The final publication is available at the TMNA webpage via https://doi.org/10.12775/TMNA.2015.014
Postprint of: Pilarczyk P., Graff G., An algorithmic approach to estimating the minimal number of periodic points for smooth self-maps of simply-connected manifolds, Topological Methods in Nonlinear Analysis, Vol. 45, Iss. 1 (2015), pp. 273-286

This is a postprint of the paper published in
Topol. Methods Nonlinear Anal., Vol. 45, No. 1 (2015), 273–286.

AN ALGORITHMIC APPROACH TO ESTIMATING THE MINIMAL NUMBER OF PERIODIC POINTS FOR SMOOTH SELF-MAPS OF SIMPLY-CONNECTED MANIFOLDS

GRZEGORZ GRAFF AND PAWEŁ PILARCZYK

Abstract. For a given self-map f of M, a closed smooth connected and simply-connected manifold of dimension $m \geq 4$, we provide an algorithm for estimating the values of the topological invariant $D_r^m[f]$, which equals the minimal number of r-periodic points in the smooth homotopy class of f. Our results are based on the combinatorial scheme for computing $D_r^m[f]$ introduced by G. Graff and J. Jezierski [J. Fixed Point Theory Appl. 13 (2013), 63–84]. An open-source implementation of the algorithm programmed in C++ is publicly available at `http://www.pawelpilarczyk.com/combtop/`.

1. Introduction

Let M be a closed smooth connected and simply-connected manifold of dimension $m \geq 4$. The problem of finding the minimal number of periodic points in the homotopy class of a self-map f of M is one of the central questions in periodic point theory.

J. Jezierski and the first author have recently developed a smooth version of Nielsen periodic point theory, searching for

$$\min\{\#\mathrm{Fix}(g^r) : g \overset{s}{\sim} f\},$$

where $g \overset{s}{\sim} f$ means that the maps g and f are C^1-homotopic. This line of research is based on the construction of smooth topological invariants that are the counterpart of the invariant $NF_r(f)$ which exists in the classical

2000 *Mathematics Subject Classification.* Primary 37C25, 55M20; Secondary 37C05.

Key words and phrases. Periodic points, Nielsen number, Lefschetz number, fixed point index, smooth maps, minimal number of periodic points.

continuous category (cf. [12, 13]). In [6, 7], two counterparts of $NF_r(f)$ were defined for the smooth category: $D_r^m[f]$ for simply-connected manifolds and its generalization $NJD_r^m[f]$ for those manifolds that are not simply-connected.

In this paper, we restrict our attention to simply-connected manifolds. Then the only obstacle to minimizing the number of periodic points comes from their fixed point indices. Nevertheless, the computation of the smooth invariant $D_r^m[f]$ remains a highly nontrivial task. So far, the value of $D_r^3[f]$ was found in some special cases only; namely, it was determined for self-maps of S^3 [8], and for self-maps satisfying some additional conditions on $S^2 \times I$ [7] and on the two-holed 3-dimensional closed ball [4]. In higher dimensions (that is, for $m \geq 4$), $D_r^m[f]$ was only found in the case in which r is a product of different primes ($m = 4$ in [9] and $m \geq 4$ in [5]).

Since the general case seems to be too difficult to tackle, in what follows we restrict our attention to odd r and to maps with fast growth of the Lefschetz numbers of iterations, that is, satisfying our standing assumptions (2.1). Simple examples of such maps are self-maps of S^m with degree d, where $|d| > 1$; more examples can be found in [15].

Under these assumptions, we apply the general combinatorial scheme introduced in [5] to develop an algorithm and its implementation (a computer program) which provide estimates for the invariant $D_r^m[f]$ for $m \geq 4$ and for an arbitrary odd r. Let us point out that such estimates provide important information concerning periodic points. Namely, the estimate $\alpha \leq D_r^m[f] \leq \beta$ implies that: (i) every smooth map g smoothly homotopic to f has at least α r-periodic points, (ii) there exists a smooth map g smoothly homotopic to f that has at most β r-periodic points.

The paper is organized in the following way. In Section 2, we introduce the invariant $D_r^m[f]$ and define the class of maps that are taken into consideration in the paper. In Section 3, we formulate the combinatorial problem for obtaining an upper bound on $D_r^m[f]$ and we recall known estimates for $D_r^m[f]$. In Section 4, we introduce terminology and definitions necessary to describe the proposed solution to this problem. In Section 5, we introduce algorithms that can be used for solving the problem. In Section 6, we give brief introduction to the software implementation of the algorithms, and we discuss some specific examples.

2. Definition of the invariant $D_r^m[f]$

Sequences of local indices of smooth maps play a crucial role in the definition of $D_r^m[f]$, and we will call them DD^m sequences (Dold differential sequences).

By p-orbit we mean an orbit with minimal period p.

Definition 2.1. A sequence of integers $\{c_n\}_{n=1}^{\infty}$ is called a $DD^m(p)$ *sequence* if there exists an open set $U \subset \mathbb{R}^m$, a C^1 map $\phi\colon U \to \mathbb{R}^m$, and an isolated p-orbit P of ϕ such that $c_n = \mathrm{ind}(\phi^n, P)$. (Notice that $c_n = 0$ if n is not a multiple of p). For a given positive integer r, the finite sequence $\{c_n\}_{n|r}$ is called a $DD^m(p|r)$ *sequence* if $c_n = \mathrm{ind}(\phi^n, P)$ for all $n > 0$ such that $n|r$.

For a fixed integer $r \geq 1$, we define $D_r^m[f]$ as the minimal decomposition of the sequence of Lefschetz numbers of iterations into $DD^m(p|r)$ sequences.

Definition 2.2. Let $\{L(f^n)\}_{n|r}$ be a finite sequence of Lefschetz numbers. We decompose $\{L(f^n)\}_{n|r}$ into the sum:

$$L(f^n) = c_1(n) + \ldots + c_s(n),$$

where c_i is a $DD^m(l_i|r)$ sequence for $i = 1, \ldots, s$. Each such decomposition determines the number $l = l_1 + \ldots + l_s$. We define the number $D_r^m[f]$ as the smallest l which can be obtained in this way.

It turns out that $D_r^m[f]$ is a topological invariant equal to the minimal number of r-periodic points in the smooth homotopy class of f. The reader may consult [5] for topological background related to the construction of the invariant.

Theorem 2.3. ([7]) *Let M be a closed smooth compact connected and simply-connected manifold of dimension $m \geq 3$, and let $r \in \mathbb{N}$. Then*

$$D_r^m[f] = \min\{\#\mathrm{Fix}(g^r) : g \overset{s}{\sim} f\}.$$

Note that it was proved in [5] that for $m \geq 4$, one may equivalently use $DD^m(1|r)$ sequences in Definition 2.2 of $D_r^m[f]$.

Definition 2.4. Let $k > 0$ be an integer. The *basic sequence* $\mathrm{reg}_k(n)$ is defined as

$$\mathrm{reg}_k(n) = \begin{cases} k & \text{if} \quad k \mid n, \\ 0 & \text{if} \quad k \nmid n. \end{cases}$$

The Lefschetz numbers of iterations (and also sequences of indices of iterations) can be represented in the form of so-called periodic expansion, as an integral combination of basic periodic sequences (cf. [14]); namely:

$$L(f^k) = \sum_{k=1}^{\infty} b_k \mathrm{reg}_k(n),$$

where $b_n = \frac{1}{n}\sum_{k|n} \mu(k) L(f^{(n/k)})$, and $\mu\colon \mathbb{N} \to \mathbb{Z}$ is the Möbius function defined by the following three properties: $\mu(1) = 1$, $\mu(k) = (-1)^s$ if k is a product of s different primes, $\mu(k) = 0$ otherwise. Note that all the coefficients b_n are integers (cf. [1, 2]).

Let us fix a natural number r. For the divisors of r, we represent the sequence of Lefschetz numbers of iterations in the form of periodic expansion

$$L(f^n) = \sum_{k|r} b_k \mathrm{reg}_k(n).$$

In the remainder of the paper, we are going to work under the following

(2.1) **Standing Assumptions:**

- **(I)** $f\colon M \to M$ is a smooth self-map of a smooth closed connected and simply-connected m-manifold M, where $m \geq 4$.
- **(II)** r is odd and $b_k \neq 0$ for all $k \neq 1$ dividing r.

All the forms of $DD^m(1)$ sequences were determined in [10, 11] (cf. also [3]). The comparison of the periodic expansion of $\{L(f^n)\}_{n|r}$ (for f satisfying our Standing Assumptions (2.1)) with the periodic expansions of $DD^m(1)$ sequences makes it possible to determine a purely combinatorial formula for $D_r^m[f]$.

3. Statement of the combinatorial problem

We use the combinatorial framework introduced in [5] in order to estimate the invariant $D_r^m[f]$. It is important to note that the only data that we need for that purpose is the following:

Input data: $r > 0$ (odd), $m \geq 4$ and $b_1 = L(f)$, all integers.

This input is preprocessed as follows. Define $s > 0$ such that $m = 2s$ if m is even or $m = 2s + 1$ otherwise. Find a decomposition of r into the product of prime numbers, that is, find a set of distinct prime numbers $p_1, \ldots, p_l$ ($l > 0$) such that $r = p_1^{a_1} \cdots p_l^{a_l}$, where $a_i > 0$ for all $i \in \{1, \ldots, l\}$. Note that after a rearrangement (if necessary), there exists $w \in \{0, \ldots, l\}$ such that $a_i > 1$ if $i \leq w$ and $a_i = 1$ if $i > w$; in other words, $r = p_1^{a_1} \cdots p_w^{a_w} \cdot p_{w+1} \cdots p_l$. After this preprocessing, the following data is passed for the algorithmic procedures:

Combinatorial input: $s > 1$, $a_1, \ldots, a_l > 0$, all integers.

The combinatorial procedure for finding an upper estimate d for $D_r^m[f]$ is based on Definition 4.1 of an $(s, a_1, \ldots, a_l, v)$-collection as well as Definition 4.2 of a sharp $(s, a_1, \ldots, a_l, v)$-collection, and also on Theorem 3.2. The core Algorithm 5.1 makes an attempt to construct a possibly small (in terms of minimizing v) $(s, a_1, \ldots, a_l, v)$-collection (but not necessarily optimal), and to try keep it a sharp collection if possible. Note that b_1 is not part of the input of this algorithm, but is used later for the interpretation of its output. The algorithm returns the following quantities:

Combinatorial output: An integer $v > 0$ and a binary value $h \in \{\texttt{true}, \texttt{false}\}$, such that an $(s, a_1, \ldots, a_l, v)$-collection (see Definition 4.1) exists, and $h = \texttt{true}$ if a sharp $(s, a_1, \ldots, a_l, v)$-collection (see Definition 4.2) was found or $h = \texttt{false}$ if the existence of such a sharp collection was not confirmed.

The following function translates the combinatorial output combined with the input data to an upper bound $d > 0$ for $D_r^m[f]$ as follows:

Definition 3.1. Let $m \geq 4$, b_1, and $v > 0$ be integers. Let $h \in \{\texttt{true}, \texttt{false}\}$. Define $\alpha(v, h, m, b_1)$ as follows:

(1) If m is even: If $h = \texttt{true}$ or $|b_1| = v$ then $\alpha(v, h, m, b_1) := v$, otherwise $\alpha(v, h, m, b_1) := v + 1$.
(2) If m is odd: If $h = \texttt{true}$ or $|b_1| \leq v$ then $\alpha(v, h, m, b_1) := v$, otherwise $\alpha(v, h, m, b_1) := v + 1$.

Output data: $d := \alpha(v, h, m, b_1)$.

Theorem 3.2 (see Theorem 5.5 in [5])**.** *Let v_0 be the smallest integer such that there exists an $(s, a_1, \ldots, a_l, v)$-collection (see Definition 4.1), and $h_0 = \texttt{true}$ if and only if there exists a sharp $(s, a_1, \ldots, a_l, v)$-collection (see Definition 4.2). Then $D_r^m[f] = \alpha(v_0, h_0, m, b_1)$.*

Corollary 3.3. *The number $d = \alpha(v, h, m, b_1)$ is an upper bound for $D_r^m[f]$.*

In addition to the upper bound d for $D_r^m[f]$, we take into account the following estimates:

Theorem 3.4 (see Theorem 6.6 in [5])**.** *If $l \geq s$ and $a_i = 1$ for all $i \in \{1, \ldots, l\}$, that is, if r is a product of l distinct odd prime numbers, then define two integers: $k \geq 0$ and $R \in \{1, \ldots, s\}$, such that $l = ks + R$. Then $v_0 = (2^l - 2^R)/(2^s - 1) + 1$, and $h_0 = \texttt{true}$ if and only if $s \nmid l$.*

Theorem 3.5 (see Theorem 7.3 in [5])**.** *Assume that $r = p_1^{a_1} \cdots p_w^{a_w} \cdot p_{w+1} \cdots p_l$, where $w + s \leq l$. Denote the smallest integer greater than or equal to x by $\lceil x \rceil$; this is so-called ceiling function. Then $G \leq v_0 \leq G + H$, where*

$$G = \left\lceil \left(2^w + (a_1 + 1) \cdots (a_w + 1) \cdot (2^{l-w} - 1) - 1\right)/(2^s - 1) \right\rceil$$

$$H = (a_1 + 1) \cdots (a_w + 1) - 2^w + \left\lceil \left((a_1 + 1) \cdots (a_w + 1) - 2^w\right)/s \right\rceil.$$

Theorem 3.6 (see Lemma 7.1 in [5])**.** $v_0 \geq \left\lceil \left((a_1 + 1) \cdots (a_l + 1) - 1\right)/(2^s - 1) \right\rceil$.

4. Notation and definitions

Denote the set of natural numbers including 0 by $\mathbb{N}_0$. The cardinality of a set A will be denoted by $\#A$.

A *multiset* is a generalization of a set which allows each element to appear a specific number times. Formally, a multiset A may be defined as a function $A\colon I \to \mathbb{N}_0$, where I is some underlying domain. We say that $a \in A$ if $a \in I$ and $A(a) > 0$. The *multiplicity* of a in A is either 0 if $a \notin A$ or $A(a)$ otherwise. If the multiplicity of a in A is positive then we say that a is an *element* of A and we denote it as $a \in A$.

The notion of a multiset is very natural do describe (positive) primary factors of a positive integer number. For example, $6125 = 5 \cdot 5 \cdot 5 \cdot 7 \cdot 7$, so the primary factors of 6125 can be informally denoted as $\{5, 5, 5, 7, 7\}$, which corresponds to the multiset $f\colon \{5, 7\} \to \mathbb{N}_0$ such that $f(5) = 3$ and $f(7) = 2$.

If we consider multisets in a fixed (large) domain D then it is convenient to restrict our attention to functions $D \to \mathbb{N}_0$. In other words, for $I \subset D$, each multiset $A\colon I \to \mathbb{N}_0$ can be extended to a function $D \to \mathbb{N}_0$ by letting $A(a) := 0$ for $a \in D \setminus I$. The common domain of all the multisets that are under consideration simplifies formal definitions.

The *union* $A \cup B$ of multisets $A\colon D \to \mathbb{N}_0$ and $B\colon D \to \mathbb{N}_0$ is another multiset $C\colon D \to \mathbb{N}_0$ in which each element of A and B appears with the multiplicity that is the maximum of the multiplicities with which it appears in A and in B. Formally, $C(a) := \max\{A(a), B(a)\}$. The union of multisets of primary factors of two integer numbers is a multiset of their least common multiple.

We say that a multiset $A\colon D \to \mathbb{N}_0$ is a *subset* of $B\colon D \to \mathbb{N}_0$ if $A(a) \leq B(a)$ for all $a \in D$. The multiset of (positive) primary factors of a positive integer x is a subset of the multiset of primary factors of another positive integer y if and only if x divides y.

We shall use the terms "family" and "collection" in the meaning of "set" in order to indicate the following gradation: a set of multisets will be called a family of multisets, and a set of families of multisets will be called a collection of families of multisets (which sounds much better than a set of sets of multisets).

If A is a family of sets then the union of these sets will be denoted by $\bigcup A = \bigcup_{B \in A} B$.

All the sets and multisets that appear in the algorithms are assumed to be finite, even if this is not written explicitly.

In what follows, we shall assume $D := \mathbb{N}_0$ for all multisets, and we shall explicitly define multisets on subsets of $\mathbb{N}_0$ to make it clear which elements may appear in the multisets with positive multiplicity.

Let $l > 0$ be an integer. Given positive integers $a_1, \ldots, a_l$, denote the multiset $\mathcal{C}\colon \{1, \ldots, l\} \ni p \mapsto a_p \in \mathbb{N}_0$ by $\mathcal{C}(a_1, \ldots, a_l)$, or by $\mathcal{C}$ alone if the numbers $a_1, \ldots, a_l$ are clear from the context.

Definition 4.1. Let s, $a_1, \ldots, a_l$, and v be positive integers ($l > 0$). An $(s, a_1, \ldots, a_l, v)$-*collection* is a collection $A_1, \ldots, A_v$ of families of multisets $B_j^k\colon \{1, \ldots, l\} \to \mathbb{N}_0$, $B_j^k \in A_j$, $k = 1, \ldots, \#A_j$, where $\#A_j \leq s$, $j = 1, \ldots, v$, such that for every multiset $C \subset \mathcal{C}(a_1, \ldots, a_l)$ there exists $i \in \{1, \ldots, v\}$ and a subset D of A_i such that $C = \bigcup D$.

Definition 4.2. An $(s, a_1, \ldots, a_l, v)$-collection is called *sharp* if $\#A_i < s$ for some $i \in \{1, \ldots, v\}$.

Denote the set of all the permutations of $\{1, \ldots, k\}$, where $k > 0$ is an integer, by $\mathcal{P}(k)$.

5. Algorithmic approach to solving the combinatorial problem

Computation of the smallest number v_0 for which there exists an $(s, a_1, \ldots, a_l, v_0)$-collection, and also determining whether there exists a sharp $(s, a_1, \ldots, a_l, v_0)$-collection or not, seems to be a highly nontrivial task, even for small values of the parameters. Checking all the possible combinations of collections of families of multisets and determining which of them satisfy the desired conditions is beyond computation due to the huge number of such combinations. Therefore, we provide an algorithmic method for constructing some $(s, a_1, \ldots, a_l, v)$-collection, possibly small, but not necessarily optimal.

5.1. **The greedy strategy.** Our construction is based upon the "greedy" strategy. The idea is to consider all the possible non-empty multisets contained in $\mathcal{C}(a_1, \ldots, a_l)$, one by one, and to build the collection $\mathcal{A} = \{A_1, \ldots, A_v\}$ gradually. At each step, a multiset C is considered, and a previously constructed collection is modified (if necessary) so that there exists an $i \in \{1, \ldots, v\}$ and a subset D of A_i such that $C = \bigcup D$. The undeniable advantage of the greedy strategy is that it turns a computationally expensive problem of checking all the possibilities into a step-by-step approach, in which a solution is constructed gradually. A disadvantage of such an approach is that even if the modifications are optimal at each step, the final result may not be optimal.

Technically, the family of all the non-empty multisets which are subsets of $\mathcal{C}$ can be easily constructed, e.g., by iterating a set of l counters that would count the multiplicity of the corresponding elements from 0 to a_p. Let S denote the number of all the possible multisets of $\mathcal{C}$.

Since the order in which the multisets $C_1, \ldots, C_S \subset \mathcal{C}$ are taken may have profound effect on the result achieved by a greedy-style algorithm, it is worth to consider several different orders (e.g., using some randomization) and run the algorithm on each re-ordered sequence of the multisets. (See also Conjecture 5.5.)

The problem of modifying the collection $\mathcal{A}$ constructed so far in order to have a specific additional multiset $C \subset \mathcal{C}$ represented by means of the union of a subset of one of the sets in the collection $\mathcal{A}$ may also be solved in several ways. The algorithm should aim at minimizing the size of $\mathcal{A}$. A simple solution which we propose is to add the set C to one of the sets if there is one whose size does not yet exceed s, or to add a new set $\{C\}$ to the collection. However, a more effective algorithm might make more thorough modifications of the collection constructed so far, but it must preserve the condition that all the previously processed multisets are still represented, each as the unions of some subsets of one of the sets in the collection.

5.2. **The algorithms.** We introduce a sequence of algorithms that combined together provide a method for automatic construction of a possibly small $(s, a_1, \ldots, a_l, v)$-collection. Pseudo-code of each algorithm is preceded by its outline and brief explanation of its purpose.

Algorithm 5.1 below computes a possibly small upper bound v for the number v_0 and a corresponding indicator h of whether a sharp collection was found. This algorithm depends on a method for constructing a family of all the non-empty multisets contained in a given multiset, which is straightforward, and on the procedure `ConstructCollection` that constructs an $(s, a_1, \ldots, a_l, v)$-collection, which is discussed in the sequel (see Algorithm 5.2). As an input, it takes the parameters of the desired $(s, a_1, \ldots, a_l, v)$-collections, and a set of permutations which are used for different runs of the construction procedure. The algorithm returns the upper bound v for v_0 that can be inferred from the constructed collections, and the corresponding indicator h.

Algorithm 5.1 (Procedure `ComputeUpperBound`).
INPUT:
$n > 0$, $a_1, \ldots, a_l > 0$ — integers;
$\mathcal{I} \subset \mathcal{P}(S)$ — a non-empty set of permutations of $\{1, \ldots, S\}$, where S is the number of all the non-empty multisets contained in $\mathcal{C}(a_1, \ldots, a_l)$;
PSEUDOCODE:
construct an ordered family $(C_1, \ldots, C_S)$ of all the non-empty multisets contained in $\mathcal{C}(a_1, \ldots, a_l)$;
`for each` $\rho \in \mathcal{I}$ `do`

$\mathcal{A}_\rho :=$ `ConstructCollection` $(s, C_{\rho(1)}, \ldots, C_{\rho(S)})$;
$v := \min\{\#\mathcal{A}_\rho \mid \rho \in \mathcal{I}\}$;
$h :=$ `true` iff there exists $\rho \in \mathcal{I}$ such that $\#\mathcal{A}_\rho = v$
and there exists $A \in \mathcal{A}_\rho$ such that $\#A < s$;
OUTPUT:
v — an integer number;
$h \in \{\texttt{true}, \texttt{false}\}$.

Algorithm 5.2 which follows is a greedy framework for constructing an $(s, a_1, \ldots, a_l, v)$-collection, given an ordered family of all the non-empty multisets that are contained in $\mathcal{C}(a_1, \ldots, a_l)$. It depends on the procedure `ProcessMultiset`, discussed below (see Algorithm 5.3), that processes a single multiset in that ordered family and modifies the collection $\mathcal{A}$ if necessary. The algorithm returns an $(s, a_1, \ldots, a_l, v)$-collection constructed on the basis of the input data.

Algorithm 5.2 (Procedure `ConstructCollection`).
INPUT:
$s > 0$ — natural numbers;
$C_1, \ldots, C_q \colon \{1, \ldots, l\} \to \mathbb{N}_0$ — multisets;
PSEUDOCODE:
$\mathcal{A} := \emptyset$;
`for` $i :=$ 1 `to` q `do`
$\mathcal{A} :=$ `ProcessMultiset` $(\mathcal{A}, s, C_i)$;
OUTPUT:
$\mathcal{A}$ — a collection of families of multisets.

Algorithm 5.3 is the core of the greedy approach to the problem. It performs one step of the overall algorithm. It analyzes a collection $\mathcal{A} = \{A_1, \ldots, A_v\}$ of families of multisets and verifies whether it contains a family A_i, with $i \in \{1, \ldots, v\}$, such that the union of one of its subsets $D \subset A_i$ equals the given multiset C (which is assumed to be a subset of $\mathcal{C}(a_1, \ldots, a_l)$). This verification can be easily done by checking all the possible subsets of each of the sets in $\mathcal{A}$, although it might also be done in a slightly more efficient way. If the collection $\mathcal{A}$ does not contain such a family then the algorithm runs the procedure `ModifyCollection` (see Algorithm 5.4) that modifies $\mathcal{A}$ in such a way that this condition is satisfied for this and for all the previously queried multisets. This procedure may modify the sets $A_i \in \mathcal{A}$ or may add new sets to the collection $\mathcal{A}$, but it must preserve the condition $\#A \leq s$ for all $A \in \mathcal{A}$.

Algorithm 5.3 (Procedure `ProcessMultiset`).
INPUT:
$\mathcal{A}$ — a collection of families of multisets;

$s > 0$ — an integer such that $\#A \leq s$ for all $A \in \mathcal{A}$;
$C \subset \mathcal{C}(a_1, \ldots, a_l)$ — a multiset;
PSEUDOCODE:
`for each` $A \in \mathcal{A}$ `do`
`if` $C = \bigcup D$ for some $D \subset A$ `then`
`return` $\mathcal{B} := \mathcal{A}$;
$\mathcal{B}$:= `ModifyCollection` ($\mathcal{A}$, s, C);
OUTPUT:
$\mathcal{B}$ — a collection of families of multisets;

Algorithm 5.4 implements the procedure `ModifyCollection` in the simplest possible way, and it adds C to A_v if $\#A_v < s$, or adds $A_{v+1} := \{C\}$ to the collection $\mathcal{A}$ otherwise. Note that there is a lot of room for improvement here, but we do not discuss the details in this paper.

Algorithm 5.4 (Procedure `ModifyCollection`).
INPUT:
$\mathcal{A} = \{A_1, \ldots, A_v\}$ — a collection of families of multisets;
$s > 0$ — an integer such that $\#A \leq s$ for all $A \in \mathcal{A}$;
$C \subset \mathcal{C}(a_1, \ldots, a_l)$ — a multiset;
PSEUDOCODE:
`if` $\#A_v < s$ `then`
$\mathcal{B} := \{A_1, \ldots, A_v \cup \{C\}\}$;
`else`
$\mathcal{B} := \{A_1, \ldots, A_v, \{C\}\}$;
OUTPUT:
$\mathcal{B}$ — a collection of families of multisets;

5.3. **Optimality of the result.** Although the strategy suggested in Algorithm 5.4 is very simple and straightforward, we speculate that it may actually provide the optimal result if one tries applying it to all the non-empty subsets of $\mathcal{C}(a_1, \ldots, a_l)$ in all the possible orders.

Conjecture 5.5. Algorithms 5.1–5.4 called with $\mathcal{I} = \mathcal{P}(l)$, provide the smallest possible integer v_0 for which there exists a $(s, a_1, \ldots, a_l, v_0)$-collection. Moreover, if there exists a sharp $(s, a_1, \ldots, a_l, v_0)$-collection then the algorithms return $h =$ `true`.

Unfortunately, even if Conjecture 5.5 was true, this fact would only have a purely theoretical meaning, because the cost of doing the computation for all the possible permutations of the family of all the non-empty multisets contained in $\mathcal{C}(a_1, \ldots, a_l)$ is prohibitive even for relatively small input data.

6. Software implementation and examples

The algorithms discussed in Section 5 have been implemented in the C++ programming language and are publicly available at the address `http://www.pawelpilarczyk.com/combtop/`. We first briefly discuss the features of the software, and then show some results of its application.

The main program is called `minper` and its purpose is to construct a possibly small and sharp $(s, a_1, \dots, a_l, v)$-collection using the algorithms introduced in Section 5, so as to minimize the upper bound d for $D_r^m[f]$ obtained in the output. The program also computes the other bounds given in Theorems 3.4, 3.5 and 3.6 whenever applicable, and in this way provides complete information on the reliable estimates for the invariant $D_r^m[f]$ that are discussed in this paper.

The source code is compatible with the GNU C++ compiler for optimal portability and can be compiled with ease; precompiled binaries for a few systems are also provided. The program `minper` is a command-line utility, that is, it must be run from a text terminal emulator (a.k.a. command prompt). The program accepts different forms of input parameters, either the general input ($m \geq 4$, $r > 0$ odd, and b_1), or the combinatorial input ($s > 1$, $a_1, \dots, a_l > 0$), or a combination of both ($m \geq 4$, $a_1, \dots, a_l > 0$, and optionally b_1). The program generates all the non-empty multisets contained in $\mathcal{C}(a_1, \dots, a_l)$ and can either take all the permutations of these multisets (which is feasible for small numbers only), run the construction procedure for the identity permutation only, or use pseudo-random permutations (their number can be requested at program's launch). Depending on the information provided on the input, the program either shows the combinatorial output $v > 0$ and h alone, or also displays an upper bound d on $D_r^m[f]$ that follows from Corollary 3.3. The amount of information shown by the program can be easily controlled by setting the verbosity parameter. Technical description of the program (such as the command line syntax) is provided at the project's website.

Table 1 shows a few selected results obtained by the program, in comparison with the bounds known from Theorems 3.5 and 3.6. 10,000 pseudo-random permutations were passed to Algorithm 5.1, or all the possible permutations if their number was smaller than 10,000. Computations were conducted for all the values of $s \in \{2, \dots, 9\}$, $l \in \{1, \dots, 4\}$, and $a_i \in \{1, \dots, 4\}$ for all $i = 1, \dots, l$. Due to the huge amount of collected data, only a few representative results are included in Table 1, and a complete summary of these computations as well as the actual information displayed by the program is available from the project's website. The number v computed by the program was always within the bounds provided by Theorems 3.5 and 3.6. The computation time given in the table

TABLE 1. Results obtained by the program for sample values of $s \geq 2$ and $a_1, \ldots, a_l > 0$ using 10,000 pseudo-random permutations, together with bounds known from Theorems 3.5 and 3.6, and the computation time.

s	$a_1, \ldots, a_l$	result	known bounds	time [sec]
2	$1, 1, 1, 2$	9, true	$8 \leq v \leq 10$	0.54
2	$1, 1, 2$	4, false	$4 \leq v \leq 6$	0.14
2	$4, 4, 4, 4$	259, false	$208 \leq v$	299
3	$1, 1, 1, 4$	9, true	$6 \leq v \leq 10$	1.70
3	$1, 1, 3$	3, true	$3 \leq v$	0.28
3	$4, 4, 4, 4$	149, true	$90 \leq v$	365
4	$1, 1, 4$	2, false	$2 \leq v$	0.58
4	$2, 3, 3, 4$	38, true	$16 \leq v$	84.3
4	$4, 4, 4, 4$	103, true	$42 \leq v$	554
5	$1, 2, 2, 2$	4, true	$2 \leq v$	6.91
5	$1, 3$	1, true	$1 \leq v$	0.00
5	$4, 4, 4, 4$	75, true	$21 \leq v$	849
6	$2, 2, 2, 2$	4, true	$2 \leq v$	25.1
6	$2, 2, 4$	2, false	$1 \leq v$	7.58
6	$2, 3, 3$	3, false	$1 \leq v$	8.53
7	$1, 1, 1, 2$	1, true	$1 \leq v$	0.00
7	$1, 1, 4, 4$	4, false	$1 \leq v$	64.9
7	$4, 4, 4$	7, true	$1 \leq v$	96.4
8	$1, 2, 3, 4$	2, false	$1 \leq v$	157
8	$2, 3, 4, 4$	10, true	$2 \leq v$	924
8	3	1, true	$1 \leq v$	0.00
9	$2, 2, 2, 4$	2, true	$1 \leq v$	338
9	$3, 3, 3, 3$	6, true	$1 \leq v$	1171
9	$3, 4, 4, 4$	11, true	$1 \leq v$	4344

was measured on a relatively modern PC (Quad-Core AMD Opteron™ Processor 8360 SE at 2.5 GHz).

Table 2 illustrates results obtained by the program in the case in which r is a product of distinct primes and $l \geq s$, and thus the exact answer is known by Theorem 3.4. 10,000 pseudo-random permutations were passed to Algorithm 5.1, or all the possible permutations if their number was smaller than 10,000. A comparison of known and computed values indicates that in many cases the number of pseudo-random permutations considered was not satisfactory. Since it was in fact a tiny fraction of the number of all the possible permutations, this does not yet suggest

TABLE 2. Results obtained by the program for $a_1 = \ldots = a_l = 1$ and $l \geq s$, where $l \in \{1, \ldots, 8\}$ and $s \in \{2, \ldots, 8\}$.

s	l	computed	known
2	2	1, false	1, false
2	3	3, true	3, true
2	4	6, true	5, false
2	5	11, false	11, true
2	6	24, true	21, false
2	7	50, false	43, true
2	8	104, false	85, false
3	3	1, false	1, false
3	4	3, true	3, true
3	5	6, true	5, true
3	6	14, true	9, false
3	7	30, true	19, true
3	8	61, true	37, true
4	4	1, false	1, false
4	5	3, false	3, true
4	6	7, false	5, true
4	7	15, false	9, true
4	8	31, false	17, false
5	5	1, false	1, false
5	6	3, false	3, true
5	7	7, false	5, true
5	8	15, false	9, true
6	6	1, false	1, false
6	7	3, false	3, true
6	8	7, false	5, true
7	7	1, false	1, false
7	8	3, false	3, true
8	8	1, false	1, false

that Conjecture 5.5 is false. Unfortunately, the huge number of all the permutations makes it practically impossible to refute Conjecture 5.5 by brute force method (checking all the permutations).

Acknowledgments. This research was supported by the Portuguese Foundation for Science and Technology (FCT) through the Pluriannual

Funding Programme for the Research Centre of Mathematics (CMAT) at the University of Minho, Braga, Portugal.

References

1. I. K. Babenko, S. A. Bogatyi, *The behavior of the index of periodic points under iterations of a mapping*, Math. USSR Izv. **38** (1992), 1–26.
2. A. Dold, *Fixed point indices of iterated maps*, Invent. Math. **74** (1983), 419–435.
3. S. N. Chow, J. Mallet-Paret, J. A. Yorke, *A periodic orbit index which is a bifurcation invariant*, Geometric dynamics (Rio de Janeiro, 1981), 109-131, Springer Lecture Notes in Math. 1007, Berlin 1983.
4. G. Graff, *Minimal number of periodic points for smooth self-maps of two-holed 3-dimensional closed ball*, Topol. Methods Nonlinear Anal. **33** (2009), no. 1, 121–130.
5. G. Graff and J. Jezierski, *Combinatorial scheme of finding minimal number of periodic points for smooth self-maps of simply connected manifolds*, J. Fixed Point Theory Appl. 13 (2013), no. 1, 63–84.
6. G. Graff and J. Jezierski, *Minimizing the number of periodic points for smooth maps. Non-simply connected case,* Topology Appl. **158** (2011), no. 3, 276–290.
7. G. Graff and J. Jezierski, *Minimal number of periodic points for C^1 self-maps of compact simply-connected manifolds*, Forum Math. **21** (2009), no. 3, 491–509.
8. G. Graff and J. Jezierski, *Minimal number of periodic points for smooth self-maps of S^3*, Fund. Math. **204** (2009), 127-144.
9. G. Graff and J. Jezierski, *Minimization of the number of periodic points for smooth self-maps of closed simply-connected 4-manifolds*, Discrete Contin. Dyn. Syst. Supl. (2011), 523-532.
10. G. Graff, J. Jezierski and P. Nowak-Przygodzki, *Fixed point indices of iterated smooth maps in arbitrary dimension*, J. Differential Equations **251** (2011) 1526–1548.
11. G. Graff and P. Nowak-Przygodzki, *Fixed point indices of iterations of C^1 maps in $\mathbb{R}^3$*, Discrete Cont. Dyn. Systems **16** (2006), no. 4, 843–856.
12. P. Heath, *A survey of Nielsen periodic point theory (fixed n)*. Nielsen theory and Reidemeister torsion (Warsaw, 1996), 159–188, Banach Center Publ., 49, Polish Acad. Sci., Warsaw, 1999.
13. B. J. Jiang, *Lectures on the Nielsen Fixed Point Theory*, Contemp. Math. **14**, Amer. Math. Soc., Providence 1983.
14. J. Jezierski and W. Marzantowicz, *Homotopy methods in topological fixed and periodic points theory*, Topological Fixed Point Theory and Its Applications, 3. Springer, Dordrecht, 2006.
15. J. Llibre, J. Paranõs, J. A. Rodriguez, *Periods for transversal maps on compact manifolds with a given homology*, Houston J. Math. **24** (1998), No. 3, 397–407.

Grzegorz Graff, Faculty of Applied Physics and Mathematics, Gdansk University of Technology, Narutowicza 11/12, 80-233 Gdansk, Poland
Email address: *familyname*@mif.pg.gda.pl

Paweł Pilarczyk, Centre of Mathematics, University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
Current address: Institute of Science and Technology (IST) Austria, Am Campus 1, 3400 Klosterneuburg, Austria
Email address: pawel.*familyname*@ist.ac.at