Theoretical Computer Science 4 (1977) 171–197.

ON THE PAGING-COMPLEXITY OF PERIODIC ARRANGEMENTS

Hans-Georg STORK

Institut für Informatik der Universität Stuttgart, Azenbergstrasse 12, D–7000, Stuttgart 1, West Germany

Communicated by Erwin Engeler
Received November 1976

Abstract. A simple model of a paging-system is investigated. The importance of the concept of pagination is stressed. The page-reference-strings under consideration stem from paginating simple loops, i.e. periodic sequences of consecutively accessed addresses. These reference strings are called periodic arrangements. The main objectives are: firstly, to prove a locality principle for this type of program structure: Given a pagination, generate as many references as possible to the same page in order to minimize page-traffic! Secondly, to show that this optimizing principle only holds for certain types of paging-algorithms, like Belady's algorithm and FIFO. It does no longer hold for the widely used paging-algorithm LRU, for instance.

0. Introduction

Paging-algorithms — i.e. algorithms which govern the exchange of code/data-blocks (pages) between main memory and auxiliary storage devices in multilevel computer systems — have been widely investigated in the past (cf. [1, 2, 3]). The main objective in the design of paging-algorithms is to minimize page-traffic, i.e. the relative frequency of swapping pages between main- and auxiliary memory. Suppose we have the following situation:

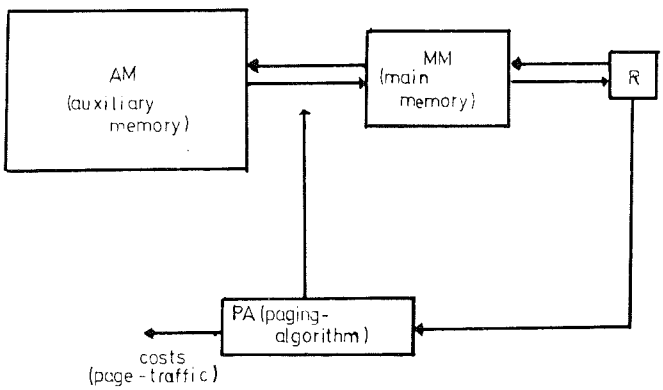

Fig. 1.

Here, in Fig. 1, AM is some auxiliary memory device whose capacity (in number of pages) may be considered infinite. Initially AM contains the whole program and

its data, both written on pages whose number exceeds a given $m \in \mathbf{N}$. The processor R of our model system however, has direct access to MM only, the main-memory whose capacity is m pages. PA, the paging-algorithm, receives information from R if R needs some page that is not yet in MM. PA then takes care that the missing page is brought from AM to MM. In general some other page has to be shifted from MM to AM in this case. These exchanges are time-consuming and hence costly. The less often these swappings occur, the better.

Theoretical work on paging-algorithms usually cuts out the following part of Fig. 1:

Fig. 2.

Then R may be modelled according to some probabilistic law, e.g. R produces page-reference-strings $x_1 \cdots x_t, \ldots$, where the x_t, $t \geq 1$, are random variables (with values in the set of pages) forming a Markov-chain. The performance of a given PA with respect to a given class of input processes can be analyzed. One may even find optimal algorithms with respect to a class of input processes. An extensive account of this approach can be found in [3].

The model of Fig. 1 and hence its restriction represented by Fig. 2 however, suffer from the following disadvantage: in reality R does not produce *page*-reference-strings in the first place, but rather *address*-references-strings. Hence, either by

1. changing the way addresses are combined to form pages (i.e. by changing the pagination) or by

2. changing the sequencing of addresses as far as this is allowed (in array-operations for instance, cf. [5]),

we will obtain entirely different page-reference-strings though the same computations are performed. We shall therefore extend Fig. 1 as follows:

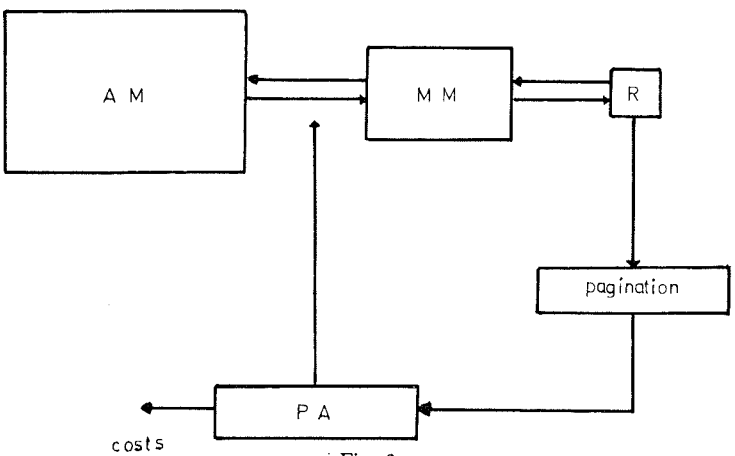

Fig. 3.

and cut out the part corresponding to Fig. 2:

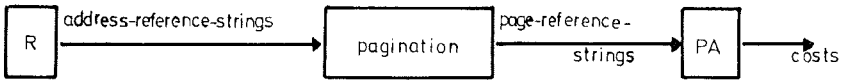

Fig. 4.

In this model R produces address-reference-strings which are translated to page-reference-strings by the pagination. For a given R (or a whole class of address-reference-generators) we may now ask the more general question:

Paging-Pagination-Problem: Find a pair (Pagination, PA) such that the costs of processing input strings generated by R are minimized.

By holding one of the components of the pagination-PA-pairs fixed we obtain two subproblems:

Analysis-Problem: Given R and a PA. Find a pagination such that the costs are minimized.

Synthesis-Problem: Given R and a pagination. Find a PA such that the costs are minimized.

Here, the pagination and the PA respectively, may be restricted to a given class.

In [6] we have tried to outline a theory of the paging-pagination-problem that is based on a deterministic model of R. In the present paper we shall treat but one special case of R which may be termed "simple loop", a situation though that occurs quite frequently in actual computations, in particular in connection with processing large arrays (for instance matrix multiplication, cf. [5]). In [7] another type of reference-generators — termed "loop-chains" — has been investigated. Furthermore, we shall confine ourselves to the class of demand-paging-algorithms (cf. [3]), i.e. pages are shifted from AM to MM on demand only. The cost of loading one page into main memory will be 1; other actions, like removing pages from main-memory, do not contribute any costs.

In Section 1 we present some motivating examples and the basic definitions. In Section 2 the unrestricted (modulo the boundary-stones just set) paging-pagination-problem will be treated. We show that one obtains an optimal pair by choosing as pagination the obvious simple-minded approach of writing adjacent addresses on the same page if possible, and as paging-algorithm some algorithm that realizes Belady's optimal strategy (cf. [2]). We note that this result may be looked upon as an exact formulation of a "locality principle" for simple loops. The proof employs an interesting lower bound argument.

In Section 3 some aspects of the synthesis problem are treated with respect to the special class of "rearrangement algorithms" which contains the well known FIFO and LRU algorithms. In Section 4 these latter algorithms will be analyzed according to the analysis-problem mentioned above. It will be shown in particular, that the simple-minded pagination is no longer optimal for LRU-paging of simple loops, and hence that the "locality principle" does not apply in this case.

1. Examples and definitions

Let A be a finite nonempty set of addresses (A may also be interpreted as a set of relocatable sectors of some program) and let X be a finite nonempty set of pages of sufficiently large cardinality. (More precisely, we should use the term "page-frames"; this distinction however, will not be substantial.) Paginations may be described as mappings from A to X such that the co-image of any $x \in X$ is of limited size:

Definition 1.1. Let $p \in N$. A (p-)pagination of A is a total map $h : A \to X$ such for all $x \in X$: $|h^{-1}(x)| \leq p$.

The natural extension of h to a mapping from $A^* \to X^*$ (where A^* and X^* are the free monoids generated by A and X respectively) will also be denoted by h. Furthermore, we may always assume without loss of generality that h *is onto*, since otherwise we would replace X by $h(A) \subset X$.

Let us now briefly sketch an example: Let $A = \{1, \ldots, 10\}$ and $X = \{a, b, c, d\}$. Let $p = 3$, i.e. each page may contain at most three addresses. Let the capacity of MM be $m = 2$ pages and let R generate the address-reference-string $w = (1\,2\,3\,4\,5\,6\,7\,8\,9\,10)^2$. We consider the following two paginations h_1 and h_2:

	1	2	3	4	5	6	7	8	9	10
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
h_1	a	a	a	b	b	b	c	c	c	d
h_2	a	b	b	b	a	c	c	c	a	d

h_1 and h_2 translate w into $h_1(w) = (a^3b^3c^3d)^2$ and $h_2(w) = (ab^3ac^3ad)^2$ respectively. Suppose we employ the following paging-algorithm P, specified by its "replacement rule":

P: *if* MM is full and page x is referenced and not contained in MM
then replace that y in MM by x which has been in MM for the longest time.

Starting with an initially empty MM we will obtain the following sequences of MM-contents corresponding to the page-reference-strings $h_1(w)$ and $h_2(w)$:

h_1: $\underline{\{a\}}, \{a\}, \{a\}, \underline{\{a, b\}}, \{a, b\}, \{a, b\}, \underline{\{b, c\}}, \{b, c\}, \{b, c\},$
$\underline{\{c, d\}}, \underline{\{d, a\}}, \{d, a\}, \{d, a\}, \underline{\{a, b\}}, \{a, b\}, \{a, b\}, \underline{\{b, c\}},$
$\{b, c\}, \{b, c\}, \underline{\{c, d\}}$

h_2: $\underline{\{a\}}, \underline{\{a, b\}}, \{a, b\}, \{a, b\}, \{a, b\}, \underline{\{b, c\}}, \{b, c\}, \{b, c\},$
$\underline{\{c, a\}}, \underline{\{a, d\}}, \{a, d\}, \underline{\{d, b\}}, \{d, b\}, \{d, b\}, \underline{\{a, b\}}, \underline{\{a, c\}},$
$\{a, c\}, \{a, c\}, \{a, c\}, \underline{\{c, d\}}.$

Here we have indicated each time the memory-contents changes and costs arise by underlining the respective sets. The costs of processing $h_1(w)$ are $K(h_1(w)) = 8$ whereas $K(h_2(w)) = 9$. Note that P is just the well-known FIFO-algorithm (i.e. first in first out).

Let us now replace P by the paging-algorithm P', defined as follows:

P': *if* MM is full and page x is referenced and not contained in MM *then* replace that y in MM by x which has not been referenced for the longest time.

Note that P' is the LRU-algorithm (i.e. least recently used). For P' we obtain the following sequences of MM-contents:

h_1: $\underline{\{a\}}, \{a\}, \{a\}, \underline{\{a, b\}}, \{a, b\}, \{a, b\}, \underline{\{b, c\}}, \{b, c\}, \{b, c\},$
$\underline{\{c, d\}}, \underline{\{d, a\}}, \{d, a\}, \{d, a\}, \underline{\{a, b\}}, \{a, b\}, \{a, b\}, \underline{\{b, c\}},$
$\{b, c\}, \{b, c\}, \underline{\{c, d\}}$

h_2: $\underline{\{a\}}, \underline{\{a, b\}}, \{a, b\}, \{a, b\}, \{a, b\}, \underline{\{a, c\}}, \{a, c\}, \{a, c\},$
$\{a, c\}, \underline{\{a, d\}}, \{a, d\}, \underline{\{a, b\}}, \{a, b\}, \{a, b\}, \{a, b\}, \underline{\{a, c\}}$
$\{a, c\}, \{a, c\}, \{a, c\}, \underline{\{a, d\}}.$

The costs of processing $h_1(w)$ by P' are again $K'(h_1(w)) = 8$ whereas now $K'(h_2(w)) = 7$.

The preceding examples show the influence of a particular pagination on the performance of some paging-algorithm (or else — what may easily be recognized as equivalent — the influence of a particular addressing-scheme). In Section 4 of this paper we shall pursue the discussion of the algorithm P'. It will be shown that the pagination h_2 is already optimal in this case.

In order to get rigorous answers to the questions posed in the introduction it is clearly necessary to set up some suitable formalization of the model and to formulate the said problems within the framework of this formalism.

For the address-reference-generator R we adopt the definition given in [7]:

Definition 1.2. A reference-generator is a total recursive function $R : \mathbf{N}_0 \to A^*$. $R(\mathbf{N}_0)$ is the set of address-reference-strings.

Here $\mathbf{N}_0$ denotes the set of natural numbers augmented by zero. Note that our model of R is strictly deterministic: the recursive function R may be represented by some deterministic string generating device like for instance a computer-program together with a recursively enumerable collection of data inputs.

Demand-paging-algorithms will be modelled as in [1], i.e. as sequential machines whose output yields the costs of processing an input-string of page-references. However, we do not represent states of the machine as pairs of "memory state" and "control state". It rather seems more convenient for our purposes to uniquely

associate with any state some memory-contents. Recall that by X we denote the set of pages. Let $m \in \mathbf{N}$ be some natural number. We shall always assume: $|X| \geq m$, since otherwise the "paging-problem" would be void.

Definition 1.3. An (m-)paging-automaton is a 5-tuple $P = (Q_P, q_{0P}, X, \delta_P, \tau_P)$ where: Q_P is a finite or countably infinite set of states; $q_{0P} \in Q_P$ is a distinguished initial state; X serves as set of input symbols and $\delta_P : Q_P \times X \to Q_P$ and $\tau_P : Q_P \to 2^X$ are total mappings such that for all $q \in Q_P$ and all $x \in X$:

(i) $x \in \tau_P(\delta_P(q, x))$,

(ii) $|\tau_P(q)| \leq m$; there exists $q \in Q_P$ such that $|\tau_P(q)| = m$,

(iii) $\tau_P(\delta_P(q, x)) \subset \tau_P(q) \cup \{x\}$,

(2^X denotes the set of subsets of X) P will be called a finite (m-)paging-automaton iff Q_P is finite.

Some remarks are necessary in order to appreciate this definition: $m \in N$ is the main-memory capacity. $\tau_P(q)$ is the main-memory contents corresponding to state q. δ_P is the next-state function. Postulate (i) then states that after input of page x (i.e. after a reference to x has been generated) that page should be in MM by the next step. (ii) says that the capacity m is actually needed, and (iii) is the demand condition: together with (i), (iii) assures that some page x is added to the "present memory contents" $\tau_P(q)$ if and only if $x \notin \tau_P(q)$ and x is the present input symbol. Usually $\tau_P(q_{0P})$ will be assumed empty, reflecting the fact that we start with an initially empty MM. For technical reasons we shall sometimes replace the input set X by a larger input set $Y \supset X$.

P will be omitted as a subscript whenever no confusion can arise.

Example 1.4. Using Definition 1.3 the FIFO- and LRU-algorithms may be written quite elegantly: Let $\not{e} \notin X$ be some extra symbol. We define a state-set $Q \subset (X \cup \{\not{e}\})^m$ (here, the superscript m denotes m-fold cartesian product) as follows:

$$q = (y_1, \ldots, y_m) \in Q \quad \text{if and only if:}$$

(i) $y_i \neq \not{e} \Rightarrow y_j \neq \not{e}$ for all $j > i$,

(ii) $y_i = \not{e} \Rightarrow y_j = \not{e}$ for all $j < i$,

(iii) $y_i, y_j \in X,\ i \neq j \Rightarrow y_i \neq y_j$.

The memory contents $\tau(y_1, \ldots, y_m) = \tau(q)$ of some state q will be the set of symbols in the m-tuple that are different from (the "dummy page") $\not{e}$, i.e. $\tau(q) = \bigcup_{y_i \neq \not{e}} \{y_i\}$. The initial state is $q_0 = (\not{e}, \ldots, \not{e}) \in Q$. Now FIFO is given by the next-state function

$$\delta_{\mathrm{FIFO}}(\underbrace{(y_1, \ldots, y_m)}_{q}, x) = \begin{cases} (y_2, \ldots, y_m, x) & \text{if } x \notin \tau(q), \\ q & \text{if } x \in \tau(q), \end{cases}$$

and LRU is given by the next-state function

$$\delta_{\mathrm{LRU}}((y_1, \ldots, y_m), x) = \begin{cases} (y_2, \ldots, y_m, x) & \text{if } x \notin \tau(q), \\ (y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_m, x) & \text{if } x = y_j. \end{cases}$$

In Section 3 a class of paging-automata will be defined that contains both FIFO and LRU.

The output of a paging-automaton P will be defined directly in terms of a cost function $k_P : Q \times X \to \mathbf{N}_0$ as follows:

$$k_P(q, x) := |\tau_P(\delta_P(q, x) - \tau_P(q)|.$$

Obviously $k_P(q, x) \in \{0, 1\}$ for any $q \in Q$ and $x \in X$, and $k_P(q, x) = 1$ if and only if $x \notin \tau_P(q)$, i.e. the costs 1 will be produced by the automaton P if and only if P has to transport an input $x \in X$ into main memory.

The mappings δ and k are inductively extended to mappings $\delta^* : Q \times X^* \to Q$ and $K : Q \times X^* \to \mathbf{N}_0$: $\delta^*(q, \square) = q$; $q \in Q$ (here $\square \in X^*$ denotes the empty word in X^*); $\delta^*(q, wx) = \delta(\delta^*(q, w), x)$; $q \in Q$, $x \in X$, $w \in X^*$; $K(q, \square) = 0$; $q \in Q$; $K(q, wx) = K(q, w) + k(\delta^*(q, w), x)$; $q \in Q$, $x \in X$, $w \in X^*$. δ^* will be denoted by δ again. $K(q, w)$ is the cost of processing input string $w \in X^*$ by P starting from $q \in Q$. We put $K(w) := K(q_0, w)$.

Let us now fix the set of words to be processed by the automata just defined. We shall employ the following notation:

Notation: For $w \in X^*$ and $x \in X$, $N(x, w)$ denotes the number of occurrences of symbol x in w; $L(w)$ denotes the length of w.

Definition 1.5. Let $A = \{a_1, \ldots, a_b\}$. The reference-generator $R : \mathbf{N}_0 \to A^*$ given by $R(n) = (a_1 \cdots a_b)^n$, $n \geq 0$, is called the simple loop over A.

Definition 1.6. Let $w \in X^*$ and b, $p \in \mathbf{N}$ be such that $L(w) = b$ and $0 < N(x, w) \leq p$ for all $x \in X$. The set $\mathrm{per}(w) = \{w^n \mid n \geq 0\}$ is called a periodic (b, p)-arrangement of X; w itself is called a (b, p)-arrangement.

Let $\mathrm{ARR}_{(b,p)}(X)$ denote the set of all (b, p)-arrangements of X. The connection between simple loops and periodic (b, p)-arrangements is quite obvious:

Proposition 1.7. *Let R be the simple loop over $A = \{a_1, \ldots, a_b\}$ and let $h : A \to X$ be a p-pagination. Then $\{h(R(n)) \mid n \geq 0\}$ is a periodic (b, p)-arrangement.*

Conversely we have:

Proposition 1.8. *Let $\mathrm{per}(w)$ be a periodic (b, p)-arrangement. Then there exists a simple loop over $A = \{a_1, \ldots, a_b\}$ and a p-pagination $h : A \to X$ such that $\mathrm{per}(w) = \{h(R(n)) \mid n \geq 0\}$.*

The proofs of both propositions are trivial. (Recall that h is always assumed to be onto!)

Propositions 1.7 and 1.8 show that looking for optimal paginations of simple loops amounts to finding optimal (b,p)-arrangements. The latter will be made precise in the following definition:

Definition 1.9. Let $\mathcal{P}$ be a class of paging-automata and $B \subset \mathrm{ARR}_{(b,p)}(X)$ a set of (b,p)-arrangements. A pair $(w,P) \in B \times \mathcal{P}$ is $(B,\mathcal{P})$-optimal iff for all $(w',P') \in B \times \mathcal{P}$ and for all $n \geq 1$: $K_P(w^n) \leq K_{P'}(w'^n)$.

If $\mathcal{P} = \{P\}$, then w is (B,P)-optimal iff (w,P) is $(B,\{P\})$-optimal. If $B = \{w\}$, then P is $(w,\mathcal{P})$-optimal iff (w,P) is $(\{w\},\mathcal{P})$-optimal.

Clearly, these definitions correspond in proper order to the three problems posed in the introduction. Since the results in Section 2 are not confined to (b,p)-arrangements only, we shall broaden Definition 1.9 a little:

Definition 1.6′. $w \in X^*$ is called an arrangement of X if $N(x,w) > 0$ for all $x \in X$.

Periodic arrangements are defined analogously, and Definition 1.9 applies word by word to arrangements of X if we replace $\mathrm{ARR}_{(b,p)}(X)$ by $\mathrm{ARR}(X)$, the set of arrangements of X, and hence admit $B \subset \mathrm{ARR}(X)$.

The paging-complexity of periodic arrangements will now be defined in terms of the cost-function K as follows: Let $\mathcal{P}$ and B be as above; let $w \in B$, $P \in \mathcal{P}$. Then:

$$C_P(w) = \limsup_{n\to\infty} n^{-1} K_P(w^n).$$

Definition 1.10. Let $\mathcal{P}$ and B be as above. $C_{\mathcal{P}}(B) = \inf\{C_P(w) \mid w \in B,\ P \in \mathcal{P}\}$ is called the $\mathcal{P}$-paging-complexity of B.

Clearly, if $(w,P) \in B \times \mathcal{P}$ is $(B,\mathcal{P})$-optimal then $C_P(w) = C_{\mathcal{P}}(B)$.

We shall first characterize $(\mathrm{ARR}(X), \mathcal{P}_m)$-optimality, where $\mathcal{P}_m$ is the class of all finite m-paging-automata.

2. The $\mathcal{P}_m$-paging-complexity of periodic arrangements

Let $|X| = M > m$ be the cardinality of X. By fixing a particular indexing of X we may without restricting generality assume that $X = \{1,\ldots,M\}$. We shall specify a pair $(w_0,P_0) \in \mathrm{ARR}(X) \times \mathcal{P}_m$ and show that it is $(\mathrm{ARR}(X), \mathcal{P}_m)$-optimal.

Definition 2.1. The automaton $P_0 \in \mathcal{P}_m$ is defined by: $Q = Q_m := \{q \mid q \in 2^X,\ |q| \leq m\}$; $q_0 = \emptyset$; $\tau = \mathrm{id}\colon Q \to 2^X$ and

$$\delta(q,x)=\begin{cases} q & \text{if } x\in q,\\ q\cup\{x\} & \text{if } x\notin q \text{ and } |q|<m,\\ (q-\{y\})\cup\{x\} & \text{if } x\notin q \text{ and } |q|=m,\end{cases}$$

where:

$$y=\begin{cases}\max\{y'\mid y'\in q\} & \text{if } x<y' \text{ for all } y'\in q,\\ \max\{y'\mid y'\in q,\ y'<x\} & \text{otherwise.}\end{cases}$$

The result stated in the following proposition is readily obtained by a straightforward calculation:

Proposition 2.2. *Let* $w_0=1^{p_1}\cdots M^{p_M}$, $p_i>0$ for all $i=1,\ldots,M$. Then for all $n>1$:

$$K_P(w_0^n)=M+(uM+t)(M-m)+t-t'$$

where $u=(n-1-t)/(M-1)\in\mathbf{N}_0$ for suitable $t\in\mathbf{N}_0$, $0\leqslant t<M-1$, and $t'=\min\{t,m-1\}$. For $n=1: K_{P_0}(w_0)=M$.

Corollary 2.3. $C_{P_0}(w_0)=(M/(M-1))(M-m)$.

We now claim: (w_0,P_0) is $(\mathrm{ARR}(X),\mathscr{P}_m)$-optimal. To show this, some classical results on paging-algorithms are employed: Belady's optimal strategy (cf. [2]) in the first place, and secondly the fact that this strategy constitutes a stack-paging-algorithm (cf. [3]). We show:

(i) P_0 realizes Belady's strategy for all reference-strings of the form w_0^n, $n\geqslant 1$. (The simple proof of this fact will be omitted.)

(ii) Sets of the form $\mathrm{per}(w_0)$ are uniquely optimal periodic arrangements with respect to Belady's strategy (modulo permutation of letters).

Belady's optimal strategy: Let $w=x_1\cdots x_{L(w)}\in X^*$. Define the m-paging-automaton $B_0^{(m)}(w)$ as follows: $Q=Q^{(m)}:=Q_m\times\mathbf{N}_0$; $q_0=(\emptyset,0)$; $\tau: Q^{(m)}\to 2^X$ is given by $\tau(q,t)=q$, $t\in N_0$. The transition rule $\delta=\delta^{(m)}: Q^{(m)}\times X\to Q^{(m)}$ is:

$$\delta^{(m)}(q,t,x)=\begin{cases}(q,t+1) & \text{if } x\in q,\\ (q\cup\{x\},t+1) & \text{if } x\notin q \text{ and } |q|<m,\\ ((q-\{y\})\cup\{x\},t+1) & \text{if } x\notin q \text{ and } |q|=m,\end{cases}$$

where $y=\max\{y'\mid y'\in q^w(t)\}$ with

$$q^w(t):=\{x\mid x\in q,\ d(t,x,w)=\max\{d(t,y,w)\mid y\in q\}\}$$

and

$$d(t,x,w) = \begin{cases} \min\{t' \mid t' > t,\ x_{t'} = x\} - t & \text{if the minimum exists,} \\ \infty & \text{otherwise.} \end{cases}$$

Loosely speaking, the strategy formalized by $B_0^{(m)}(w)$ is to banish — if necessary — that page from main-memory contents with the longest recurrence time $d(t,x,w)$ (with a tie-breaking rule in case that page is not uniquely defined).

For shortness we shall write in the sequel:

$$K^{(m)}((q,t),w) := K_P((q,t),w), \quad \text{where} \quad P = B_0^{(m)}(w), \quad (q,t) \in Q^{(m)}, \quad w \in X^*.$$

A simple proof of the following theorem — based on the principle of optimality of dynamic programming — can be found in [3], 251–253]:

Theorem 2.4. (Belady.) *Let $P \in \mathscr{P}_m$ and $w \in X^\times$. Then for all $q \in Q_P$: $K_P(q,w) \geqslant K^{(m)}((\tau_P(q),0),w)$.*

Now let w_0 be as in Proposition 2.2. One easily shows by induction:

Proposition 2.5. *For all $n \geqslant 1$: $K_{P_0}(w_0^n) = K^{(m)}(w_0^n)$.*

Proposition 2.5 settles proof-step (i) above. Inorder to prove (ii) (the harder part) we define:

Definition 2.6. $w \in \mathrm{ARR}(X)$ is an m-optimal arrangement of X iff for all $w' \in \mathrm{ARR}(X)$ and for all $n > 1 : K^{(m)}(w^n) \leqslant K^{(m)}(w'^n)$.

Note that in this definition we do not postulate the inequality for $n = 1$. The latter will turn out to be a consequence of the inequalities for $n > 1$. (Lemma 2.19 below.)

In the remaining part of this section m-optimal arrangements of X will be characterized:

Theorem 2.7. *The following statements are equivalent:*

(i) *$w \in \mathrm{ARR}(X)$ is an m-optimal arrangement of X,*

(ii) *$w = \pi(1)^{p_1} \cdots \pi(M)^{p_M}$ for some permutation $\pi : X \to X$; $p_i > 0$ for $i = 1, \ldots, M$.*

It follows in particular from Theorem 2.7 that $w_0 \in \mathrm{ARR}(X)$ is m-optimal, and together with Proposition 2.5 we get the $(\mathrm{ARR}(X), \mathscr{P}_m)$-optimality of (w_0, P_0). The key step in the proof of Theorem 2.7 is to establish a non-trivial lower bound on $K^{(m)}(w^n)$, $w \in \mathrm{ARR}(X)$, $n \geqslant 1$. The crucial lemma (2.15) yields an estimate of the costs saved by passing from $B_0^{(m)}$ to $B_0^{(m+1)}$, i.e. by enlarging main-memory capacity by one unit.

Lemma 2.8. *Let* $w \in \mathrm{ARR}(X)$. *Then* $K^{(m)}(w) \geq M$, *for all* m, $1 \leq m < M$.

The trivial proof, using the fact that for any paging-automaton P $\tau(\delta(q, x_1 \cdots x_t)) \subset \tau(q) \cup \bigcup_{i=1}^{t} \{x_i\}$ for all $q \in Q_P$, is omitted.

Lemma 2.9. *Let* $w \in \mathrm{ARR}(X)$. *Then for all* m, $1 \leq m < M$, *and for all* $q \in Q_m$: $K^{(m)}((q, 0), w) \geq M - m$.

Proof. Since $|q| \leq m$ and $|X| = M$, there are at least $M - m$ elements in X not contained in q. Each of these elements generates at least the costs 1. □

Lemma 2.10. *Let* $w \in \mathrm{ARR}(X)$ *and* $q \in Q_m$, $1 \leq m < M$. *Then*

$$\max\{n \mid K^{(m)}((q, 0), w^n) = n(M - m)\} \leq m.$$

Proof. If $K^{(m)}((q, 0), w) > M - m$ then — by Lemma 2.9 — there is nothing to show. Hence suppose $K^{(m)}((q, 0), w) = M - m$. Let $w = x_1 \cdots x_{L(w)}$. For $y \in X$ we put: $t_1(y) = \min\{t \mid x_t = y,\ 1 \leq t \leq L(w)\}$, i.e. $t_1(y)$ is the first instance of time y occurs in the string w.

Let $v_1, \ldots, v_M$ be the indexing of X such that $t_1(v_i) < t_1(v_j)$ iff $i < j$, $1 \leq i, j \leq M$.

Now $K^{(m)}((q, 0), w) = M - m$ implies $|q| = m$ (otherwise an immediate contradiction would be obtained). Hence there are exactly $M - m$ elements $v_{i_1}, \ldots, v_{i_{M-m}} \in X$ not contained in q, $-i_1 < \cdots < i_{M-m}$. Obviously $i_1 \leq m + 1$. Let $v \in q$ be the element removed from q by v_{i_1} (since $|q| = m$ such an element must exist). Then $x_t \neq v$ for all $t > t_1(v_{i_1})$, since otherwise we get an immediate contradiction to $K^{(m)}((q, 0), w) = M - m$. Moreover $v = v_i$ for some $i < i_1$ and also $v \notin \tau(\delta((q, 0), w))$.

Now taking $\tau(\delta((q, 0), w))$ instead of q and $v = v_i$ instead of v_{i_1}, the same argument applies to the second run of w, etc. It follows that there are at most $i_1 - 1 \leq m$ subsequent runs of w with the minimum costs $M - m$ (according to Lemma 2.9), as stated in the lemma. □

The indexing $\{v_1, \ldots, v_M\} = X$ as defined in the preceding proof will be kept fixed in the sequel. The proof of Lemma 2.10 shows that its statement can be sharpened as follows:

Corollary 2.11. *Let* $w \in \mathrm{ARR}(X)$ *and* $q \in Q_m$, $1 \leq m < M$, *be such that* $q \neq \{v_1, \ldots, v_m\}$. *Then* $\max\{n \mid K^{(m)}((q, 0), w) = n(M - m)\} \leq m - 1$.

Proof. Observe that $q \neq \{v_1, \ldots, v_m\}$ implies $i_1 \leq m$ (where $i_1 = m$ if and only if $\{v_1, \ldots, v_{m-1}\} \subset q$). □

Before stating and proving three "comparison lemmas" (2.13–2.15) we note an immediate consequence of Belady's algorithm being a stack-algorithm (cf. [3]):

Lemma 2.12. *Let* $w \in X^*$ *and* $q \in Q_m$, $q' \in Q_{m+1}$ *be such that* $q \subset q'$. *Then*: $K^{(m+1)}((q', 0), w) \leq K^{(m)}((q, 0), w)$.

Intuitively, Lemma 2.12 says that increasing main-memory capacity by 1 certainly does not increase costs if the start-state of the larger machine contains the start-state of the smaller machine. The straightforward proof can also be found in [3].

Lemma 2.13. *Let* $w \in \mathrm{ARR}(X)$ *and* $q \in Q_m$, $q' \in Q_{m+1}$, $1 \leq m < M$, *be such that* $q \subset q'$. *Then*:

$$K^{(m)}((q, 0), w) - K^{(m+1)}((q', 0), w) \geq 1.$$

Proof. By Lemma 2.12 $K^{(m+1)}((q', 0), w) \leq K^{(m)}((q, 0), w)$. By constructing an $(m+1)$-paging-automaton P that processes w cheaper than $B_0^{(m)}$ by at least the cost 1 and then applying Theorem 2.4 we show in fact that equality cannot hold:

Let $w = x_1 \cdots x_{L(w)}$, $w_t = x_1 \cdots x_t$, $1 \leq t \leq L(w)$; let $\{u\} \subset q' - q$ and let $t_1(u)$ be defined as in the proof of Lemma 2.10. P is given by: $Q_P = Q^{(m+1)}$, τ as in $B_0^{(m)}$ and

$$\delta_P((q, t), x) = \begin{cases} (\tau\delta^{(m)}((q - \{u\}, t), x) \cup \{u\}, t + 1) & \text{if } t < t_1(u) \text{ and } u \in q, \\ \delta^{(m+1)}((q, t), x), & \text{otherwise.} \end{cases}$$

(Recall that $\delta^{(m)}$ is the transition function of $B_0^{(m)}$!) With q and q' as given in the assumptions of the lemma one immediately checks:

1. $K_P((q', 0), w_t) = K^{(m)}((q, 0), w_t)$ for $t = t_1(u) - 1$, i.e. up to $t_1(u)$ P is just as expensive as $B_0^{(m)}$.
2. $K_P((q', 0), w_t) + 1 = K^{(m)}((q, 0), w_t)$ for $t = t_1(u)$, i.e. at time-instant $t_1(u)$ P saves cost 1 since u is already contained in $\tau\delta_P((q', 0), w_t)$.
3. $\tau\delta^{(m)}((q, 0), w_t) \subset \tau\delta_P((q', 0), w_t)$ for $t = t_1(u)$, and — by Lemma 2.12:
4. $K_P(\delta_P((q', w_t), x_{t+1} \cdots x_{L(w)}) = K^{(m+1)}(\delta_P((q', 0), w_t), x_{t+1} \cdots x_{L(w)} \leq K^{(m)}(\delta^{(m)}((q, 0), w_t), x_{t+1} \cdots x_{L(w)})$ for $t = t_1(u)$, whence:
5. $1 + K^{(m+1)}((q', 0), w) \leq 1 + K_P((q', 0), w) \leq K^{(m)}((q, 0), w)$. □

Lemma 2.14. *Let* $w \in \mathrm{ARR}(X)$ *and* $q \in Q_m$, $1 \leq m < M$, *be such that* $|q| = m$ *and* $K^{(m)}((q, 0), w) > M - m$. *Then there exists* $q' \in Q_{m+1}$ *such that* $q \subset q'$ *and* $K^{(m)}((q, 0), w) - K^{(m+1)}((q', 0) \geq 2$.

Proof. Again, we construct an $(m+1)$-paging-automaton P that processes w cheaper than $B_0^{(m)}$ by at least the costs 2. Let w and w_t be as in the preceding proof. Let $U = \{v_{i_1}, \ldots, v_{i_{M-m}}\} = X - q$, $i_1 < \cdots < i_{M-m}$.

(Recall the remark following the proof of Lemma 2.10!) Then there exists a first (in the given indexing) $u \in U$ such that the element $y \in X$ removed by u from $\tau\delta^{(m)}((q,0), w_{t_1(u)-1})$ recurs after $t_1(u)$, i.e. there exists $t > t_1(u)$ such that $x_t = y$. (If this were not the case an immediate contradiction to $K^{(m)}((q,0), w) > M - m$ would be obtained!).

Hence u generates at least the costs 2: at $t = t_1(u)$ by itself and at $t = t_2(y) := \min\{t \mid t > t_1(u), x_t = y\}$ by y. Now P will be defined such that up to $t_1(u)$ P works like $B_0^{(m)}$ with u added to the memory contents; from $t_1(u)$ up to $t_2(y)$ P will work like $B_0^{(m)}$ with y added to the memory contents and after $t_2(y)$ P will work like $B_0^{(m+1)}$. We forego the formal definition of P. With q as assumed in the lemma and $q' = q \cup \{u\}$, we get: $K_P((q',0), w) \leq K^{(m)}((q,0), w) - 2$. The claim now follows via Theorem 2.4. □

Lemma 2.15. *Let $w \in \mathrm{ARR}(X)$ and $q \in Q_m$, $1 \leq m < M-1$, be such that $|q| = m$. Then there exists $q' \in Q_{m+1}$ such that $q \subset q'$ and*

$$M + K^{(m+1)}((q',0), w^{M-1}) \leq K^{(m)}((q,0), w^{M-1}).$$

Proof. By definition of $B_0^{(m)}$ we have for all $n > 0$:

$$K^{(m)}((q,0), w^n) = \sum_{i=0}^{n-1} K^{(m)}((q(i),0), w),$$

where we put: $q(i) := \tau\delta^{(m)}((q,0), w^i)$, $i \geq 0$. Now let $q' \in Q_{m+1}$ be such that $q \subset q'$. Let $q'(i)$ be analogously defined. Then $q(i) \subset q'(i)$, $i \geq 0$, by the fact that Belady's algorithm is a stack-algorithm. Therefore, by Lemma 2.13, $1 + K^{(m+1)}((q'(i),0), w) \leq K^{(m)}((q(i),0), w)$ for all $i \geq 0$, and hence: $M - 1 + K^{(m+1)}((q',0), w^{M-1}) \leq K^{(m)}((q,0), w^{M-1})$. By Lemma 2.10 and since $m < M-1$ there is at least one $j < M-1$ such that $K^{(m)}((q(j),0), w) > M - m$. Now Lemma 2.14 guarantees the existence of $q_j'' \in Q_{m+1}$ with $q(j) \subset q_j''$ and $2 + K^{(m+1)}((q_j'',0), w) \leq K^{(m)}((q(j),0), w)$. As in the preceding proofs we therefore have to construct an $(m+1)$-paging-automaton P whose state $\bar{q} \in Q_P$ at the beginning of the $j+1$ run of w satisfies $\tau_P(\bar{q}) = q_j''$ and which saves enough costs up to that time. Again, the formal definition of P is straightforward and will be omitted. The lemma now follows via Theorem 2.4. □

Lemma 2.15 immediately yields the following corollaries:

Corollary 2.16. *For $w \in \mathrm{ARR}(X)$ and $q \in Q_m$, $1 \leq m < M-1$:*

$$M + \min\{K^{(m+1)}((q',0), w^{m-1}) \mid q \subset q', q' \in Q_{m+1}\} \leq K^{(m)}((q,0), w^{M-1}).$$

Corollary 2.17. *Under the same assumptions:*

$$K^{(m)}((q,0), w^{M-1}) \geq M(M-m-1) + \min\{K^{(M-1)}((q',0), w^{M-1}) \mid q \subset q', q' \in Q_{M-1}\}.$$

The second one of these corollaries is obtained by repeated application of the first one. Corollary 2.17 compares the costs of processing w^{M-1} by an optimal algorithm with main-memory capacity m to the costs of processing the same string by an optimal algorithm that has a main-memory with capacity $M-1$ at hand: the smaller machine is more expensive by at least the costs $M(M-m-1)$. Thus the search for a lower bound on $K^{(m)}$ is reduced to estimating $K^{(M-1)}$, which is trivial:

Lemma 2.18. *For $w \in \mathrm{ARR}(X)$ and $q \in Q_m$, $1 \leq m < M$:*

$$\min\{K^{(M-1)}((q',0), w^{M-1}) \mid q \subset q', q' \in Q_{M-1}\} \begin{cases} \geq M-1 & \text{if } v_M \notin q, \\ \geq M & \text{if } v_M \in q, \end{cases}$$

where v_M is the last element in the indexing of X as defined in the proof of Lemma 2.10.

Proof. Lemma 2.18 follows from Lemma 2.10 in connection with Corollary 2.11. □

With $q(1) := \tau\delta^{(m)}((\emptyset,0), w)$ we can write:

$$K^{(m)}(w^n) = K^{(m)}((\emptyset,0), w^n) = K^{(m)}(w) + K^{(m)}((q(1),0), w^{n-1}), \qquad n \geq 1.$$

The following lemma then in particular shows that $v_M \in q(1)$ must be the case if w is an m-optimal arrangement. (It also shows that $K^{(m)}(w) = M$ if $w \in \mathrm{ARR}(X)$ is m-optimal!) This will be explicated in the subsequent proof of Theorem 2.7.

Lemma 2.19. *Let $w \in \mathrm{ARR}(X)$ and $1 \leq m < M$. If $K^{(m)}(w) > M$ then for all $q \in Q_m$: $K^{(m)}((q,0), w) > M - m$.*

Proof. The claim is trivial if $|q| < m$; hence suppose that $|q| = m$. Let $\{v_1, \ldots, v_M\} = X$ be the indexing defined via t_1 (the time of first occurrence, cf. proof of Lemma 2.10). Also recall that for $w = x_1 \cdots x_{L(w)}$ we put $w_t = x_1 \cdots x_t$, $1 \leq t \leq L(w)$. Then $\tau\delta^{(m)}((\emptyset,0), w_{t_1(v_m)}) = \{v_1, \ldots, v_m\}$ and $K^{(m)}((\emptyset,0), w_{t_1(v_m)}) = m$. By assumption we get: $K^{(m)}((\{v_1, \ldots, v_m\}, 0), x_{t_1(v_m)+1} \cdots x_{L(w)}) > M - m$. The latter inequality implies:

There exists v_i, $i > m$ minimal with the following property:

"Let $t(i) := t_1(v_i) - 1$, $q_{t(i)} := \tau\delta^{(m)}((\emptyset,0), w_{t(i)})$ and $\{x\} = q_{t(i)} - q_{t(i)+1}$. There exists $t' > t_1(v_i)$ with $x_{t'} = x$."

Thus v_i is the first element (in the given indexing) after v_m which removes some x from the memory contents that recurs after $t_1(v_i)$ and produces additional costs. For that x we put:

$$\bar{t} = \min\{t' \mid t' > t_1(v_i), x_{t'} = x\}.$$

Now by definition of $B_0^{(m)}$, x is that element among the elements of $q_{t(i)}$ which is at maximal distance from $t_1(v_i)$; it follows that at least $m-1$ elements $y_1, \ldots, y_{m-1} \in X$ (different from x) occur between $t_1(v_i)$ and $\bar{t}$. Moreover $t_1(y) < t_1(v_i)$ must hold for all $y \in \{y_1, \ldots, y_{m-1}\}$. (Hence w must be of the following form:

$$w = v_1 \cdots x \cdots \cdots \underset{\substack{\uparrow \\ t=t_1(v_i)}}{v_i} \cdots y_1 \cdots \cdots y_j \cdots \cdots y_{m-1} \cdots \underset{\substack{\uparrow \\ t=\bar{t}}}{x} \cdots).$$

For $q \in Q_m$, $|q| = m$, we now put $\bar{q} := \tau\delta^{(m)}((q,0), w_{t(i)})$.

Claim. *Without loss of generality we may assume* $\{y_1, \ldots, y_{m-1}\} \subset \bar{q}$.

Proof. Suppose this is not the case. Let $y \in \{y_1, \ldots, y_{m-1}\}$ and $y \notin \bar{q}$. Then there is an instant t, $t < t(i)$, with $q_{t-1} - q_t = \{y\}$. This banishment of y was accompanied by cost 1. On the other hand, y recurs after $t_1(v_i)$ and hence generates cost 1 once more. Consequently $K^{(m)}((q,0), w) \geq M - m + 1$.

Thus we only have to consider the cases $x \in \bar{q}$ and $x \notin \bar{q}$ respectively. For $x \in \bar{q}$ we get: $\bar{q} = \{x, y_1, \ldots, y_{m-1}\}$, the old situation: $\tau\delta^{(m)}((\emptyset,0), w_{t(i)})$. But in the case $x \notin \bar{q}$ we may argue as in the proof of the claim above. This proves the lemma. □

We are now ready to prove Theorem 2.7:

Proof of Theorem 2.7. (i) $\Rightarrow$ (ii): Let $w \in \mathrm{ARR}(X)$ be an m-optimal arrangement. Then $K^{(m)}(w) = M$. If this were not the case then $K^{(m)}(w^n)$, $n > 1$, would certainly surpass the upper bound stated in Proposition 2.2. This follows from Lemma 2.19.

Now $K^{(m)}(w) = M$ implies $v_M \in q(1) = \tau\delta^{(m)}((\emptyset,0), w)$. (Otherwise we would have $K^{(m)}(w) > M$.)

Hence, by Corollary 2.17 and Lemma 2.18: $K^{(m)}(w^M) \geq M + M(M-m)$. This lower bound is achieved — according to Propositions 2.2 and 2.5 — by any $\bar{w} \in \mathrm{ARR}(X)$ of the form $\bar{w} = 1^{p_1} \cdots M^{p_M}$, $p_i > 0$, $1 \leq i \leq M$. Moreover, it follows from Lemma 2.10 that

$$K^{(m)}(w^{1+i}) \geq K^{(m)}(\bar{w}^{1+i}) = M + i(M-m) + i - \min\{i, m-1\},$$

for $i = 0, \ldots, M-1$. Since w is m-optimal equality must hold and in particular: $K^{(m)}(w^m) = M + (m-1)(M-m)$. Using the arguments in the proof of Lemma 2.10 it can be shown that:

1. $\tau\delta^{(m)}((\emptyset,0), w) = \{v_1, \ldots, v_{m-1}, v_M\}$
2. $\tau\delta^{(m)}((\emptyset,0), w^2) = \{v_1, \ldots, v_{m-2}, v_{M-1}, v_M\}$
⋮
i. $\tau\delta^{(m)}((\emptyset,0), w^i) = \{v_1, \ldots, v_{m-i}, v_{M-i+1}, \ldots, v_M\}$ for $i < m$.

Together with $K^{(m)}(\delta^{(m)}((\emptyset, 0), w^i), w) = M - m$, $i < m$, we obtain from equations 1., 2., ..., $m-1$.: "For all $v_i, v_j \in X$, $i < j : x_t \neq v_i$ for $t > t_1(v_j)$". This is equivalent to: "w is of the form $w = v_1^{p_1} \cdots v_M^{p_M}$, $p_i > 0$, $1 \leq i \leq M$". But this is (ii).

(ii) $\Rightarrow$ (i): Let $\pi : X \to X$ be a permutation and $\bar{w} = \pi(1)^{p_1} \cdots \pi(M)^{p_M}$, $p_i > 0$, $1 \leq i \leq M$. We have to show: for all $w \in \mathrm{ARR}(X)$ and for all $n > 1 : K^{(m)}(w^n) \geq K^{(m)}(\bar{w}^n)$. Suppose this is not true. Then there exists $w \in \mathrm{ARR}(X)$ and some $n_0 > 1$ such that $K^{(m)}(w^{n_0}) < K^{(m)}(\bar{w}^{n_0})$. By Lemma 2.19 and by the first part of this proof we have:

1. $K^{(m)}(w) = M$; 2. $n_0 > M$; 3. $K^{(m)}(w^m) > M + (m-1)(M-m)$. By assumption there exists at least one $i > 1$ such that for $q = \tau\delta^{(m)}((\emptyset, 0), w^{1+i(M-1)})$:

$$K^{(m)}((q, 0), w^{M-1}) = M(M-m-1) + K^{(M-1)}((q', 0), w^{M-1}) = M(M-m) - 1$$

where

$$v_M \notin q,\ q' = \{v_1, \ldots, v_{M-1}\} \quad \text{and} \quad K^{(M-1)}((q', 0), w^{M-1}) = M - 1.$$

As in the first part of this proof one now obtains: w is of the form $w = \bar{w}$ ($\bar{w}$ as above), contradicting 3. This completes the proof of Theorem 2.7. □

It follows in particular from Theorem 2.7 that an arrangement $w \in \mathrm{ARR}_{(b,p)}(X)$ is $(\mathrm{ARR}_{(b,p)}(X), \mathscr{P}_m)$-optimal if $w = 1^p \cdots (M-1)^p M^s$, where $b = (M-1)p + s$, $1 \leq s \leq p$.

By Propositions 1.7 and 1.8 this proves the simple-minded pagination of simple loops optimal for any algorithm which realizes Belady's optimal algorithm. Theorem 2.7 may thus be regarded as an exact formulation of a principle of locality for simple loops: Among all paginations of a simple loop those and only those are optimal with respect to an optimal algorithm which write adjacent addresses (i.e. addresses that are accessed consecutively) on the same page, if possible. Equivalently (as explicated in the introduction), this may be formulated as follows: Given a pagination of a simple loop and an optimal paging-algorithm. Then in order to obtain minimal page-traffic the addresses have to be accessed page by page, i.e. first generate as many references as possible to the first page, then to the second page, etc.!

It should be noted that this result, though intuitively quite obvious (one might add: just as the optimality of the simple-minded approach to monotone matrix-multiplication is intuitively obvious!), deserves and *needs* a proof if we want to appreciate the way by which the simple-minded pagination is non-optimal for non-optimal paging-algorithms. We defer further discussion of that matter to Section 4 where some specific examples (namely FIFO and LRU) will be investigated.

3. Optimal and nearly-optimal rearrangement-automata

The costs produced by the finite paging-automaton P_0 that has been defined in the previous section, are not invariant under permutations of symbols of X. (Or else, speaking in terms of paginations, they are not invariant with respect to renaming the pages.) It is therefore of interest to look for classes of finite paging-automata which do posses this invariance property. In this section we shall specify such a class which contains both the automata FIFO and LRU. We ask whether this class also contains an automaton that realizes $B_0^{(m)}$ for m-optimal (b, p)-arrangements. It turns out that in general this is not the case. This leads to the distinction of "nearly-optimal" members of that class.

Let us put $[m] := \{1, 2, \ldots, m\}$ for $m \in N$.

Definition 3.1. An m-rearrangement-automaton is a 5-tuple $P = (Q, q_0, X, \delta, \tau)$ where: Q, q_0 and τ are as defined in Example 1.4 (over the input-set X) and:

$$\delta(\underbrace{(y_1, \ldots, y_m)}_{q}, y_{m+1}) = \begin{cases} (y_{\alpha_{jl}(1)}, \ldots, y_{\alpha_{jl}(m)}) & \text{if } y_{m+1} = y_j \text{ for } j \in [m] \text{ and } |\tau(q)| = l \\ (y_{\beta_l(1)}, \ldots, y_{\beta_l(m)}) & \text{if } y_{m+1} \notin \tau(q) \text{ and } |\tau(q)| = l \end{cases}$$

where

(i) $\alpha = \alpha_{jl} : [m] \to [m]$ is a permutation satisfying $\alpha(i) = i$ if $y_i = \not\epsilon$

(ii) $\beta = \beta_l : [m] \to [m+1]$ is an injection satisfying $m+1 \in \beta([m])$ and

$$y_1 = \not\epsilon \Rightarrow 1 \notin \beta([m]);\ y_i = \not\epsilon,\ i > 1 \Rightarrow \beta(i-1) = i$$

for $l = 0, 1, \ldots, m$; $j = 1, \ldots, m$.

It is easily checked that an m-rearrangement-automaton is a finite m-paging-automaton with $\tau(q_0) = \emptyset$. We forego a formal statement and proof of the invariance property mentioned above which is satisfied by rearrangement-automata. The term "rearrangement-automaton" has been chosen in order to indicate that any state-transition is accompanied by a rearrangement of the state-vector.

Example 3.2. Continuing Example 1.4 we write the automata FIFO and LRU as rearrangement-automata. All that remains to be done is to specify the mappings α_{jl} and β_l:

1. FIFO(m): In this case, $\alpha = \alpha_{jl} = \mathrm{id}$ (= identity) for all l and j ($l = 0, \ldots, m$; $j = 1, \ldots, m$). For $l = 0, \ldots, m$ $\beta_l : [m] \to [m+1]$ is given by $\beta_l(i) = i + 1$. Evidently, both α and $\beta = \beta_0 = \cdots = \beta_m$ satisfy conditions (i) and (ii) of Definition 3.1.

2. LRU(m): $\beta = \beta_0 = \cdots = \beta_m$ is as in 1 above. $\alpha_{jl} : [m] \to [m]$ is given by: $\alpha_{jl}(i) = i$ for $i < j$; $\alpha_{jl}(i) = i + 1$ for $j \leq i < m$; $\alpha_{jl}(m) = j$.

Again, it is easily checked that $\alpha_j = \alpha_{j1} = \cdots = \alpha_{jm}$ satisfies condition (i) of Definition 3.1.

We may now ask whether there are rearrangement-automata that realize $B_0^{(m)}$ for m-optimal (b,p)-arrangements. The preliminary answer is yes:

Proposition 3.3. *Let* $w = v_1^p \cdots v_{M-1}^p v_M^s \in \mathrm{ARR}_{(b,p)}(X)$ *(where* $\{v_1, \ldots, v_M\} = X$, $b = (M-1)p + s$, $1 < s \leq p$, $p \geq 2$*). Then for all* $m \geq 1$ *there exists an* m*-rearrangement-automaton* P *such that for all* $n \geq 1 : K_P(w^n) = K^{(m)}(w^n)$.

Proof. The claim is trivial for $M \leq m$. Hence let $M > m$. We define a suitable m-rearrangement-automaton by specifying the mappings α_{jl}, β_l:

1. $\alpha_{il} = \mathrm{id}$ for $l = 1, \ldots, m-1$ and $i = m-l+1, \ldots, m$

   $\alpha_{im} = \mathrm{id}$ for $i = 1, \ldots, m-1$;

   $$\alpha_{mm}(j) = \begin{cases} j-1 & \text{for } j \neq 1 \\ & \text{(cyclic shift)} \\ m & \text{for } j = 1, \end{cases}$$

2. $\beta_l(j) = j+1$ for $l = 0, \ldots, m-2, m$ and $j = 1, \ldots, m$;

   $$\beta_{m-1}(j) = \begin{cases} j & \text{for } j \neq 1, \\ & \\ m+1 & \text{for } j = 1. \end{cases}$$

It is easily verified that these mappings define a rearrangement-automaton that realizes (in the sense explicitly stated in the proposition) the optimal strategy $B_0^{(m)}$. □

However, the automaton defined in the preceding proof no longer satisfies $K_P(q, w^n) = K^{(m)}((\tau_P(q), 0), w^n)$ for arbitrary $q \in Q_P$. On the other hand the memory contents — represented by $\tau(q)$ — need not be empty when a program is entering one of its loops. In order to model this situation let X (i.e. the set of pages a simple loop is written on) be contained in some superset $Y \supset X$ and consider the class of rearrangement-automata constructed over the input set Y instead of X. It will be shown that no such rearrangement-automaton can achieve the optimal costs on optimal (b,p)-arrangements if it starts in a state whose associated memory contents is contained in $Y - X$:

Proposition 3.4. *Let* $X \subset Y$, $|Y - X| \geq 2$; *let* $w \in \mathrm{ARR}_{(b,p)}(X)$ *be as in Proposition* 3.3, *and let* P *be some* m*-rearrangement-automaton with input set* Y $(1 < m < M)$. *Let* $q \in Q_P$ *be such that* $|\tau(q)| > 1$ *and* $\tau(q) \cap X = \emptyset$. *Then for at least one* $n > 0 : K_P(q, w^n) > K^{(m)}(w^n)$.

Proof. Suppose $K_P(q, w^n) = K^{(m)}(w^n)$ for all $n > 0$. By the invariance property of P we may without loss of generality assume $w = 1^p \cdots (M-1)^p M^s$. Now let $q = (y_1, \ldots, y_m)$ where for some l, $1 \leq l < m-1$: $y_i = \not\in$ for $i < l$ and $y_i \neq \not\in$ for $i \geq l$.

Arguing as in the proof of Lemma 2.10, we get: $\tau\delta_P(q, w) = \{1, \ldots, m-1, M\}$. Hence $\tau\delta_P(q, 1^p \cdots m^p) = \{1, \ldots, m\}$. Let $\delta_P(q, 1^p \cdots m^p) = (y'_1, \ldots, y'_m)$. Then we must have: $y'_j = m$, where $\{j\} = [m] - \beta_m([m])$ is the index of that element in the state-vector $(y'_1, \ldots, y'_m)$ which will be banned from main-memory by the next β_m-transition step. But since $|\tau(q)| > 1$, this implies: in the state-vector $\delta_P(q, 1^p \cdots (m-1)^p) = (y''_1, \ldots, y''_m)$ the element $m-1$ must have been on the fatal position j (i.e. $y''_j = m-1$) already, which contradicts $\tau\delta_P(q, w) = \{1, \ldots, m-1, M\}$. □

Corollary 3.5. *Under the assumptions of Proposition* 3.4: *for at least one* $n > 0$: $K_P(q, w^n) > K^{(m)}((\tau_P(q), 0), w^n)$, *where the algorithm* $B_0^{(m)}$ *is defined over the larger input-set* Y.

Proof. Obviously, under the said assumptions: $K^{(m)}(w^n) = K^{(m)}((\tau_P(q), 0), w^n)$. □

It follows that for periodic (b, p)-arrangements, rearrangement-automata are in general only sub-optimal. However, it is possible to find rearrangement-automata that work nearly optimal on periodic (b, p)-arrangements for any start state $q \in Q$ with $\tau(q)$ not containing any symbol of w. Observe that in the remainder of this section, rearrangement-automata will be defined over the larger input-set Y.

Definition 3.6. Let $(w, P) \in \mathrm{ARR}_{(b,p)}(X) \times \mathcal{P}_m (m > 1)$ be not $(w, \mathcal{P}_m)$-optimal and let $q \in Q_p$. Then P is called w-nearly-optimal with respect to q iff for all $n > 0$:

$$K_P(q, w^{n+1}) - K_P(q, w^n) \leq M - m + 1,$$

where $M = \lceil b/p \rceil$ (= least integer greater than b/p).

Example 3.7. Define the m-rearrangement-automaton $P = NO(m)$, $m > 1$, by specifying the α's and β's as follows:

(i) $\alpha_{il} = \mathrm{id}$, for $l = 1, \ldots, m-1$ and $i = m-l+1, \ldots, m$;

$\alpha_{im} = \mathrm{id}$, for $i = 1, \ldots, m-1$

$$\alpha_{mm}(j) = \begin{cases} j-1 & \text{for } j \neq 1 \\ & \text{(cyclic shift)} \\ m & \text{for } j = 1 \end{cases}$$

(ii) $\beta_l(j) = j+1$ for $l = 0, \ldots, m$ and $j = 1, \ldots, m$.

Lemma 3.8. *Let* $P = NO(m)$, $m > 1$. *Let*

$w_1 = 12 \cdots (m-1)m^{p-1}(m+1)^p \cdots M^p m 1^{s-1} 2^{p-1} \cdots (m-1)^{p-1}$, $p > 2$, $0 < s \leq p$, *and*

$w_2 = 12 \cdots (m-1)m^2 \cdots (M-1)^2 1^{s-1} 2 \cdots (m-2)M(m-1)M$, *where* $s \in \{1, 2\}$.

Then for all $q \in Q_P$ *such that* $\tau(q) \cap X = \emptyset$ *and for all* $n > 0 : K_P(\delta_P(q, w_i^n), w_i) = M - m + 1$, $i = 1, 2$.

The proof of Lemma 3.8 follows from 3.7 in a straightforward manner.

Proposition 3.9. *Let* $P = NO(m)$, $m > 1$. *Then for any* $p \geq 2$ *and* $b = (M-1)p + s$, $0 < s \leq p$, *there exists* $w \in ARR_{(b,p)}(X)$ *such that* P *is* w*-nearly-optimal with respect to any* $q \in Q_P$, *provided* $\tau(q) \cap X = \emptyset$.

Proof. For $p = 2$ choose $w = w_2$, else $w = w_1$, with w_1, w_2 as defined in Lemma 3.8. □

In terms of paginations of simple loops the latter result means that for any page-size $p \geq 2$ there exists a pagination and a suitable finite paging-automaton — that is invariant with respect to renaming the pages — such that the simple loop is processed at nearly optimal costs.

In the following section, the theorems on FIFO and LRU in particular show that in general for an arbitrary rearrangement-automaton P there is no (b, p)-arrangement w such that P is w-nearly-optimal.

4. The FIFO- and LRU-complexities of periodic arrangements

We consider the analysis problem for the special rearrangement-automata FIFO(m) and LRU(m) which have been described in Examples 1.4 and 3.2. We shall first prove a statement about the automaton FIFO(m) which shows that m-optimal arrangements are also optimal with respect to FIFO(m), and which also shows that FIFO(m) is far from processing any periodic (b, p)-arrangement at nearly-optimal costs.

Again, in this section the automata in question are assumed to work over some larger input-set $Y \supset X = \{1, \ldots, M\}$.

Theorem 4.1. *Let* $P = \mathrm{FIFO}(m)$, $1 \leq m < M$, *and let* $w \in \mathrm{ARR}(X)$. *Then for all* $q \in Q_P$ *with* $\tau(q) \cap X = \emptyset$: $K_P(q, w^n) \geq n \cdot M$ *for all* $n > 0$.

Proof. We show that during each run of w each symbol $x \in X$ generates at least the cost 1. To this end we define the costs $\kappa_P(x, q, w)$ some element $x \in X$ is responsible for while P processes w starting from state $q \in Q_P$:

$$\kappa_P(x, q, w) := |\{t \mid x_t = x,\ x \notin \tau q(t-1),\ 1 \leq t \leq L(w)\}|.$$

Here, we put: $w = x_1 \cdots x_{L(w)}$, $w_t = x_1 \cdots x_t$ and $q(0) = q$, $q(t) = \delta(q, w_t)$ for $1 \leq t \leq L(w)$. With P and q as in the assumptions of the theorem we now claim:

(A) For all $x \in X$ and for all $n \geq 0$: $\kappa_P(x, \delta(q, w^n), w) \geq 1$. Obviously, this implies the theorem, since:

$$\kappa_P(x, q, w^n) = \sum_{r=0}^{n-1} \kappa_P(x, \delta(q, w^r), w)$$

and

$$K_P(q, w^n) = \sum_{x \in X} \kappa_P(x, q, w^n).$$

Since $\tau(q) \cap X = \emptyset$, we trivially get $\kappa_P(x, q, w) \geq 1$ for all $x \in X$. To prove claim (A) it remains to show:

(B) If $q \in Q_P$ is such that $\kappa_P(x, q, w) \geq 1$ for all $x \in X$ then $\kappa_P(x, \delta(q, w), w) \geq 1$ for all $x \in X$.

Claim (A) then follows by induction. Now let $\kappa_P(x, q, w) \geq 1$ for all $x \in X$. Define $T(x) := \max\{t \mid t \leq L(w), x_t = x, x \notin \tau q(t-1)\}$ (i.e. the last time x generates costs while P processes w). By assumption $T(x)$ exists for all $x \in X$. Let $X = \{y_1, \ldots, y_M\}$ be the indexing such that $T(y_1) < T(y_2) < \cdots < T(y_M)$. We claim:

(C) $\delta(q, w) = (y_{M-m+1}, \ldots, y_M)$.

In order to prove claim (C) we look how the FIFO-rearrangement of state-vectors works: For $q = (z_1, \ldots, z_m)$ and $x \in \tau(q)$ we define $\mathrm{pos}(x, q)$, the "position of x in q", to be

$$\mathrm{pos}(x, q) = j \quad \text{iff} \quad x = z_j. \text{ For } x \notin \tau(q), \mathrm{pos}(x, q) \text{ remains undefined.}$$

For pos the following statements hold:

(i) $\mathrm{pos}(x, \delta(q, w_{t+1})) \leq \mathrm{pos}(x, \delta(q, w_t))$ for $t = 0, \ldots, L(w) - 1$ if $\mathrm{pos}(x, \delta(q, w_t))$ is defined and greater than 1.

(ii) $x \notin \tau\delta(q, w_{t+1})$ if $\mathrm{pos}(x, \delta(q, w_t)) = 1$ and $x_{t+1} \notin \tau\delta(q, w)$.

Claim (C) immediately follows from these observations. (C) implies $\kappa_P(x, \delta(q, w), w) \geq 1$ for all $x \in \{y_1, \ldots, y_{M-m}\}$. Now let $T'(y_i) = \min\{t \mid x_t = y_i\}$, $i = 1, \ldots, M$, be the first time y_i occurs in w. By definition of $T(x)$, we have:

$$T'(y_i) < T(y_{M-m+1}) < \cdots < T(y_M) \quad \text{for all } i = 1, \ldots, M - m.$$

With (i) and (ii) above one checks: y_{M-m+1} will be removed from $\delta(q, w)$ by some $x \in \{y_1, \ldots, y_{M-m}\}$ and hence generates costs at some later time. The same happens to y_{M-m+2} and so on, up to y_M. Hence $\kappa_P(x, \delta(q, w), w) \geq 1$ for all $x \in \{y_{M-m+1}, \ldots, y_M\}$ also. The theorem now follows. □

Corollary 4.2. $C_{\mathrm{FIFO}(m)}(\mathrm{ARR}(X)) = C_{\mathrm{FIFO}(m)}(\mathrm{ARR}_{(b,p)}(X)) = M$, $1 \leq m < M$, *where* $b = (M-1)p + s$, $0 < s \leq p$.

Recall that $C_{\mathrm{FIFO}(m)}(\mathrm{ARR}(X))$ denotes the FIFO(m)-complexity of periodic arrangements according to Definition 1.10.

The corollary follows from the fact that the lower bound in Theorem 4.1 is assumed by any $w = v_1^{p_1} \cdots v_M^{p_M} \in \mathrm{ARR}(X)$.

In order to analyse the automata LRU(m), $m \geq 1$, we shall first introduce a larger class of paging-automata that contains LRU. This class is distinguished by the property: After each run of some word $w \in X^*$ main-memory contains exactly the same symbols (pages). Thus we reduce the analysis problem for LRU to estimating the number of pages that reside in main-memory during subsequent runs of the same string.

In the sequel we shall denote by $\delta_P(q_0, Y^*) \subset Q_P$ the set of states of some paging-automaton P with input-set Y that can be reached if P is started in q_0.

Definition 4.3. An m-paging-automaton P is recurrent iff for all $w \in Y^*$ and $q \in \delta_P(q_0, Y^*)$: $\tau\delta_P(q, w) = \tau\delta_P(q, w^2)$.

Thus a paging-automaton P is called recurrent if and only if the memory contents associated with states of P are the same after each run of w. Before noting that LRU is recurrent we shall give a general estimate of the costs of processing periodic arrangements by recurrent automata:

Lemma 4.4. *Let P be a recurrrent m-paging-automaton. Then for all $w \in \mathrm{ARR}(X)$ and for all $q \in \delta_P(q_0, Y^*)$:*

$$K_P(\delta_P(q, w), w) \geq M - \left| \bigcap_{t=1}^{L(w)} \tau\delta_P(q, ww_t) \right|,$$

where $w = x_1 \cdots x_{L(w)}$, $w_t = x_1 \cdots x_t$, $1 \leq t \leq L(w)$.

Proof. Let $\kappa_P(x, q, w)$ be defined as in the proof of Theorem 4.1. Let $x \in \bigcap_{t=1}^{L(w)} \tau\delta_P(q, ww_t)$. Since P is recurrent $x \in \tau\delta_P(q, w)$ must hold and hence $\kappa_P(x, \delta_P(q, w), w) = 0$. Let $x \notin \bigcap_{t=1}^{L(w)} \tau\delta_P(q, ww_t)$. We claim: in this case, $\kappa_P(x, \delta_P(q, w), w) \geq 1$. This is obviously so if $x \notin \tau\delta_P(q, w^2)$ since then also $x \notin \tau\delta_P(q, w)$. Hence let $x \in \tau\delta_P(q, w)$. Then there exists some $t, 1 \leq t \leq L(w)$, such that $x \notin \tau\delta_P(q, ww_t)$. On the other hand, by the recurrency of P: $x \in \tau\delta_P(q, w^2)$. Therefore there exists some $t', t < t' \leq L(w)$, such that $x_{t'} = x$ and $x \notin \tau\delta_P(q, ww_{t'-1})$ but $x \in \tau\delta_P(q, ww_{t'})$, i.e. $\kappa_P(x, \delta_P(q, w), w) \geq 1$, whence the lemma. □

Corollary 4.5. *With the assumptions of Lemma 4.4 and $\tau(q) \cap X = \emptyset$:*

$$K_P(q, w^n) \geq n \cdot M - (n-1) \left| \bigcap_{t=1}^{L(w)} \tau\delta_P(q, ww_t) \right|, \quad \text{for all } n > 0.$$

For $w = x_1 \cdots x_{L(w)} \in Y^*$ we now define the mapping $d_w : N \to N \cup \{\infty\}$ as follows:

$$d_w(t) = \begin{cases} t - \max\{t' \mid t' < t,\ x_{t'} = x\}, & \text{if } x_t = x_{t'} \text{ for some } t' < t \\ \infty, & \text{otherwise.} \end{cases}$$

Hence, for $x = x_t$, $d_w(t)$ gives the backward distance to the preceding x in w.

With these notations the next lemma — whose straightforward proof will be omitted — formalizes the fact that LRU actually banishes — if necessary — the least recently used page from main-memory (cf. [3]):

Lemma 4.6. *Let* $P = \mathrm{LRU}(m)$, $m \geq 1$, *and* $w = x_1 \cdots x_{L(w)} \in Y^*$. *Then the following statements are equivalent* ($t = 0, 1, \ldots, L(w) - 1$):

(i) $x_{t+1} \notin \tau\delta_P(q_0, w_t)$,

(ii) $k_P(\delta_P(q_0, w_t), x_{t+1}) = 1$,

(iii) *Either* $d_w(t+1) = \infty$ *or* $|\bigcup_{i=t'}^{t}\{x_i\}| \geq m$, *where* $t' = t + 2 - d_w(t+1)$.

As an immediate consequence of Lemma 4.6 we get:

Corollary 4.7. $P = \mathrm{LRU}(m)$, $m \geq 1$, *is recurrent.*

Let us now consider (b, p)-arrangements again. The solution to the LRU-analysis problem is summarized in the following theorem:

Theorem 4.8. *Let* $P = \mathrm{LRU}(m)$, $1 \leq m < M$; *let* $q \in \delta_P(q_0, Y^*)$ *and let* $b, p \in \mathbf{N}$, $u \in \mathbf{N}_0$ *be such that* $b = (M-1)p + s$, $0 < s \leq b$, *and* $u < m$. *Then the following statements are equivalent*:

(i) $b \leq p^2(m-u) + up$,

(ii) *There exists* $w \in ARR_{(b,p)}(X)$ *such that* $|\bigcap_{t=1}^{b} \tau\delta_P(q, ww_t)| \geq u$ ($w = x_1 \cdots x_b$; $w_t = x_1 \cdots x_t$, $1 \leq t \leq b$).

Because of its length and technicality the proof of this theorem will be given in an appendix.

We note several immediate consequences of Theorem 4.8: In the first place, a straightforward computation yields:

Corollary 4.9. *Under the assumptions of Theorem* 4.8:

$$\left| \bigcap_{t=1}^{b} \tau\delta_P(q, ww_t) \right| \leq \max\left\{0, \left\lfloor \frac{1}{p-1}\left(pm - \frac{b}{p}\right) \right\rfloor\right\} \quad \textit{for all } w \in \mathrm{ARR}_{(b,p)}(X),$$

and this estimate is sharp. ($\lfloor a \rfloor$ *denotes the greatest integer smaller than the real number* a).

In connection with Corollaries 4.5 and 4.7, Corollary 4.9 implies:

Corollary 4.10. *Under the assumptions of Theorem* 4.8 *and with* $\tau(q) \cap X = \emptyset$:

$$K_P(q, w^n) \geq n \cdot M - (n-1) \cdot \max\left\{0, \left\lfloor \frac{1}{p-1}\left(pm - \frac{b}{p}\right)\right\rfloor\right\}$$

for all $w \in \mathrm{ARR}_{(b,p)}(X)$ *and all* $n > 0$. *This estimate is sharp.*

Taking into account Definition 1.10 this yields:

Corollary 4.11. *Under the assumptions of Theorem* 4.8:

$$C_{\mathrm{LRU}(m)}(\mathrm{ARR}_{(b,p)}(X)) = M - \max\left\{0, \left\lfloor \frac{1}{p-1}\left(pm - \frac{b}{p}\right)\right\rfloor\right\}.$$

Comparing this result with Corollary 4.2 we obtain:

Corollary 4.12. *The following statements are equivalent*:
(i) $C_{\mathrm{LRU}(m)}(\mathrm{ARR}_{(b,p)}(X)) < C_{\mathrm{FIFO}(m)}(\mathrm{ARR}_{(b,p)}(X))$,
(ii) $b < p^2(m-1) + p$.

Talking again in terms of paginations of simple loops the last corollary may be put as follows: There exists a p-pagination of a simple loop of length b such that the resulting page-reference-string is processed cheaper by LRU(m) than any other string (resulting from some other pagination) is processed by FIFO(m), if and only if $b < p^2(m-1) + p$.

On the other hand, it can be easily seen that, given the simple-minded pagination (which is optimal for FIFO(m), according to Theorem 4.1), LRU(m) is just as costly as FIFO(m), irrespective of the loop-length b. Hence, our so termed principle of locality which demands to write consecutively accessed addresses on the same page if possible and which yields optimal page-reference-strings for optimal paging-automata as well as for FIFO, does no longer hold for LRU.

The dependence of the costs of LRU-paging of simple loops on the loop-length b, on the page size p and on the main-memory capacity m, is explicitly given in Corollary 4.11. Note that M, the number of pages a loop actually needs, is given by $M = \lceil b/p \rceil$.

Results of this type are certainly of interest to programmers of virtual-memory computers who want to efficiently implement operations involving for instance large matrices: According to the remarks at the end of Section 2, these results show how the way of sequencing address operations influences paging-performance, given a particular pagination and a particular paging-algorithm like LRU or FIFO. And it should be noted that for instance LRU is a widely used paging-algorithm.

In the proof of Theorem 4.8 a (b, p)-arrangement (and hence a p-pagination of a simple loop of length b) that is optimal for LRU(m) will be explicitly constructed.

Appendix

Proof of Theorem 4.8. For the sake of brevity we introduce the following notations: Let $w = x_1 \cdots x_{L(w)}$. The subword $x_t \cdots x_{t'}$, $1 \leq t < t' \leq L(w)$, of w will be denoted by $w_{t,t'}$ and for $w_{t,t'}$ we define $S(w_{t,t'})$ to be the set of symbols occurring in w, i.e.

$$S(w_{t,t'}) = \bigcup_{i=t}^{t'} \{x_i\}.$$

$(ii) \Rightarrow (i)$: Let $w \in \mathrm{ARR}_{(b,p)}(X)$ be such that $|\bigcap_{t=1}^{b} \tau\delta_P(q, ww_t)| \geq u$. Then there exist at least u pairwise distinct elements $y_1, \ldots, y_u$ that are contained in the intersection of the sets $\tau\delta_P(q, ww_t)$, $1 \leq t \leq b$. For $j = 1, \ldots, u$ let $1 \leq t_1^{(j)} < \cdots < t_{k_j}^{(j)} \leq b$ be the instants t, $1 \leq t \leq b$, when $x_t = y_j$. Since w is a (b,p)-arrangement, $1 \leq k_j \leq p$ must hold for all $j = 1, \ldots, u$.

For definiteness we now pick y_1 and count the number of occurrences of some y_j, $j \in \{2, \ldots, u\}$ between any two subsequent occurrences of y_1, i.e. we put:

$$k_j^{\alpha} = |\{t \mid t_{\alpha}^{(1)} < t < t_{\alpha+1}^{(1)}, x_t = y_j\}|$$

for $\alpha = 1, \ldots, k_1 - 1$ and $j = 2, \ldots, u$. We claim:

Claim. *For* $\alpha = 1, \ldots, k_1 - 1$:

$$t_{\alpha+1}^{(1)} - t_{\alpha}^{(1)} - \left(1 + \sum_{j=2}^{u} k_j^{\alpha}\right) \leq p(m-u).$$

Note that the left hand side of this inequality is just the number of times a symbol different from $y_1, \ldots, y_u$ occurs between two subsequent occurrences of y_1.

Proof. To prove this claim let us first consider the case where $k_j^{\alpha} \geq 1$ for all $j = 2, \ldots, u$. Application of Lemma 4.6 then yields:

$$|S(w_{t_{\alpha}^{(1)}+1, t_{\alpha+1}^{(1)}-1})| < m, \quad \text{whence} \quad |S(w_{t_{\alpha}^{(1)}, t_{\alpha+1}^{(1)}})| \leq m.$$

By the special choice of $y_1, \ldots, y_u$ we get: $|S(w_{t_{\alpha}^{(1)}, t_{\alpha+1}^{(1)}}) - \{y_1, \ldots, y_u\}| \leq m - u$ for the number of different symbols other than $y_1, \ldots, y_u$ occuring between $t_{\alpha}^{(1)}$ and $t_{\alpha+1}^{(1)}$. On the other hand, since $w \in \mathrm{ARR}_{(b,p)}(X)$, we have for all $x \in X$: $|\{t \mid t_{\alpha}^{(1)} < t < t_{\alpha+1}^{(1)}, \ x_t = x\}| \leq p$, hence at most $p(m-u)$ times a symbol different from $y_1, \ldots, y_u$ occurs between $t_{\alpha}^{(1)}$ and $t_{\alpha+1}^{(1)}$, but this has been our claim.

Now suppose $k_j^{\alpha} = 0$ for at least one $j \in \{2, \ldots, u\}$. Assume:

$$t_{\alpha+1}^{(1)} - t_{\alpha}^{(1)} - \left(1 + \sum_{j=2}^{u} k_j\right) > p(m-u).$$

This assumption will lead us to a contradiction: For $j \in \{2, \ldots, u\}$ with $k_j^{\alpha} = 0$ let $t_{\max}^{(j)}$ be the last time before $t_{\alpha}^{(1)}$ y_j occurs. Let $s = \min\{t_{\max}^{(j)} \mid k_j^{\alpha} = 0\}$. Then $s = t_{\beta}^{(i)}$ for some $i \in \{2, \ldots, u\}$ and β, $1 \leq \beta \leq k_i$. Since $k_i^{\alpha} = 0$, we have $t_{\beta+1}^{(i)} > t_{\alpha+1}^{(1)}$. Let $k_j'^{\beta}$ be defined as k_j^{α} above, with 1 replaced by i and α by β. Then, by definition of

$s: k_j^{\prime\beta} \geq 1$ for all $j \neq i, j \in \{1, \ldots, u\}$, hence: $\{y_1, \ldots, y_u\} \subset S(w_{t_\beta^{(i)}+1, t_{\beta+1}^{(i)}})$. On the other hand, our assumption implies: $|S(w_{t_\alpha^{(i)}, t_{\alpha+1}^{(i)}}) - \{y_1, \ldots, y_u\}| > m - u$. For the interval $(t_\beta^{(i)}, t_{\beta+1}^{(i)})$ we therefore obtain: $|S(w_{t_\beta^{(i)}, t_{\beta+1}^{(i)}})| > m$, and hence: $|S(w_{t_\beta^{(i)}+1, t_{\beta+1}^{(i)}-1})| \geq m$. By Lemma 4.6, this contradicts the fact $y_i \in \bigcap_{t=1}^{b} \tau\delta_P(q, ww_t)$. In like manner one shows:

$$t_1^{(1)} + (b - t_{k_1}^{(1)}) - \left(1 + \sum_{j=2}^{u} k_j^0\right) \leq p(m-u), \quad \text{where}$$

$$k_j^0 = |\{t \mid t < t_1^{(1)} \quad \text{or} \quad t_{k_1}^{(1)} < t \leq b,\ x_t = y_j\}|, \quad \text{for } j = 2, \ldots, u.$$

Summing up over α, we get:

$$\sum_{\alpha=1}^{k_1-1} (t_{\alpha+1}^{(1)} - t_\alpha^{(1)}) - \left(k_1 - 1 + \sum_{\alpha=1}^{k_1-1} \sum_{j=2}^{u} k_j^\alpha\right)$$

$$+ t_1^{(1)} + (b - t_{k_1}^{(1)}) - \left(1 + \sum_{j=2}^{u} k_j^0\right) \leq k_1 p(m-u).$$

Since $\sum_{\alpha=0}^{k_1-1} k_j^\alpha = k_j$, this yields $b - \sum_{j=1}^{u} k_j \leq k_1 p(m-u)$. The inequality (i) of Theorem 4.8 now follows from the fact that $k_j \leq p$ for $j = 1, \ldots, u$.

(i) $\Rightarrow$ (ii): Let $b \leq p^2(m-u) + up$. An arrangement $w \in \mathrm{ARR}_{(b,p)}(X)$ satisfying (ii) will be constructed: (i) implies: $lp(m-u) \leq b - up \leq (l+1)p(m-u)$ for suitable $l, 1 \leq l < p$. Let $c = b - up - lp(m-u)$; then $c < p(m-u)$, hence $c = u'p + r$ with suitable $u' < m - u$ and $0 \leq r < p$. Furthermore, let $p = el + f$, $e \geq 1$, $0 \leq f < l$.

With l, u', e, f and r thus defined, w is given by:

$$w = 1^e \ldots\ldots\ u^e (u+1)^p \ldots\ldots\ldots\ldots\ldots\ldots\ m^p$$

$$1^e \ldots\ldots\ u^e (m+1)^p \ldots\ldots\ldots\ldots\ldots\ldots (2m-u)^p$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$1^e \ldots\ldots\ u^e ((l-1)m - u + 1)^p \ldots\ldots\ldots\ldots\ (lm-u)^p$$

$$1^f \ldots\ldots\ u^f (lm - u + 1)^p \ldots\ldots\ldots\ldots (lm - u + u')^r (lm - u + u' + 1)^r.$$

Evidently, $w \in \mathrm{ARR}_{(b,p)}(X)$ and between any two subsequent $y \in \{1, \ldots, u\}$ there occur at most $m - 1$ symbols different from y. The claim follows again via Lemma 4.6. □

Remark. The observation essential for the proof of Theorem 4.8 is contained in Lemma 4.6 (iii). We shall conclude by giving an alternative formulation of this lemma, thus relating the analysis problem for paging-automata to the analysis problem for finite acceptors (cf. [4]).

To this effect let $P = (Q, q_0, X, \delta, \tau)$ be some paging-automaton. For any $x \in X$

we associate the acceptor $P^{(x)} = (Q, q_0, X, \delta, F_x)$ with Q, q_0, X, δ as in P and $F_x = \{q \mid q \in \delta(q_0, X^*),\ x \notin \tau(q)\}$ as set of final states. Let $L(x) = \{w \mid w \in X^*,\ \delta(q_0, w) \in F_x\}$ be the set of words accepted by $P^{(x)}$. Then Lemma 4.6 can be rewritten as follows:

Lemma. *Let* $P = \mathrm{LRU}(m)$, $m \geq 1$. *Then for all* $x \in X$:

$$L(x) = (X - \{x\})^* \cup X^* \cdot \{x\} \cdot J(x),$$

where $J(x) = \{w \mid w \in (X - \{x\})^*,\ |S(w)| \geq m\}$.

The following problem arises: Find a procedure that for any paging-automaton P (of a given class) yields an optimal (b, p)-arrangement w, given the family of sets $\{L(x) \mid x \in X\}$!

Acknowledgement

This research has mainly been done while I was a member of the Forschungsgruppe Automatentheorie und Formale Sprachen at TH Darmstadt. I am grateful to H. Walter, leader of that group, and to my colleagues for providing a stimulating atmosphere.

References

[1] A.V. Aho, P.J. Denning, J.D. Ullman, Principles of optimal page replacement, *J. ACM* **18** (1) (1971) 80–93.
[2] L.A. Belady, A study of replacement algorithms for virtual storage computers, *IBM Systems J.* **5** (2) (1966) 78–101.
[3] E.G. Coffman, P.J. Denning, *Operating Systems Theory* (Prentice-Hall, Englewood Cliffs, 1973).
[4] G. Hotz, H. Walter, *Automatentheorie und formale Sprachen* II (Bibliographisches Institut, Mannheim, 1969).
[5] A.C. McKellar, E.G. Coffman, Organizing matrices and matrix operations for paged memory systems, *Commun. ACM* **12** (3) (1969) 153–165.
[6] H.-G. Stork, Zur Seitenwechselkomplexität einfacher Referenzenstrukturen, Dissertation, Darmstadt (1975).
[7] H.-G. Stork, A theorem on linear diophantine equations and the paging-complexity of loop-chains, *Computing* **17** (1976) 105–113.