},\ldots,i_R) = \hat{\overrightarrow{a}}_{i_{K+1}i_{K+2}\cdots i_R(N+1)}, \text{ with } i_{K+1},\ldots,i_R=0,\ldots,N$$

and the best-path backward probability of the HMM $\Phi_{\mathrm{R,lc}}$ is

$$\overrightarrow{\epsilon}_T(i_1,\ldots,i_R) = \overrightarrow{a}_{i_1i_2\cdots i_Ki_{K+1}\cdots i_R(N+1)}, \text{ with } i_1,\ldots,i_R=0,\ldots,N$$

According to Eq. 3.19, the $(R-K)$-order state transition pseudo probability is defined to be

$$\begin{aligned} \hat{\overrightarrow{a}}_{i_{K+1}i_{K+2}\cdots i_R(N+1)} &= \max_{i_1,\cdots,i_K}\left[\overrightarrow{a}_{i_1i_2\cdots i_R(N+1)}\right] \\ &\geq \overrightarrow{a}_{i_1i_2\cdots i_R(N+1)}, \ \forall\ i_1,\ldots,i_R=0,\ldots,N \end{aligned}$$

and therefore

$$\begin{aligned} \hat{\overrightarrow{\epsilon}}_T(i_{K+1},\ldots,i_R) &= \hat{\overrightarrow{a}}_{i_{K+1}i_{K+2}\cdots i_R(N+1)} \\ &\geq \overrightarrow{a}_{i_1i_2\cdots i_R(N+1)} \\ &= \overrightarrow{\epsilon}_T(i_1,i_2,\ldots,i_R)\ \forall\ i_1=0,1,\ldots,N \end{aligned} \tag{B.2}$$

As the pdfs of the $(R-K)$-order, pseudo HMM and the R^{th}-order HMM are identical (and associated with the same state indexes), according to the definition of the pseudo HMM in Eq. 3.19), we know that

$$\hat{\overrightarrow{a}}_{i_2 i_3 \cdots i_{R+1}} \hat{b}_{i_{R+1}}(\mathbf{x}_{t+1}) \;\; \geq \;\; \overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}} b_{i_{R+1}}(\mathbf{x}_{t+1}) \tag{B.3}$$

We will prove by induction that the $(R-K)$-order, best-path backward probability at time index t is an overestimate of the R^{th}-order, best-path backward probability at time index t:

$$\hat{\overrightarrow{\epsilon}}_t(i_{K+1}, \ldots, i_R) \geq \overrightarrow{\epsilon}_t(i_1, \ldots, i_R) \tag{B.4}$$

by first assuming that Eq. B.4 is true at time-index $t+1$ i.e.

$$\hat{\overrightarrow{\epsilon}}_{t+1}(i_2, \ldots, i_R) \geq \overrightarrow{\epsilon}_{t+1}(i_1, \ldots, i_R) \tag{B.5}$$

The best-path backward probability of the HMM $\Phi_{\text{R,lc}}$ at time arbitrary time index t is

$$\overrightarrow{\epsilon}_t(i_1, \ldots, i_R) = \max_{i_{R+1}} \left[\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}} b_{i_{R+1}}(\mathbf{x}_{t+1}) \overrightarrow{\epsilon}_{t+1}(i_2, i_3, \ldots, i_{R+1}) \right] \tag{B.6}$$

At the arbitrary time index t, with $t = 0, \ldots, T-1$, the best-path backward probability of the pseudo HMM $\hat{\Phi}_{\text{R-K,lc}}$ is

$$\begin{aligned}
\hat{\overrightarrow{\epsilon}}_t(i_{K+1}, \ldots, i_R) &= \max_{i_{R+1}} \left[\hat{\overrightarrow{a}}_{i_{K+1} i_{K+2} \cdots i_{R+1}} \hat{b}_{i_{R+1}}(\mathbf{x}_{t+1}) \hat{\overrightarrow{\epsilon}}_{t+1}(i_{K+2}, \ldots, i_{R+1}) \right] \\
&= \max_{i_{R+1}} \left[\hat{\overrightarrow{a}}_{i_{K+1} i_{K+2} \cdots i_{R+1}} b_{i_{R+1}}(\mathbf{x}_{t+1}) \hat{\overrightarrow{\epsilon}}_{t+1}(i_{K+2}, \ldots, i_{R+1}) \right] \\
&\geq \max_{i_{R+1}} \left[\hat{\overrightarrow{a}}_{i_{K+1} i_{K+2} \cdots i_{R+1}} b_{i_{R+1}}(\mathbf{x}_{t+1}) \overrightarrow{\epsilon}_{t+1}(i_2, \ldots, i_{R+1}) \right] \\
&\geq \max_{i_{R+1}} \left[\overrightarrow{a}_{i_1 i_2 \cdots i_{R+1}} b_{i_{R+1}}(\mathbf{x}_{t+1}) \overrightarrow{\epsilon}_{t+1}(i_2, \ldots, i_{R+1}) \right] \\
&= \overrightarrow{\epsilon}_t(i_1, \ldots, i_R) \; \forall \; i_1, \ldots, i_R = 0, \ldots, N
\end{aligned} \tag{B.7}$$

Therefore, we have proven by induction that the heuristic function is an overestimate of the best-path to the final state i.e.

$$\hat{\overrightarrow{\epsilon}}_t(i_{K+1}, \ldots, i_R) = h\left[n_t(i_1, i_2, \ldots, i_R)\right] \;\; \geq \;\; h^*\left[n_t(i_1, i_2, \ldots, i_R)\right] = \overrightarrow{\epsilon}_t(i_1, \ldots, i_R)$$

B.2 Decoding Problem Equivalence

Suppose we have a R^{th}-order, left-context HMM $\Phi_{\text{R,lc}}$ from which we derive a $R-K$-order, left-context pseudo HMM $\hat{\Phi}_{\text{R-K,lc}}$ and a $R-K$ order, right-context, pseudo HMM $\hat{\Phi}_{\text{R-K,rc}}$. The parameters of $\Phi_{\text{R,lc}}$, $\hat{\Phi}_{\text{R-K,lc}}$ and $\hat{\Phi}_{\text{R-K,rc}}$ are respectively defined in Eqs. (2.4), (3.19)

and (4.42). Although it is possible to prove the decoding equivalence for the general case, the proof is much easier to understand if we prove it for a value of $K = 1$

We now prove that $\hat{\Phi}_{\mathrm{R-1,rc}}$ will result in the same partially decoded paths as $\hat{\Phi}_{\mathrm{R-1,lc}}$. This is done by proving that

$$\overrightarrow{\hat{\Psi}}^{\epsilon}_{t}(i_2, i_3, \ldots, i_R) = \overleftarrow{\hat{\Psi}}^{\epsilon}_{t}(i_2, i_3, \ldots, i_R), \text{ where } i_2, \ldots, i_R = 0, 1, \ldots, N+1 \tag{B.8}$$

which requires that we first prove by induction that

$$\begin{aligned} & \overrightarrow{\hat{\epsilon}}'_{t}(i_2, i_3, \ldots, i_R) \\ = \; & \frac{\overleftarrow{\hat{\epsilon}}_{t-R+2}(i_2, i_3, \ldots, i_R)}{P(s_{t-R+2} = i_2, s_{t-R+3} = i_3, \ldots, s_t = i_R) b_{i_2}(\mathbf{x}_{t-R+2}) \ldots b_{i_{R-1}}(\mathbf{x}_{t-1})} \end{aligned} \tag{B.9}$$

When decoding the lower-order, left-context pseudo-HMM, we will refer to the best-path backward probability as the left-context, best-path backward probability $\overrightarrow{\hat{\epsilon}}_{t}(i_2, i_3, \ldots, i_R)$ and when decoding the lower-order, right-context pseudo-HMM, we will refer to the best-path backward probability as the right-context, best-path backward probability $\overleftarrow{\hat{\epsilon}}_{t}(i_2, i_3, \ldots, i_R)$. The right-context, best-path backward probabilities for time $t \geq T - R + 2$ can be written as:

$$\begin{aligned} \overleftarrow{\hat{\epsilon}}_{T-R+2}(i_2, i_3, \ldots, i_R) &= b_{i_2}(\mathbf{x}_{T-R+2}) \overleftarrow{\hat{a}}_{i_2 i_3 \cdots i_R (N+1)} \overleftarrow{\hat{\epsilon}}_{T-R+3}(i_3, i_4, \ldots, i_R) \\ \overleftarrow{\hat{\epsilon}}_{T-R+3}(i_3, i_4, \ldots, i_R) &= b_{i_3}(\mathbf{x}_{T-R+3}) \overleftarrow{\hat{a}}_{i_3 i_4 \cdots i_R (N+1)} \overleftarrow{\hat{\epsilon}}_{T-R+4}(i_4, i_5, \ldots, i_R) \\ &\vdots \\ \overleftarrow{\hat{\epsilon}}_{T-1}(i_{R-1}, i_R) &= b_{i_{R-1}}(\mathbf{x}_{T-1}) \overleftarrow{\hat{a}}_{i_{R-1} i_R (N+1)} \overleftarrow{\hat{\epsilon}}_{T}(i_R) \\ \overleftarrow{\hat{\epsilon}}_{T}(i_R) &= b_{i_R}(\mathbf{x}_T) \overleftarrow{\hat{a}}_{i_R (N+1)} \end{aligned} \tag{B.10}$$

and thus the right-context, best-path backward probability at time $t = T - R + 2$ can be rewritten as:

$$\begin{aligned} & \overleftarrow{\hat{\epsilon}}_{T-R+2}(i_2, i_3, \ldots, i_R) \\ = \; & b_{i_2}(\mathbf{x}_{T-R+2}) \overleftarrow{\hat{a}}_{i_2 i_3 \cdots i_R (N+1)} b_{i_3}(\mathbf{x}_{T-R+3}) \overleftarrow{\hat{a}}_{i_3 i_4 \cdots i_R (N+1)} b_{i_4}(\mathbf{x}_{T-R+4}) \ldots \times \\ & \overleftarrow{\hat{a}}_{i_{R-1} i_R (N+1)} b_{i_R}(\mathbf{x}_T) \overleftarrow{\hat{a}}_{i_R (N+1)} \\ = \; & \overleftarrow{\hat{a}}_{i_2 i_3 \cdots i_R (N+1)} b_{i_2}(\mathbf{x}_{T-R+3}) b_{i_3}(\mathbf{x}_{T-R+4}) \ldots b_{i_R}(\mathbf{x}_T) \overleftarrow{\hat{a}}_{i_3 i_4 \cdots i_R (N+1)} \times \\ & \ldots \overleftarrow{\hat{a}}_{i_{R-1} i_R (N+1)} \overleftarrow{\hat{a}}_{i_R (N+1)} \end{aligned} \tag{B.11}$$

Using the definition in Eq. (4.42) we make the substitution

$$\overleftarrow{\hat{a}}_{i_2 i_3 \cdots i_R (N+1)} = \overrightarrow{a}_{i_1^* i_2 i_3 \cdots i_R (N+1)} \frac{P(s_{T-R+1} = i_2, s_{T-R+2} = i_3, \ldots, s_{T-1} = i_R)}{P(s_{T-R+2} = i_3, s_{T-R+3} = i_4, \ldots, s_T = N+1)} \tag{B.12}$$

and note that

$$\begin{aligned} & \overleftarrow{\hat{a}}_{i_3 i_4 \cdots i_R (N+1)} \ldots \overleftarrow{\hat{a}}_{i_{R-1} i_R (N+1)} \overleftarrow{\hat{a}}_{i_R (N+1)} \\ = \; & P(s_{T-R+2} = i_3, s_{T-R+3} = i_4, \ldots, s_T = N+1) \end{aligned} \tag{B.13}$$

and therefore obtain:

$$\begin{aligned}
& \overleftarrow{\hat{\epsilon}}_{T-R+2}(i_2, i_3, \ldots, i_R) \\
= \; & \overrightarrow{a}_{i_1^* i_2 i_3 \cdots i_R (N+1)} P(s_{T-R+1} = i_2, s_{T-R+2} = i_3, \ldots, s_{T-1} = i_R) \times \\
& b_{i_2}(\mathbf{x}_{T-R+2}) b_{i_3}(\mathbf{x}_{T-R+3}) \ldots b_{i_R}(\mathbf{x}_T) \qquad \text{(B.14)}
\end{aligned}$$

The left-context, best-path backward probability at time $t = T$ can be written as:

$$\begin{aligned}
& \overrightarrow{\hat{\epsilon}}'_T(i_2, i_3, \ldots, i_R) \\
= \; & b_{i_R}(\mathbf{x}_T) \overrightarrow{\hat{a}}_{i_2 i_3 \cdots i_R (N+1)} \\
= \; & b_{i_R}(\mathbf{x}_T) \overrightarrow{a}_{i_1^* i_2 i_3 \cdots i_R (N+1)} \\
= \; & \frac{\overleftarrow{\hat{\epsilon}}_{T-R+2}(i_2, i_3, \ldots, i_R)}{P(s_{T-R+2} = i_2, s_{T-R+3} = i_3, \ldots, s_T = i_R) b_{i_2}(\mathbf{x}_{T-R+1}) \ldots b_{i_{R-1}}(\mathbf{x}_{T-1})} \qquad \text{(B.15)}
\end{aligned}$$

The left-context, best-path backward probability at time $t = T - 1$ can be written as:

$$\begin{aligned}
\overrightarrow{\hat{\epsilon}}'_{T-1}(i_2, i_3, \ldots, i_R) &= b_{i_R}(\mathbf{x}_{T-1}) \max_{i_{R+1}} \left[\overrightarrow{\hat{a}}_{i_2 i_3 \cdots i_{R+1}} \overrightarrow{\hat{\epsilon}}'_T(i_3, i_4, \ldots, i_{R+1}) \right] \\
&= b_{i_R}(\mathbf{x}_{T-1}) \max_{i_{R+1}} \left[\overrightarrow{a}_{i_1^* i_2 i_3 \cdots i_{R+1}} b_{i_{R+1}}(\mathbf{x}_T) \overrightarrow{a}_{i_2^* i_3 \cdots i_{R+1} (N+1)} \right] \qquad \text{(B.16)}
\end{aligned}$$

The right-context, best-path backward probability at time $t = T - R + 1$ can be written as:

$$\begin{aligned}
& \overleftarrow{\hat{\epsilon}}_{T-R+1}(i_2, i_3, \ldots, i_R) \\
= \; & b_{i_2}(\mathbf{x}_{T-R+1}) \max_{i_{R+1}} \left[\overleftarrow{\hat{a}}_{i_2 i_3 \cdots i_{R+1}} \overleftarrow{\hat{\epsilon}}_{T-R+2}(i_3, i_4, \ldots, i_{R+1}) \right] \\
= \; & b_{i_2}(\mathbf{x}_{T-R+1}) \max_{i_{R+1}} \left[\begin{array}{l} \overleftarrow{\hat{a}}_{i_2 i_3 \cdots i_{R+1}} \overrightarrow{a}_{i_2^* i_3 \cdots i_{R+1} (N+1)} \times \\ P(s_{T-R+2} = i_3, s_{T-R+3} = i_4, \ldots, s_T = i_{R+1}) \times \\ b_{i_3}(\mathbf{x}_{T-R+2}) b_{i_4}(\mathbf{x}_{T-R+3}) \ldots b_{i_{R+1}}(\mathbf{x}_T) \end{array} \right] \\
= \; & b_{i_2}(\mathbf{x}_{T-R+1}) \max_{i_{R+1}} \left[\begin{array}{l} \overrightarrow{\hat{a}}_{i_1^* i_2 \cdots i_{R+1}} \frac{P(s_{T-R+1} = i_2, s_{T-R+2} = i_3, \ldots, s_{T-1} = i_R)}{P(s_{T-R+2} = i_3, s_{T-R+3} = i_4, \ldots, s_T = i_{R+1})} \times \\ \overrightarrow{a}_{i_2^* i_3 \cdots i_{R+1} (N+1)} \times \\ P(s_{T-R+2} = i_3, s_{T-R+3} = i_4, \ldots, s_T = i_{R+1}) \times \\ b_{i_3}(\mathbf{x}_{T-R+2}) b_{i_4}(\mathbf{x}_{T-R+3}) \ldots b_{i_{R+1}}(\mathbf{x}_T) \end{array} \right] \\
= \; & b_{i_2}(\mathbf{x}_{T-R+1}) \max_{i_{R+1}} \left[\begin{array}{l} \overrightarrow{\hat{a}}_{i_1^* i_2 \cdots i_{R+1}} P(s_{T-R+1} = i_2, s_{T-R+2} = i_3, \ldots, s_{T-1} = i_R) \\ \overrightarrow{a}_{i_2^* i_3 \cdots i_{R+1} (N+1)} b_{i_3}(\mathbf{x}_{T-R+2}) \ldots b_{i_{R+1}}(\mathbf{x}_T) \end{array} \right] \\
= \; & b_{i_2}(\mathbf{x}_{T-R+1}) \ldots b_{i_R}(\mathbf{x}_{T-1}) \max_{i_{R+1}} \left[\overrightarrow{\hat{a}}_{i_1^* i_2 \cdots i_{R+1}} b_{i_{R+1}}(\mathbf{x}_T) \overrightarrow{a}_{i_2^* i_3 \cdots i_{R+1} (N+1)} \right] \times \\
& P(s_{T-R+1} = i_2, s_{T-R+2} = i_3, \ldots, s_{T-1} = i_R) \\
= \; & b_{i_2}(\mathbf{x}_{T-R+1}) \ldots b_{i_{R-1}}(\mathbf{x}_{T-2}) \overrightarrow{\hat{\epsilon}}'_{T-1}(i_2, i_3, \ldots, i_R) \times \\
& P(s_{T-R+1} = i_2, s_{T-R+2} = i_3, \ldots, s_{T-1} = i_R) \qquad \text{(B.17)}
\end{aligned}$$

Suppose that the following statement is true:

$$\begin{aligned} & \overrightarrow{\hat{\epsilon}}'_t(i_2, i_3, \ldots, i_R) \\ = & \frac{\overleftarrow{\hat{\epsilon}}_{t-R+2}(i_2, i_3, \ldots, i_R)}{P(s_{t-R+2} = i_2, s_{t-R+3} = i_3, \ldots, s_t = i_R) b_{i_2}(\mathbf{x}_{t-R+2}) \ldots b_{i_{R-1}}(\mathbf{x}_{t-1})} \end{aligned} \tag{B.18}$$

The left-context, best-path backward probability at time $t = t - 1$ can be written as:

$$\begin{aligned} \overrightarrow{\hat{\epsilon}}'_{t-1}(i_2, i_3, \ldots, i_R) &= b_{i_R}(\mathbf{x}_{t-1}) \max_{i_{R+1}} \left[\overrightarrow{\hat{a}}_{i_2 i_3 \cdots i_{R+1}} \overrightarrow{\hat{\epsilon}}_t(i_3, i_4, \ldots, i_{R+1}) \right] \\ &= b_{i_R}(\mathbf{x}_{t-1}) \max_{i_{R+1}} \left[\overrightarrow{a}_{i_1^* i_2 i_3 \cdots i_{R+1}} \overrightarrow{\hat{\epsilon}}_t(i_3, i_4, \ldots, i_{R+1}) \right] \end{aligned} \tag{B.19}$$

The right-context, best-path backward probability at time $t = t - R + 1$ can be written as:

$$\begin{aligned} & \overleftarrow{\hat{\epsilon}}_{t-R+1}(i_2, i_3, \ldots, i_R) \\ = & b_{i_2}(\mathbf{x}_{t-R+1}) \max_{i_{R+1}} \left[\overleftarrow{\hat{a}}_{i_2 i_3 \cdots i_{R+1}} \overleftarrow{\hat{\epsilon}}_{t-R+2}(i_3, i_4, \ldots, i_{R+1}) \right] \\ = & b_{i_2}(\mathbf{x}_{t-R+1}) \max_{i_{R+1}} \left[\begin{array}{l} \overleftarrow{\hat{a}}_{i_1^* i_2 i_3 \cdots i_{R+1}} \frac{P(s_{t-R+1}=i_2, s_{t-R+2}=i_3, \ldots, s_{t-1}=i_R)}{P(s_{t-R+2}=i_3, s_{t-R+3}=i_4, \ldots, s_t=i_{R+1})} \times \\ P(s_{t-R+2} = i_3, s_{t-R+3} = i_4, \ldots, s_t = i_{R+1}) \times \\ b_{i_3}(\mathbf{x}_{t-R+2}) \ldots b_{i_R}(\mathbf{x}_{t-1}) \overleftarrow{\hat{\epsilon}}_t(i_3, i_4, \ldots, i_{R+1}) \end{array} \right] \\ = & b_{i_2}(\mathbf{x}_{t-R+1}) \ldots b_{i_R}(\mathbf{x}_{t-1}) \max_{i_{R+1}} \left[\overrightarrow{\hat{a}}_{i_1^* i_2 i_3 \cdots i_{R+1}} \overrightarrow{\hat{\epsilon}}_t(i_3, i_4, \ldots, i_{R+1}) \right] \times \\ & P(s_{t-R+1} = i_2, s_{t-R+2} = i_3, \ldots, s_{t-1} = i_R) \\ = & b_{i_2}(\mathbf{x}_{t-R+2}) \ldots b_{i_{R-1}}(\mathbf{x}_{t-2}) \overrightarrow{\hat{\epsilon}}'_{t-1}(i_2, i_3, \ldots, i_R) \times \\ & P(s_{t-R+1} = i_2, s_{t-R+2} = i_3, \ldots, s_{t-1} = i_R) \end{aligned} \tag{B.20}$$

Thus we have proven by induction that Eq. (B.9) is true for $t = T, T-1, \ldots, R-1$.

According to Section 4.3.2.2, for $t = R-2, \ldots, 2, 1$ the left-context, best-path backward probability is:

$$\begin{aligned} & \overrightarrow{\hat{\epsilon}}'_t(i_2, \ldots, i_t) \\ = & b_{i_t}(\mathbf{x}_t) \max_{i_{t+1}} \left[\overrightarrow{\hat{a}}_{0 i_1 \cdots i_{t+1}} \overrightarrow{\hat{\epsilon}}'_{t+1}(i_1, i_2, \ldots, i_{t+1}) \right] \\ = & \max_{i_{t+1}} \left[b_{i_t}(\mathbf{x}_t) \overrightarrow{\hat{a}}_{0 i_1 \cdots i_t i_{t+1}} b_{i_{t+1}}(\mathbf{x}_{t+1}) \max_{i_{t+2}} \left[\overrightarrow{\hat{a}}_{0 i_1 \cdots i_{t+1} i_{t+2}} \overrightarrow{\hat{\epsilon}}'_{t+1}(i_1, \ldots, i_{t+1}, i_{t+2}) \right] \right] \\ = & \max_{i_{t+1}, \cdots, i_{R-1}} \left[\begin{array}{l} b_{i_t}(\mathbf{x}_t) \overrightarrow{\hat{a}}_{0 i_1 \cdots i_t i_{t+1}} b_{i_{t+1}}(\mathbf{x}_{t+1}) \overrightarrow{\hat{a}}_{0 i_1 \cdots i_t i_{t+1} i_{t+2}} b_{i_{t+2}}(\mathbf{x}_{t+2}) \ldots \\ b_{i_{R-2}}(\mathbf{x}_{R-2}) \overrightarrow{\hat{a}}_{0 i_1 \cdots i_t i_{t+1} \cdots i_{R-1}} \overrightarrow{\hat{\epsilon}}'_{R-1}(i_1, i_2, \ldots, i_t, i_{t+1}, \ldots, i_{R-1}) \end{array} \right] \\ = & \max_{i_{t+1}, \cdots, i_{R-1}} \left[\begin{array}{l} b_{i_t}(\mathbf{x}_t) \overrightarrow{\hat{a}}_{0 i_1 \cdots i_{t+1}} b_{i_{t+1}}(\mathbf{x}_{t+1}) \overrightarrow{\hat{a}}_{0 i_1 \cdots i_{t+2}} b_{i_{t+2}}(\mathbf{x}_{t+2}) \ldots \\ b_{i_{R-2}}(\mathbf{x}_{R-2}) \overrightarrow{\hat{a}}_{0 i_1 \cdots i_t i_{t+1} \cdots i_{R-1}} \times \\ \frac{\overleftarrow{\hat{\epsilon}}_1(i_1, i_2, \ldots, i_t, i_{t+1}, \ldots, i_{R-1})}{b_{i_1}(\mathbf{x}_1) \ldots b_{i_t}(\mathbf{x}_t) b_{i_{t+1}}(\mathbf{x}_{t+1}) \ldots b_{i_{R-2}}(\mathbf{x}_{R-2}) P(s_1=i_1, \ldots, s_t=i_t, s_{t+1}=i_{t+1}, \ldots, s_{R-1}=i_{R-1} \mid \Phi)} \end{array} \right] \\ = & \max_{i_{t+1}, \cdots, i_{R-1}} \left[\frac{\overrightarrow{\hat{a}}_{0 i_1 \cdots i_t i_{t+1}} \overrightarrow{\hat{a}}_{0 i_1 \cdots i_{t+2}} \ldots \overrightarrow{\hat{a}}_{0 i_1 \cdots i_t i_{t+1} \cdots i_{R-1}} \overleftarrow{\hat{\epsilon}}_1(i_1, i_2, \ldots, i_t, i_{t+1}, \ldots, i_{R-1})}{b_{i_1}(\mathbf{x}_1) \ldots b_{i_{t-1}}(\mathbf{x}_{t-1}) P(s_1=i_1, \ldots, s_t=i_t, i_{t+1}=i_{t+1}, \ldots, s_{R-1}=i_{R-1} \mid \Phi)} \right] \end{aligned}$$

$$
\begin{aligned}
&= \max_{i_{t+1},\cdots,i_{R-1}} \left[\frac{\overrightarrow{a}_{0i_1\cdots i_t i_{t+1}} \overrightarrow{a}_{0i_1\cdots i_{t+2}} \ldots \overrightarrow{a}_{0i_1\cdots i_t i_{t+1}\cdots i_{R-1}} \hat{\overleftarrow{\epsilon}}_1(i_1,i_2,\ldots,i_t,i_{t+1},\ldots,i_{R-1})}{b_{i_1}(\mathbf{x}_1)\ldots b_{i_{t-1}}(\mathbf{x}_{t-1})P(s_1=i_1,\ldots,s_t=i_t,i_{t+1}=i_{t+1},\ldots,s_{R-1}=i_{R-1}|\Phi)} \right] \\
&= \max_{i_{t+1},\cdots,i_{R-1}} \left[\frac{P(s_1=0,s_1=i_1,\ldots,s_t=i_t,s_{t+1}=i_{t+1},\ldots,s_{R-1}=i_{R-1}|\Phi)\hat{\overleftarrow{\epsilon}}_1(i_1,i_2,\ldots,i_t,i_{t+1},\ldots,i_{R-1})}{P(s_1=0,s_1=i_1,\ldots,s_t=i_t|\Phi)b_{i_1}(\mathbf{x}_1)\ldots b_{i_{t-1}}(\mathbf{x}_{t-1})P(s_1=i_1,\ldots,s_t=i_t,s_{t+1}=i_{t+1},\ldots,s_{R-1}=i_{R-1}|\Phi)} \right] \\
&= \max_{i_{t+1},\cdots,i_{R-1}} \left[\frac{\overleftarrow{a}_{0i_1\cdots i_t i_{t+1}\cdots i_{R-1}} \hat{\overleftarrow{\epsilon}}_1(i_1,i_2,\ldots,i_t,i_{t+1},\ldots,i_{R-1})}{b_{i_1}(\mathbf{x}_1)\ldots b_{i_{t-1}}(\mathbf{x}_{t-1})P(s_1=i_1,\ldots,s_t=i_t|\Phi)} \right] \\
&= \frac{\max_{i_{t+1},\cdots,i_{R-1}} \left[\overleftarrow{a}_{0i_1\cdots s_{R-1}} \hat{\overleftarrow{\epsilon}}_1(i_1,i_2,\ldots,i_t,i_{t+1},\ldots,s_{R-1}) \right]}{b_{i_1}(\mathbf{x}_1)\ldots b_{i_{t-1}}(\mathbf{x}_{t-1})P(s_1=i_1,\ldots,s_t=i_t|\Phi)}
\end{aligned}
\tag{B.21}
$$

The left-context, "forward" pointer at time $t = T$ given the state sequence $\mathbf{S}_{T-R+2}^T$ is simply the final state $N+1$. The left-context, "forward" pointer at time t given the state sequence $\mathbf{S}_{t-R+2}^T$ can be written as:

$$
\begin{aligned}
\hat{\overrightarrow{\Psi}}_t^{\epsilon}(i_2,i_3,\ldots,i_R) &= \arg\max_{i_{R+1}} \left[\overrightarrow{a}_{i_2 i_3\cdots i_{R+1}} \hat{\overrightarrow{\epsilon}}'_{t+1}(i_3,i_4,\ldots,i_{R+1}) \right] \\
&= \arg\max_{i_{R+1}} \left[\overrightarrow{a}_{i_1^* i_2 i_3\cdots i_{R+1}} \hat{\overrightarrow{\epsilon}}'_{t+1}(i_3,i_4,\ldots,i_{R+1}) \right]
\end{aligned}
\tag{B.22}
$$

The right-context, "forward" pointer at time T given the state sequence $\mathbf{S}_{T-R+2}^T$ is simply the final state $N+1$. The right-context, "forward" pointer at time $t = T, T-1, \ldots, R-1$ given the state sequence $\mathbf{S}_{t-R+2}^T$ can be written as:

$$
\hat{\overleftarrow{\Psi}}_t^{\epsilon}(i_2,i_3,\ldots,i_R) = \arg\max_{i_{R+1}} \left[\hat{\overleftarrow{a}}_{i_2 i_3\cdots i_{R+1}} \hat{\overleftarrow{\epsilon}}_{t-R+3}(i_3,i_4,\ldots,i_{R+1}) \right]
\tag{B.23}
$$

By using Eq. (B.9) the right-context, "forward" pointer for $t = T, T-1, \ldots, R-1$ can be rewritten as:

$$
\begin{aligned}
\hat{\overleftarrow{\Psi}}_t^{\epsilon}(i_2,i_3,\ldots,i_R) &= \arg\max_{i_{R+1}} \left[\hat{\overleftarrow{a}}_{i_2 i_3\cdots i_{R+1}} \hat{\overleftarrow{\epsilon}}_{t-R+3}(i_3,i_4,\ldots,i_{R+1}) \right] \\
&= \arg\max_{i_{R+1}} \left[\begin{array}{l} \overrightarrow{a}_{i_1^* i_2 i_3\cdots i_{R+1}} \frac{P(s_{t-R+1}=i_2,s_{t-R+2}=i_3,\ldots,s_{t-1}=i_R)}{P(s_{t-R+2}=i_3,s_{t-R+3}=i_4,\ldots,s_t=i_{R+1})} \times \\ P(s_{t-R+1}=i_3,s_{t-R+2}=i_4,\ldots,s_t=i_{R+1}) \times \\ b_{i_3}(\mathbf{x}_{t-R+3})\ldots b_{i_R}(\mathbf{x}_t) \hat{\overrightarrow{\epsilon}}'_{t+1}(i_3,i_4,\ldots,i_{R+1}) \end{array} \right] \\
&= \arg\max_{i_{R+1}} \left[\overrightarrow{a}_{i_1^* i_2 i_3\cdots i_{R+1}} \hat{\overrightarrow{\epsilon}}'_{t+1}(i_3,i_4,\ldots,i_{R+1}) \right] \\
&= \hat{\overrightarrow{\Psi}}_t^{\epsilon}(i_2,i_3,\ldots,i_R)
\end{aligned}
\tag{B.24}
$$

By using Eq. (B.21) the left-context, "forward" pointer for $t = R-2, \ldots, 2, 1$ can be rewritten as:

$$
\begin{aligned}
&\hat{\overrightarrow{\Psi}}_t^{\epsilon}(i_1,i_2,\ldots,i_t) \\
&= \arg\max_{i_{t+1}} \left[\hat{\overrightarrow{a}}_{i_1 i_2\cdots i_{t+1}} \hat{\overrightarrow{\epsilon}}'_{t-R+1}(i_1,i_2,\ldots,i_{t+1}) \right]
\end{aligned}
$$

$$
\begin{aligned}
&= \arg\max_{i_{t+1}} \left[\overrightarrow{\hat{a}}_{0 i_1 \cdots i_{t+1}} \frac{\max_{i_{t+2},\cdots,i_{R-1}} \left[\overleftarrow{\hat{a}}_{0 i_1 \cdots s_{R-1}} \overleftarrow{\hat{\epsilon}}_1(i_1, i_2, \ldots, i_{t+1}, i_{t+2}, \ldots, s_{R-1}) \right]}{b_{i_1}(\mathbf{x}_1) \ldots b_{i_t}(\mathbf{x}_t) P(s_1 = i_1, \ldots, s_{t+1} = i_{t+1} | \Phi)} \right] \\
&= \arg\max_{i_{t+1}} \left[\frac{\max_{i_{t+2},\cdots,i_{R-1}} \left[\overleftarrow{\hat{a}}_{0 i_1 \cdots s_{R-1}} \overleftarrow{\hat{\epsilon}}_1(i_1, i_2, \ldots, i_{t+1}, i_{t+2}, \ldots, s_{R-1}) \right]}{b_{i_1}(\mathbf{x}_1) \ldots b_{i_t}(\mathbf{x}_t) P(s_1 = i_1, \ldots, s_t = i_t | \Phi)} \right] \\
&= \arg\max_{i_{t+1}} \left[\max_{i_{t+2},\cdots,i_{R-1}} \left[\overleftarrow{\hat{a}}_{0 i_1 \cdots s_{R-1}} \overleftarrow{\hat{\epsilon}}_1(i_1, i_2, \ldots, i_{t+1}, i_{t+2}, \ldots, s_{R-1}) \right] \right] \qquad \text{(B.25)}
\end{aligned}
$$

Appendix C

Tables of decoding results

C.1 Expense of determining the heuristic function

C.1.1 Decoding of derived low-order, right-context HMMs

Table C.1: *The computational expense of the Viterbi-MAP-beam decoder during the backward decoding of the derived $R-K$-order, right-context HMMs, with beams set wide enough to decode all segments correctly.*

Order[1] R	$R-K$	C_{tot}	$C_{\text{tot,n}}$ [%]	**time** [s]	**Beam** B
2	1	1,498,473	80.91	281.64	36
3	1	1,055,876	73.73	140.45	29
	2	1,497,585	28.97	762.49	28
4	1	890,071	65.06	119.49	24
	2	1,161,390	27.26	615.29	26
	3	1,945,270	13.79	1,111.33	28
5	1	940,822	69.28	119.96	26
	2	1,203,410	29.32	629.96	27
	3	2,301,432	18.43	1,279.71	32
	4	3,493,832	10.46	2,342.95	33
6	1	941,274	69.52	120.04	26
	2	1,064,792	26.25	579.88	25
	3	2,981,605	24.67	1,546.07	37
	4	4,558,258	14.68	2,938.76	38
	5	5,683,323	7.91	4,583.46	37

[1] $R-K$ is the order of the derived HMM. R is the order of the HMM from which the $R-K$-order HMM is derived. C_{tot} is the total decoding cost and $C_{\text{tot,n}}$ is the normalised total decoding cost.

C.1.2 Decoding of derived low-order, right-context pseudo HMMs

Table C.2: *The computational expense of the Viterbi-MAP-beam decoder during the backward decoding of the derived $R-K$-order, right-context pseudo HMMs, with beams set wide enough to decode all segments correctly.*

Order[2] R	$R-K$	C_{tot}	$C_{\text{tot,n}}$ [%]	**time** [s]	**Beam** B
2	1	1,527,958	82.50	285.27	36
3	1	1,005,315	70.20	137.85	24
	2	1,910,982	36.96	916.60	32
4	1	979,414	71.59	122.79	23
	2	1,262,777	29.64	655.19	26
	3	2,260,267	16.02	1,253.54	30
5	1	877,581	64.62	116.74	19
	2	1,270,029	30.95	649.41	26
	3	2,247,430	17.97	1,241.44	31
	4	3,498,269	10.47	2,326.00	33
6	1	894,352	66.05	117.86	19
	2	1,222,064	30.13	649.33	25
	3	3,011,492	24.92	1,568.34	36
	4	4,656,233	15.00	2,965.03	38
	5	5,266,305	7.33	4,289.06	36

[2] $R-K$ is the order of the derived HMM. R is the order of the HMM from which the $R-K$-order HMM is derived. C_{tot} is the total decoding cost and $C_{\text{tot,n}}$ is the normalised total decoding cost.

C.2 Forward-Backward Search of high-order HMMs

C.2.1 FBS-based decoding vs. Viterbi-beam decoding

Table C.3: *A comparison of the computational expense of the FBS-Viterbi-beam, A* and the base-line Viterbi-MAP-beam decoders, during the forward decoding of high-order left-context HMMs, with beams set wide enough to decode all segments correctly.*

Order[3] R	**Decoder**	C_{tot}	$C_{\text{tot,n}}$ [%]	**time** [s]	B	$R-K$	B_h
1	Viterbi-beam	1,348,668	72.90	138.64	29	-	-
2	Viterbi-beam	2,544,329	49.25	998.92	40	-	-
	A*	1,837,265	35.56	872.63	17	1	34
	FBS-Viterbi	1,781,124	34.48	630.26	14	1	36
3	Viterbi-beam	2,748,162	19.53	1,349.16	33	-	-
	A*	1,941,888	13.75	1,881.25	23	1	24
	A*	2,090,642	14.80	1,194.74	18	2	28
	FBS-Viterbi	1,804,384	12.78	677.58	21	1	29
	FBS-Viterbi	2,028,973	14.37	949.40	17	2	28
4	Viterbi-beam	2,645,305	7.91	1,811.39	29	-	-
	A*	2,936,999	8.79	5,568.22	28	1	23
	A*	2,907,273	8.94	3,969.82	27	2	26
	A*	2,328,170	6.96	1,580.70	15	3	28
	FBS-Viterbi	2,488,942	7.45	1,426.59	26	1	29
	FBS-Viterbi	2,920,063	8.73	1,843.67	27	2	28
	FBS-Viterbi	2,520,640	7.54	1,686.20	17	3	28

[3] R denotes the order of the left-context HMM. $R-K$ denotes the order of the derived HMM. C_{tot} is the total decoding cost and $C_{\text{tot,n}}$ is the normalised total decoding cost. B is the beam-width used in the decoder. B_h is the beam-width used in the heuristic decoder.

C.2.2 The influence of the HMM emitting state size N

Table C.4: *A comparison of the computational expense of the FBS-Viterbi-beam, A* and Viterbi-MAP-beam decoders, during the forward decoding of high-order, $N = 10$ emitting-state left-context HMMs, which use DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*

N	R	**Decoder**	C_{tot}	$C_{\text{tot,n}}$ [%]	**time** [s]	B	$R-K$	B_h
10	1	Viterbi-beam	175,929	88.85	35.01	28	-	-
	2	Viterbi-beam	242,359	49.06	105.61	27	-	-
		A*	176,768	35.78	72.69	10	1	24
		FBS-Viterbi	220,566	44.65	62.04	10	1	25
	3	Viterbi-beam	242,411	22.83	130.31	23	-	-
		A*	192,201	18.10	113.13	12	1	26
		FBS-Viterbi	202,693	19.09	65.77	11	1	24
	4	Viterbi-beam	327,299	16.38	192.07	24	-	-
		A*	266,138	13.32	250.38	17	1	25
		FBS-Viterbi	235,142	11.77	99.58	14	1	23
	5	Viterbi-beam	536,773	15.41	301.43	28	-	-
		A*	320,991	9.21	196.39	14	2	28
		FBS-Viterbi	270,107	7.75	142.58	15	1	27
	6	Viterbi-beam	759,227	12.36	467.38	30	-	-
		A*	427,959	6.97	279.17	14	3	29
		FBS-Viterbi	359,379	5.85	219.04	15	2	29
	7	Viterbi-beam	497,350	4.82	414.85	22	-	-
		A*	507,189	4.92	773.01	21	1	26
		FBS-Viterbi	381,641	3.70	336.76	18	1	24
	8	Viterbi-beam	1,536,345	9.14	1,135.80	35	-	-
		A*	630,322	3.75	925.87	21	2	35
		FBS-Viterbi	557,037	3.31	560.76	22	1	24
	9	Viterbi-beam	2,500,879	9.30	1,832.07	40	-	-
		A*	850,278	3.16	988.80	22	3	36
		FBS-Viterbi	602,495	2.24	733.39	20	2	34

[4] R denotes the order of the left-context HMM. $R-K$ denotes the order of the derived HMM. C_{tot} is the total decoding cost and $C_{\text{tot,n}}$ is the normalised total decoding cost. B is the beam-width used in the decoder. B_h is the beam-width used in the heuristic decoder.

Table C.5: *A comparison of the computational expense of the FBS-Viterbi-beam, A* and Viterbi-MAP-beam decoders, during the forward decoding of high-order, $N = 40$ emitting-state left-context HMMs, which use DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*

N	R	**Decoder**[4]	C_{tot}	$C_{\text{tot,n}}$ [%]	**time** [s]	B	$R-K$	B_h
40	1	Viterbi-beam	1,348,668	72.90	138.64	29	-	-
	2	Viterbi-beam	2,544,329	49.25	998.92	40	-	-
		A*	1,837,265	35.56	872.63	17	1	34
		FBS-Viterbi	1,781,124	34.48	630.26	14	1	36
	3	Viterbi-beam	2,748,162	19.46	1,349.16	33	-	-
		A*	1,941,888	13.75	1,881.25	23	1	24
		FBS-Viterbi	1,804,384	12.78	677.58	21	1	29
	4	Viterbi-beam	2,645,305	7.91	1,811.39	29	-	-
		A*	2,328,170	6.96	1,580.70	15	3	28
		FBS-Viterbi	2,488,942	7.45	1,426.59	26	1	29

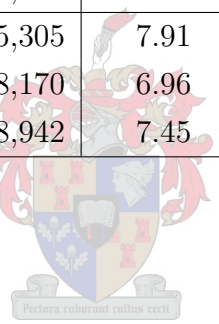


Table C.6: *A comparison of the computational expense of the FBS-Viterbi-beam, A* and Viterbi-MAP-beam decoders, during the forward decoding of high-order, $N = 50$ emitting-state left-context HMMs, which use DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*

N	R	**Decoder**[5]	C_{tot}	$C_{\text{tot,n}}$ [%]	**time** [s]	B	$R-K$	B_h
50	1	Viterbi-beam	1,676,150	65.63	157.41	27	-	-
	2	Viterbi-beam	876,383	12.12	437.43	18	-	-
		A*	1,838,696	25.43	736.24	18	1	21
		FBS-Viterbi	1,595,559	22.07	527.95	20	1	19
	3	Viterbi-beam	1,275,166	6.44	795.51	21	-	-
		A*	2,027,399	10.24	1,715.12	22	2	21
		FBS-Viterbi	1,227,369	6.20	544.46	16	1	16
	4	Viterbi-beam	1,769,351	3.72	1,398.56	23	-	-
		A*	2,478,153	5.21	2,734.90	24	3	17
		FBS-Viterbi	2,081,924	4.38	1,328.02	23	1	18

C.2.3 The influence of the complexity of the output pdfs

Table C.7: *A comparison of the computational expense of the FBS-Viterbi-beam, A* and Viterbi-MAP-beam decoders, during the forward decoding of high-order HMMs with $N = 50$ emitting-states, which use DC-Gaussian, 8-mixture DC-Gaussian and 16-mixture DC-Gaussian state output pdfs, with beams set wide enough to decode all segments correctly.*

R	**Pdf**	**Decoder**	C_{tot}	$C_{\text{tot,n}}$ [%]	**time [s]**	B	$R-K$	B_h
1	DC Gaussian	Viterbi-beam	1,676,150	65.63	157.41	27	-	-
	GMM 8	Viterbi-beam	1,147,181	44.36	276.68	19	-	-
	GMM 16	Viterbi-beam	1,675,814	64.45	430.76	28	-	-
2	DC Gaussian	Viterbi-beam	876,383	12.12	437.43	18	-	-
		A*	1,838,696	25.43	736.24	18	1	21
		FBS-Viterbi	1,595,559	22.07	527.95	20	1	19
	GMM 8	Viterbi-beam	711,437	11.49	654.72	19	-	-
		A*	821,334	13.26	566.02	12	1	19
		FBS-Viterbi	854,731	13.80	819.07	13	1	18
	GMM 16	Viterbi-beam	4,032,319	62.75	1,973.91	43	-	-
		A*	1,749,641	27.23	1,221.68	15	1	54
		FBS-Viterbi	1,788,864	27.84	1,712.40	16	1	53
3	DC Gaussian	Viterbi-beam	1,275,166	6.44	795.51	21	-	-
		A*	2,027,399	10.24	1,715.12	22	2	21
		FBS-Viterbi	1,227,369	6.20	544.46	16	1	16
	GMM 8	Viterbi-beam	886,762	4.82	807.69	20	-	-
		A*	1,130,802	6.14	932.42	17	1	19
		FBS-Viterbi	1,051,967	5.71	1,072.89	17	1	18
	GMM 16	Viterbi-beam	4,499,000	23.62	3,118.82	44	-	-
		A*	2,207,504	11.59	1,959.11	23	1	52
		FBS-Viterbi	2,206,824	11.59	2,210.36	23	1	52
4	DC Gaussian	Viterbi-beam	1,769,351	3.72	1,398.56	23	-	-
		A*	2,478,153	5.21	2,734.90	24	1	17
		FBS-Viterbi	2,081,924	4.38	1,328.02	23	1	18
	GMM 8	Viterbi-beam	1,216,626	2.69	1,474.17	22	-	-
		A*	2,732,023	6.04	6,476.07	30	1	20
		FBS-Viterbi	1,211,580	2.68	1,589.86	18	1	19
	GMM 16	Viterbi-beam	7,233,175	15.61	5,430.67	48	-	-
		A*	2,735,254	5.90	4,464.42	26	1	55
		FBS-Viterbi	2,415,373	5.21	3,012.83	24	1	52

C.2.4 What is the optimal choice of derived HMM order?

Table C.8: *An analysis of the total FBS-Viterbi-beam decoding cost.*

N	R	C_{tot}	C_s [%]	C_h [%]	C_c [%]	**time** [s]	B	$R-K$	B_h
10	2	220,556	33.40	56.57	8.02	62.04	10	1	25
10	3	202,693	42.09	49.38	8.52	65.77	11	1	24
		298,447	29.62	58.78	11.60	101.35	12	2	26
10	4	235,142	53.30	39.36	7.34	99.58	14	1	23
		266,481	44.38	47.46	11.16	113.79	13	2	22
		271,240	27.52	62.42	10.05	134.37	9	3	21
10	5	270,107	58.53	35.12	6.34	142.58	15	1	27
		329,487	41.72	47.50	10.78	156.70	14	2	28
		412,063	26.50	62.43	11.07	178.67	12	3	30
		479,095	15.01	75.46	9.53	228.48	8	4	28
10	6	373,892	70.75	24.47	4.78	249.44	19	1	26
		359,379	47.38	42.52	10.09	219.04	15	2	29
		403,672	36.67	52.59	10.75	237.68	14	3	27
		512,290	28.21	60.55	11.24	297.20	14	4	27
		577,446	14.63	75.49	9.88	368.44	9	5	27
10	7	381,641	72.13	23.26	4.61	336.76	18	1	24
		472,413	61.31	30.48	8.21	368.42	19	2	27
		591,509	38.26	51.30	10.44	365.28	17	3	37
		627,901	31.80	57.13	11.06	406.59	16	4	30
		738,245	23.94	64.43	11.63	528.76	15	5	29
		1,041,606	20.54	67.04	12.42	713.23	17	6	30
10	8	557,037	80.81	15.92	3.26	560.76	22	1	24
		651,364	65.27	27.60	7.13	568.75	22	2	37
		631,732	43.12	47.04	9.84	509.39	18	3	37
		929,295	36.63	50.75	10.62	652.31	21	4	37
		1,221,225	31.66	56.48	11.86	811.96	22	5	37
		1,388,975	18.51	69.99	11.50	916.19	18	6	37
		1,568,365	7.48	84.36	8.16	1,043.56	11	7	37
10	9	785,182	86.53	11.10	2.37	963.50	25	1	23
		602,495	64.81	28.13	7.06	733.39	20	2	34
		715,641	52.97	38.24	8.79	782.33	20	3	34
		1,041,459	38.83	53.18	9.99	859.62	21	4	41
		1,323,015	33.92	54.85	11.22	1,028.76	22	5	38
		2,665,955	11.62	77.35	1.02	2,070.15	19	8	41

Table C.9: *An analysis of the total A* decoding cost.*

N	R	C_{tot}	C_s [%]	C_h [%]	C_c [%]	**time** [s]	B	$R-K$	B_h
10	2	176,768	30.55	59.42	10.02	72.69	10	1	24
10	3	192,201	38.07	52.95	8.98	113.13	12	1	26
		229,000	19.82	68.07	12.11	104.66	9	2	25
10	4	266,138	57.11	36.04	6.85	250.38	17	1	25
		268,737	30.92	56.61	12.47	164.74	12	2	25
		278,380	21.91	66.99	11.10	185.76	10	3	22
10	5	411,422	72.14	23.31	4.55	401.30	22	1	27
		320,991	38.83	49.39	11.78	196.39	14	2	28
		344,444	26.25	62.12	11.63	204.91	12	3	25
		485,252	13.69	75.73	10.58	250.58	10	4	28
10	6	579,540	80.13	16.58	3.29	706.43	25	1	27
		430,324	52.14	37.92	9.93	378.17	18	2	29
		427,959	30.81	57.47	11.72	279.17	14	3	29
		520,445	26.52	61.56	11.92	352.56	15	4	27
		543,817	10.03	80.32	9.65	370.89	8	5	27
10	7	507,189	77.50	18.84	3.66	773.01	21	1	26
		519,902	62.41	29.24	8.35	637.43	20	2	26
		617,213	34.94	54.14	10.92	452.36	17	3	39
		650,775	31.03	57.36	11.61	511.06	17	4	30
		641,760	14.33	75.00	10.67	469.16	11	5	29
		868,913	10.15	79.79	10.06	662.83	11	6	30
10	8	704,128	83.75	15.59	2.66	1,263.92	24	1	26
		630,322	63.25	29.08	7.67	925.87	21	2	35
		756,447	48.52	41.63	9.85	705.94	21	3	37
		1,026,979	41.01	48.08	10.91	846.47	23	4	37
		957,342	15.51	74.08	10.40	556.75	14	5	37
		1,244,891	11.37	78.57	10.06	734.68	14	6	37
		1,550,039	6.78	84.59	8.63	1,008.33	12	7	37
10	9	1,266,676	90.97	7.50	1.53	2,433.41	30	1	25
		1,288,352	82.01	13.59	4.40	2,192.54	30	2	32
		850,278	54.98	36.11	8.91	988.80	22	3	36
		927,048	38.18	51.30	10.52	852.62	20	4	36
		1,135,832	26.74	61.99	11.27	890.14	19	5	37
		1,277,383	20.46	67.86	11.68	996.87	18	6	35
		1,783,326	14.01	74.78	11.21	1,394.17	18	7	38
		2,062,004	7.23	83.70	9.07	1,660.68	14	8	38

C.2.5 The computational consistency of the FBS-based decoders

Table C.10: *The computational consistency of decoding high-oder HMMS with $N = 10$ first-order emitting states with respectively the Viterbi-MAP-beam, A* and FBS-Viterbi-beam decoder.*

		Average		**Std. dev.**		**Minimum**		**Maximum**	
R	**Decoder**	$C_{\text{tot,n}}$	**t [s]**	$C_{\text{tot,n}}$	**t [s]**	$C_{\text{tot,n}}$	**t [s]**	$C_{\text{tot,n}}$	**t [s]**
1	Viterbi-beam	88.47	62.08	3.29	12.81	76.41	25.00	97.41	100.00
2	Viterbi-beam	48.96	68.09	4.86	6.66	14.72	18.18	71.02	100.00
	A*	35.52	46.87	2.03	7.08	21.44	18.18	41.65	72.73
	FBS-Viterbi	44.30	40.00	2.21	4.67	24.94	18.18	50.46	45.45
3	Viterbi-beam	22.75	61.61	3.04	7.69	4.67	13.33	37.38	100.00
	A*	18.47	36.74	1.26	5.25	9.40	6.67	22.22	53.33
	FBS-Viterbi	18.96	31.10	1.18	3.83	8.21	13.33	22.25	40.00
4	Viterbi-beam	16.31	61.92	2.56	8.44	2.17	13.64	30.31	100.00
	A*	13.24	80.72	1.45	13.00	3.75	13.64	19.49	136.36
	FBS-Viterbi	11.69	32.10	1.05	3.31	3.20	13.64	14.83	40.91
5	Viterbi-beam	15.45	56.26	2.69	7.66	1.79	13.16	32.54	100.00
	A*	9.20	36.65	1.21	7.33	1.37	13.16	30.75	226.32
	FBS-Viterbi	7.72	26.61	0.85	3.41	2.28	13.16	18.98	102.63
6	Viterbi-beam	12.44	52.62	2.38	7.66	1.16	12.70	28.50	100.00
	A*	6.96	28.48	1.06	5.66	0.79	12.70	20.27	152.38
	FBS-Viterbi	5.84	24.66	0.70	2.36	1.10	14.29	9.57	55.56
7	Viterbi-beam	4.86	30.65	1.04	4.57	0.48	10.42	14.04	100.00
	A*	4.96	57.11	1.00	14.09	0.64	10.42	23.15	361.46
	FBS-Viterbi	3.71	24.88	0.59	3.03	0.52	13.54	10.64	92.71
8	Viterbi-beam	9.27	45.00	2.11	7.70	0.76	11.17	25.49	100.00
	A*	3.79	36.68	0.77	10.56	0.31	8.94	15.70	221.79
	FBS-Viterbi	3.34	22.22	0.65	3.13	0.41	11.17	9.16	77.65
9	Viterbi-beam	9.47	42.74	2.26	7.61	0.62	10.53	27.63	100.00
	A*	3.20	23.07	0.58	5.02	0.28	7.57	9.16	102.96
	FBS-Viterbi	2.26	17.11	0.39	2.27	0.27	9.88	5.67	59.88