Abstract: Natarajan reduced the problem of designing a certain type of mechanical parts orienter to that of finding reset sequences for monotonic deterministic finite automata. He gave algorithms that in polynomial time either find such sequences or prove that no such sequence exists. In this paper we present a new algorithm based on breadth first search that runs in faster asymptotic time than Natarajan's algorithms, and in addition finds the shortest possible reset sequence if such a sequence exists. We give tight bounds on the length of the minimum reset sequence. We further improve the time and space bounds of another algorithm given by Natarajan, which finds reset sequences for arbitrary deterministic finite automata when all states are initially possible.

Introduction

Natarajan [5] considered the design of automated parts orienters; that is, devices which accept mechanical parts in any orientation or in a wide class of orientations, and output them in some predetermined orientation. One such orienter is a pan handler, in which the part slides around on a tray as that tray is tilted, turning in a well-defined way when it hits the walls of the tray. These devices had been previously been described in [2] and [4].

For a given tray and object, and for a given set of possible initial orientations for the object, one has the problem of determining whether there is a sequence of tilt angles that will cause the object to always end up in the same orientation. Natarajan made the assumptions that the set of angles is finite, that the set of orientations in which the part can rest on a tray face is also finite, that tilting the tray with a given angle and with the object in a given initial orientation always results in the same final orientation, and that this relation between angles, initial orientations, and final orientations is known. With these assumptions he reduced the problem to the following combinatorial one.

One is given a deterministic finite automaton $(S, E)$. $S = \{s_1, s_2, \ldots, s_n\}$ is the set of the states of the automaton, corresponding to orientations of the part to be oriented. $E = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ is the set of the transition functions of the automaton, which we also identify with the input alphabet; these functions correspond to the angles at which the pan may be tilted. One is further given a set of initial states, or orientations, $X \subseteq S$.

In what follows, sequences of input symbols to the automaton will be denoted using the letter $r$. The effect of the input sequence $r$ on the states of the automaton is given by the composition of the transition functions for each input symbol of $r$; as with the input symbols themselves, we identify $r$ with its effect as a transition function. Note that, if $r = \tau_1 \tau_2$, then $r(s) = \tau_2(\tau_1(s))$. If $r$ is the empty input sequence, $r(s) = s$. We denote the set of all possible input sequences by $E^*$.

The problem to which Natarajan reduced the pan handler problem is to find an input sequence $r \in E^*$ such that $|r(X)| = 1$; that is, such that the application of $r$ will leave the automaton in one particular state no matter which state in $X$ it started at. We call $r$ a reset sequence for $(S, E)$ and $X$.

Natarajan gave an algorithm for finding a reset sequence when $X = S$. This algorithm takes $O(n^4)$ time. The sequence produced is not guaranteed to be the shortest possible, but Natarajan bounded its length by $O(kn^3)$. As one of the results of this paper, we improve this algorithm to take time $O(n^3 + kn^2)$, and working space bounded by $O(n^2)$. We also prove a tighter bound of $O(n^3)$.
on the length of the resulting sequence, and show that finding the minimum length reset sequence is NP-complete.

It turns out that for general automata and general $X$, finding a reset sequence is PSPACE-complete. However Natarajan observed that the automata arising in the pan handler problem have a property which he called monotonicity, and that with this property the problem becomes solvable in polynomial time. He gave algorithms with asymptotic time complexity $O(kn^4)$ (or $O(kn^3 \log n)$ when $X = S$), which find sequences of length at most $O(kn^2)$ (respectively $O(kn^2 \log n)$). The sequences found are again not guaranteed to be optimal.

This paper presents a new algorithm for finding reset sequences on monotonic automata, which takes time $O(kn^2)$ and is guaranteed to find the shortest possible sequence. Further, this leads to tight worst case bounds of $n^2 - 2n + 1$ on the number of input symbols in the optimal reset sequence. The algorithm works by defining a new automaton, the states of which correspond to intervals in the cyclic order of the original automaton's states. Reset sequences in the original automaton correspond to paths in the new automaton leading to a singleton interval. Therefore we can find our desired sequence using a simple breadth first search technique.

Definitions and Lemmas

First, we define monotonic automata. Assume that the states of a given deterministic finite automaton (DFA) are arranged in some known cyclic order $s_1, s_2, \ldots, s_n$. A transition function $\sigma$ is monotonic if it preserves the cyclic order of the states. Formally, the sequence of states $\sigma(s_1), \sigma(s_2), \ldots, \sigma(s_n)$, after removal of possible adjacent duplicate states, must be a subsequence of a cyclic permutation of $s_1, s_2, \ldots, s_n$. If a set of transition functions is monotonic, then all compositions of those transition functions will also be monotonic.

We call an automaton monotonic when all of its transition functions are monotonic. From now on when we refer to the automaton $(S, \Sigma)$ we will assume that it is monotonic.

Next let us define an interval $[s_i, s_j]$. This consists of all those states between $s_i$ and $s_j$ (inclusive) in the cyclic order of the states; e.g., $[s_1, s_3] = \{s_1, s_2, s_3\}$. Note that there are $n$ different ways of representing the full set of states $S$ as an interval $[s_i, s_i-1]$; any other set of states that can be represented as an interval has exactly one such representation. We say that an interval $[s_h, s_i]$ is contained in another interval $[s_g, s_j]$ when the endpoints of the intervals appear in the cyclic order $s_g, s_h, s_i, s_j$. Containment as an interval implies containment as a set of states, but the reverse may be false in the case that the containing interval is all of $S$.

Lemma 1. For all $\tau \in \Sigma^*$, and for any interval $I$, $\tau^{-1}(I)$ is an interval.

Proof: If not, there would be $s_{i_1}, s_{i_2}, s_{i_3},$ and $s_{i_4}$ in cyclic order such that $\tau(s_{i_1})$ and $\tau(s_{i_3})$ are in $I$ but $\tau(s_{i_2})$ and $\tau(s_{i_4})$ are not; but this violates monotonicity.

Unlike their inverses, the transition functions of the DFA do not necessarily take intervals to intervals. However we can define a new set of transition functions, corresponding to the original ones, that do take intervals to intervals. For $\sigma \in \Sigma$, let $\sigma'([s_i, s_j])$ be (1) $[\sigma(s_i), \sigma(s_j)]$ if $\sigma(s_i) \neq \sigma(s_j)$; (2) if this is a singleton; i.e., if $\sigma$ maps the whole interval to one state; and (3) undefined otherwise.

The new transition functions we have defined give us a new DFA $(S \times S, \Sigma')$ whose states are the intervals of the original automaton, and which takes the same input alphabet as the original automaton. Note that this DFA, which is of size $O(kn^2)$, can be constructed in time linear in its size. The only complication is how to determine whether the result of a transition in which the two endpoints are mapped to a single point should be that point or undefined. This can be done by first constructing for each $\sigma \in \Sigma$ and $s \in S$ the interval $\sigma^{-1}(s)$, which must exist by lemma 1. This construction takes time $O(n)$, and there are $O(kn)$ intervals to construct, so all such intervals can be constructed