Computing All $\ell$-Cover Automata Fast*

Artur Jeż$^1$,** and Andreas Maletti$^2$,***

$^1$ Institute of Computer Science, University of Wrocław
ul. Joliot-Curie 15, 50–383 Wrocław, Poland
aje@cs.uni.wroc.pl

$^2$ Institute for Natural Language Processing, Universität Stuttgart
Azenbergstraße 12, 70174 Stuttgart, Germany
andreas.maletti@ims.uni-stuttgart.de

Abstract. Given a language $L$ and a number $\ell$, an $\ell$-cover automaton for $L$ is a DFA $M$ such that its language coincides with $L$ on all words of length at most $\ell$. It is known that an equivalent minimal $\ell$-cover automaton can be constructed in time $O(n \log n)$, where $n$ is the number of states of $M$. This is achieved by a clever and sophisticated variant of Hopcroft’s algorithm, which computes the $\ell$-similarity inside the main algorithm. This contribution presents an alternative simple algorithm with running time $O(n \log n)$, in which the computation is split into three phases. First, a compact representation of the gap table is created. Second, this representation is enriched with information about the length of a shortest word leading to the states. These two steps are independent of the parameter $\ell$. Third, the $\ell$-similarity is extracted by simple comparisons against $\ell$. In particular, this approach allows the calculation of all the sizes of minimal $\ell$-cover automata (for all valid $\ell$) in the same time bound.

1 Introduction

Deterministic finite automata (DFA) are widely used in computer science due to their simplicity and flexibility. Their minimisation is one of the oldest problems that is motivated both theoretically and practically and almost every DFA toolkit implements it. More precisely, the DFA minimisation problem asks for a smallest DFA that recognises the same language as a given input DFA $M$. The asymptotically best solution is due to Hopcroft [9,7], who presented an $O(n \log n)$ algorithm where $n$ is the number of states of $M$. Whether an asymptotically faster algorithm exists, remains one of the most challenging open questions in the area. In many applications the desired language $L$ is finite. It was

* This work was done when A. Maletti was visiting Wrocław University thanks to the support of the “Visiting Professors” programme of the Municipality of Wrocław.

** Supported by the MNiSW grant N206 492638 2010–2012 and by the Young Researcher scholarship of University of Wrocław.

*** Supported by the German Research Foundation (DFG) grant MA/4959/1-1.
observed in [3] that membership of a word \( w \) in \( L \) can then be decided by:

(i) checking whether \( w \) is short (i.e., \(|w| \leq \ell \) where \( \ell = \max \{|u| : u \in L\} \)) and
(ii) checking it with a DFA \( M \). This allows \( M \) to accept words that are longer than \( \ell \), which yields that \( M \) need not recognise \( L \). Thus, we arrive at the notion of ‘cover automata’. We say that a DFA \( M \) is a deterministic finite cover automaton (DFCA or cover automaton) for a finite language \( L \) if \( L(M) \cap \Sigma^{\leq \ell} = L \),

where \( \ell = \max\{|u| : u \in L\} \) and \( \Sigma^{\leq \ell} \) contains all words of length at most \( \ell \). It is a minimal DFCA for \( L \) if no DFCA for \( L \) has (strictly) fewer states.

It is well-known that the minimal DFCA for \( L \) can be substantially smaller than the minimal DFA for \( L \). Already [3] present a DFA minimisation algorithm that runs in time \( O(n^2 \cdot \ell^2) \). It also allowed the input language to be presented as a DFA \( M \), which could potentially recognise an infinite language. In that case, an explicit word length \( \ell \) needs to be supplied. An \( \ell \)-DFCA for \( M \) is simply a DFCA for \( L(M) \cap \Sigma^{\leq \ell} \). C˘ampeanu et al. [2] improved the minimisation algorithm for finite languages to \( O(n^2) \). Their algorithm can be trivially extended to arbitrary DFA, but it then runs in time \( O(n^2 \cdot \ell^2) \). The currently fastest algorithm for DFCA minimisation is due to Körner [12], who developed an algorithm that runs in time \( O(n \log n) \), and is a clever and refined modification of Hopcroft’s algorithm for DFA minimisation.

Minimal DFCA are theoretically characterised [3,12,4]. All known algorithms for constructing a minimal \( \ell \)-DFCA are based on a similarity relation \( \sim \) on states, which is defined such that a minimal \( \ell \)-DFCA consists of pairwise dissimilar states. The relation \( \sim \) is defined using two very basic notions: (i) the level of a state, which is the length of a shortest word leading to it, and (ii) the gap between two states, which is the length of a shortest word on which they differ.

Lossy compression of DFA has received some attention recently, and DFCA minimisation can be considered as an instance. Hyper-minimisation [1] is another instance and aims to find a smallest DFA \( N \) for a given DFA \( M \) such that \( L(M) \) and \( L(N) \) have finite symmetric difference. This notion was refined to \( \ell \)-minimisation [5], where the languages are allowed to differ only on words of length at most \( \ell \). Yet another variant was proposed by Schewe [13].

It is noteworthy that \( \ell \)-minimisation and \( \ell \)-DFCA minimisation are dual. It was already observed by Badr et al. [1] that there are languages \( L \), which are best represented by a pair consisting of an \( \ell \)-minimal automaton (that makes errors on words of length at most \( \ell \)) and a minimal \( \ell \)-DFCA. This combination can be substantially smaller than a single minimal DFA for \( L \). An input word \( w \) is processed by such a pair by selecting the authoritative DFA based on the word’s length.

In principle, this approach works for all possible values of \( \ell \). Thus, it is desirable to construct an algorithm that decides for which value of \( \ell \) the size of the representation is minimal. For this, we need to have algorithms that for a given DFA \( M \) return the size of an \( \ell \)-minimal DFA and a minimal \( \ell \)-DFCA for several values \( \ell \). We note that such an algorithm is known for \( \ell \)-minimal DFA [6], and the current contribution adds the algorithm for minimal \( \ell \)-DFCA.

In this paper, we give an alternative \( \ell \)-DFCA minimisation algorithm, which proceeds in three phases. First, we calculate the function ‘gap’ and represent it