Computer Science • 17 (2) 2016 http://dx.doi.org/10.7494/csci.2016.17.2.225

Piotr Faliszewski
Maciej Smołka
Robert Schaefer
Maciej Paszyński

ON THE COMPUTATIONAL COST AND COMPLEXITY OF STOCHASTIC INVERSE SOLVERS

Abstract *The goal of this paper is to provide a starting point for investigations into a mainly underdeveloped area of research regarding the computational cost analysis of complex stochastic strategies for solving parametric inverse problems. This area has two main components: solving global optimization problems and solving forward problems (to evaluate the misfit function that we try to minimize). For the first component, we pay particular attention to genetic algorithms with heuristics and to multi-deme algorithms that can be modeled as ergodic Markov chains. We recall a simple method for evaluating the first hitting time for the single-deme algorithm and we extend it to the case of HGS, a multi-deme hierarchic strategy. We focus on the case in which at least the demes in the leaves are well tuned. Finally, we also express the problems of finding local and global optima in terms of a classic complexity theory. We formulate the natural result that finding a local optimum of a function is an NP-complete task, and we argue that finding a global optimum is a much harder, DP-complete, task. Furthermore, we argue that finding all global optima is, possibly, even harder (#P-hard) task. Regarding the second component of solving parametric inverse problems (i.e., regarding the forward problem solvers), we discuss the computational cost of hp-adaptive Finite Element solvers and their rates of convergence with respect to the increasing number of degrees of freedom. The presented results provide a useful taxonomy of problems and methods of studying the computational cost and complexity of various strategies for solving inverse parametric problems. Yet, we stress that our goal was not to deliver detailed evaluations for particular algorithms applied to particular inverse problems, but rather to try to identify possible ways of obtaining such results.*

Keywords hierarchic genetic strategy, inverse problem, hybrid method

Citation Computer Science 17 (2) 2016: 225–264

1. Introduction

Studying computational cost and complexity is one of the most important tasks in the analysis of algorithms. Generally, the goal of such considerations is to establish a relationship, possibly as a function, between the resources necessary to solve a particular problem (e.g., computational time or memory usage), and the amount of data to be processed or the quality of the results (e.g., their numerical accuracy). Studying computational cost is particularly important in the context of algorithms that are used to simulate physical, environmental, and technological phenomena as well as in the context of identification, optimal design, and control problems. This is so, because such problems are pertinent to a large number of real-life engineering tasks; understanding their computational complexity translates to direct practical benefits. Indeed, studies of the computational costs of algorithms provide a means of measuring their quality and form a basis for speeding up existing computational systems. The results of complexity analysis may lead to new algorithmic ideas and new architectural inventions that allow one to face new, more-involved problems.

Throughout this paper, we focus on the computational cost of solving inverse problems. Inverse problems form an important area of contemporary research related to fundamental problems in science and engineering. Typically, an inverse problem consists of finding some unknown value (e.g., a property of a medium or of an object) based on some given observations (e.g., based on the response of this medium or this object to a probing signal). A general framework of such problems provides an analytical means of building mathematical models, establishing physical constants for these models, and giving insights into the design of experiments [4, 27, 74].

Typically, a mathematical formulation of an inverse problem is composed of the following: (a) a *forward problem model*, which mimics the response of a real-word system under the assumptions that all relevant parameters are known; (b) a *misfit function*, which measures the discrepancy between the simulated system and the measurements taken from the real-world system; and (c) a *global optimization problem*, often equated with the inverse problem itself, where the goal is to find parameter values for the forward problem model – from a prescribed admissible parameter domain – that minimize the misfit function.

Inverse problems are not frequently well-posed in the sense of Hadamard; i.e., their solutions might not be unique and/or might be unstable under data perturbations. Thus, they pose severe numerical difficulties.

There are a number of reasons for the ill-posedness of a given inverse problem. First, it might be an inherent feature of its mathematical formulation. This source of its ill-posedness may be possible to anticipate based on the physical evidence regarding the system at hand (see; e.g., the work of Cabib et al. [12]). Second, ill-posedness may appear as a consequence of uncertainties in the misfit function. For example, the misfit function might be represented inaccurately due to insufficient knowledge of the problem due to errors in data measurements or errors regarding its representation (see; e.g., the works of Koper et al. [31], Meruane and Heylen [36], and Caicedo and

Yun [13]). Finally, the ill-posedness of the problem may result from approximation and arithmetic errors introduced throughout the computation. Some global optimization strategies (both deterministic and stochastic) may produce artifacts in the form of local objective extrema (see; e.g., the work of Barabasz et al. [7]). Moreover, an unavoidable error of misfit evaluation makes it difficult to distinguish the global minimizers among many local ones.

There are two main components involved in solving a given inverse problem. First, one needs an algorithm for solving the forward problem. Second, one needs an appropriate global optimization strategy. Regarding the first issue, we consider inverse problems where the forward problems are modeled through partial differential equations (PDEs). There are plenty of methods for solving boundary value problems for PDEs, and broad literature is available. We focus on methods for solving variational PDEs due to their efficiency and flexibility [14, 16, 17, 20].

The choice of the second component (the global optimization strategy) is far more involved, because many standard optimization methods are not applicable due to the ill-posedness of the problems. For example, convex optimization methods such as gradient-based ones may be inapplicable because of the misfit function irregularity (e.g., they may fail to find some of the solutions due to the misfit function multimodality or insensitivity in some areas of the admissible domain). Such problems only become larger with the increasing multimodality of the misfit function or appearance of large plateaus, in which case popular regularization methods tend to return artifacts rather than real solutions [22].

Thus, it is often necessary to seek other optimization strategies. One possibility is to use stochastic metaheuristics, which are *solution methods that orchestrate an interaction between local improvement procedures and higher-level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space* [25, 37, 39]. Among metaheuristic search strategies, we highlight evolutionary algorithms (EAs; we point the reader to the works of Burczyński and Beluch [10], Burczyński et al. [11], and Meruane and Heylen [36] for some examples of using EAs to solve inverse problems). Our particular interest is devoted to EAs that can handle misfit multimodality, such as niching and sequential niching strategies (see, e.g., the work of Mahfoud [35]), Hierarchic Genetic Strategy HGS [5, 7, 24, 29, 30, 62], and adaptive, stochastic multi-start method (see, e.g., the work of Telega et al. [76]).

Perhaps the main disadvantage of stochastic global optimization strategies is their enormous computational cost, caused by a huge number of the objective evaluations. Each objective evaluation requires one to solve the given forward problem (which typically is computationally intensive) and, based on the result, to compute the misfit function. Thus, to decrease the computational cost of a stochastic strategy, one should attempt both to reduce the number of objective evaluations and decrease the computational cost of the forward problem solver.

The study of the computational cost and complexity of strategies for solving inverse problems is quite complicated. The goals of this paper are to develop a preliminary taxonomy of necessary definitions and to present preliminary results regarding

the computational cost of crucial parts of several strategies for solving inverse problems. do not aim to deliver detailed evaluations for particular strategies; but, we rather wish to find ways of finding such types of results.

The paper is organized as follows: in Section 2, we provide necessary background information, including broad introductory information regarding several strategies for solving inverse problems. In Section 3, we present basic features and tools for dealing with our problems, and in Section 4 we present some preliminary results. We conclude in Section 5.

2. Preliminaries

In this section, we provide necessary background information regarding inverse problems and various strategies for solving them.

2.1. Forward and inverse problems

We use the following (quite general) definition of an inverse problem. The goal is to find the value of parameter $\omega^* \in \mathcal{D}$ that is a solution of the following global optimization problem:

$$\arg\min_{\omega\in\mathcal{D}} \{f(u_o, u(\omega)) : \ A_\omega(u(\omega)) = 0\}, \tag{1}$$

where A_ω is the family of *forward problem operators*, $u(\omega) \in V$ is the direct solution corresponding to ω, $u_o \in \mathcal{O}$ is an observation (typically a measured quantity related somehow to the direct solution), and $f : \mathcal{O} \times V \longrightarrow \mathbb{R}_+$ is *the misfit function*. In a typical situation, V is a Sobolev space and $A : V \longrightarrow V'$ is a differential operator between V and its conjugate V'. In this work, we restrict our attention to those inverse problems where:

1. $V \subset (H^r(\Omega))^d$; $r, d \in \mathbb{N}$, is a set of real- or vector-valued, generalized functions defined on the bounded domain $\Omega \subset \mathbb{R}^N, N \in \mathbb{N}$, with the Lipschitz boundary $\partial\Omega$ (see; e.g., the books of Denkowski et al. [18, 19]), so $V \ni v : \Omega \to \mathbb{R}^d$(or $\mathbb{C}^d$), where d is the finite dimension of the candidate solution. Moreover, the definition of V may include some other assumptions imposed by the boundary conditions on the solution $u(\omega)$.
2. The admissible set $\mathcal{D}$ is a subset of $L^\infty(\Omega \to \mathbb{K})$, where $\mathbb{K}$ is a proper subset of the set of parameter values; *e.*g., a set of tensor fields satisfying appropriate physical restrictions. Typically, $\mathcal{D}$ is a set of uniformly bounded step functions corresponding to a predefined, regular decomposition of Ω. In the sequel, we shall restrict the admissible set $\mathcal{D}$ to a discrete or even finite representation.
3. The operators A_ω have an 'elliptic-like' form, i.e., $\langle A_\omega(u(\omega)), v\rangle = a(\omega; u(\omega), v) - l(v)$ for all $v \in V$, where l belongs to V' and a is a function, $a : \mathcal{D} \times V \times V \to \mathbb{R}$, that satisfies $a(\omega, u, \cdot) \in V'$ for all $\omega \in \mathcal{D}, u \in V$.

Summing up, the *forward problem* under consideration has the following form: Given $\omega \in \mathcal{D}$, find $u \in V$ such that:

$$\langle A_\omega(u), v\rangle = 0, \quad \forall v \in V,$$

or, in other words:

$$a(\omega; u, v) = l(v), \quad \forall v \in V. \tag{2}$$

We consider only well-posed forward problems. That is, we require that, for each $\omega \in \mathcal{D}$, there exists exactly one $u \in V$ satisfying (2), and the solution depends continuously on the right-hand side l. Moreover, we assume that $u(\omega)$ can be sufficiently well-approximated by Galerkin solutions of (2). Typical assumptions ensuring these features are the uniform Lipschitz continuity and coercivity (a condition stronger than ellipticity) of the operators A_ω (see; e.g., the book of Ciarlet [14]). Thus, with a minor abuse of notation, we use $u(\omega)$ to denote the solutions for our forward problems.

We say that problem (1) is inverse to the forward problem (2). We refer to inverse problems of this form as *inverse parametric problems* (or inverse coefficient problems) for partial differential equations.

Typically, the misfit function has the form of the 'energy' discrepancy $|\mathcal{E}(u(\omega)) - \mathcal{E}_0|$, where $\mathcal{E}(u(\omega)) = \frac{1}{2}a(\omega; u(\omega), u(\omega)) - l(u(\omega))$ and $\mathcal{E}_0$ is the measured energy associated with the observation u_0. The misfit function might also have the form of the boundary value discrepancy $\|u(\omega) - u_0\|_\Gamma$, where $\|\cdot\|_\Gamma$ is the norm (or semi-norm) evaluating the trace of the function from V (see; e.g., the books of Denkowski et al. [18,19]), or the form of the discrepancy $|L(u(\omega)) - L_0|$ in the so-called 'quantity of interest', where $L \in V'$ and L_0 is a measured value of L associated with observation u_0. Finally, the misfit function might be a composition of all of the expressions mentioned above.

2.2. Adaptive forward solvers

Let us now discuss a group of numerical methods for finding approximate solutions to the forward problem (2) by using the self-adaptive hp–Finite Element Method (hp–FEM) (further references are available in the literature [14, 16, 17, 20, 53]). These methods consists of constructing a subspace $V_{h,p} \subset V$ with finite basis $\{e^i_{h,p}\}_{i=1,\ldots,N_{h,p}}$, $N_{h,p} < +\infty$. The subspace $V_{h,p}$ is constructed by partitioning the solutions domain Ω into a finite number of non-intersecting polyhedrons – called elements – and by defining basis functions as polynomials satisfying prescribed conditions over the vertices, edges, faces, and interiors of the elements. Each non-zero restriction of the basis function $e^i_{h,p}$ to a given element is called a *shape function*.

Assuming the value of parameter $\omega \in \mathcal{D}$ is given, the approximate solution $u_{h,p}$ to (2) is obtained as a linear combination of the basis functions:

$$u_{h,p} = \sum_{i=1}^{N_{h,p}} u^i_{h,p}\, e^i_{h,p}. \tag{3}$$

Using a similar representation for $v \in V_{h,p} \subset V$, we obtain a system of linear equations for (2):

$$\sum_{i=1}^{N_{h,p}} u_{h,p}^i \, a\left(g; e_{h,p}^i, e_{h,p}^j\right) = l\left(e_{h,p}^j\right) \quad j = 1, ..., N_{h,p}. \tag{4}$$

The coefficients $u_{h,p}^i$ are called *degrees of freedom*. The detailed mathematical description of the FEM method (as applied to the elliptic variational problems) may be found in Chapter 2 of the book of Ciarlet [14], while the convergence of this method for various FEM spaces is discussed in Chapters 3 and 4, or in the book of Descloux [20].

The accuracy of the approximation provided by the FEM method depends on the quality of the basis functions. The self-adaptive *hp*–FEM method is an algorithm for the automatic construction of the basis functions, delivering an exponential convergence of the accuracy with respect to the mesh size (i.e., with respect to the partition of the space into elements). The algorithm has been formulated in the works of Rachowicz et al. [53], Demkowicz [16], and Demkowicz et al. [17], and can be summarized in the following steps:

1. Generate an initial basis functions family $\{e_{h,p}^i\}_{i=1,...,N_{h,p}}$ spanned over the so-called *initial mesh*. The initial mesh becomes the so-called *coarse mesh* for the first iteration.
2. Solve the coarse mesh problem by computing the degrees of freedom $\{u_{h,p}^i\}$, $i = 1, ..., N_{h,p}$. After this step, we obtain an approximate solution $u_{h,p}$ for the coarse mesh in the form (3).
3. Generate the fine basis $\{e_{\frac{h}{2},p+1}^i\}_{i=1,...,N_{\frac{h}{2},p+1}}$ spanned over the so-called *fine mesh*. The fine mesh is obtained from the coarse mesh by breaking each coarse mesh element into several elements (their number depends mainly on the Ω dimension N and on the type of the element) and by increasing the polynomial order of approximation uniformly by one.
4. Solve the fine mesh problem by computing the degrees of freedom $\{u_{\frac{h}{2},p+1}^i\}$, $i = 1, ..., N_{\frac{h}{2},p+1}$. After this step, we obtain the fine mesh approximate solution

$$u_{\frac{h}{2},p+1} = \sum_{i=1}^{N_{\frac{h}{2},p+1}} u_{\frac{h}{2},p+1}^i \, e_{\frac{h}{2},p+1}^i. \tag{5}$$

5. Select an optimal refinement strategy for each finite element from the coarse mesh. This selection process should be based on the error estimations $\|e_{rel}\|_V^2$, where $e_{rel} = u_{\frac{h}{2},p+1} - u_{h,p}$ is computed using the coarse mesh and the fine mesh solutions (see the book of Demkowicz [16]). The optimal refinements contain a list of h refinements (requests to break some elements) and p refinements (increasing some polynomial orders of approximations by one).
6. Execute all required h refinements.
7. Execute all required p refinements.

8. If the maximum relative error of the solution is greater than the required accuracy, go to Step 2. The new optimal mesh becomes the coarse mesh for the next iteration. Otherwise, output the current solution.

The self-adaptive hp–FEM algorithm has also been redesigned by using the graph grammar approach [44–48, 55, 72] and has been efficiently implemented on parallel machines [49, 51, 52].

Since the applied approximation is an internal one (i.e., $V_{h,p} \subset V,\ \forall h, p$), each approximate solution is obtained unambiguously (this, again, justifies the use of functional notation $u_{h,p}(\omega)$).

2.3. Population-Based Stochastic Optimization

Population-based stochastic search strategies are well-known from their many incarnations: Monte Carlo Methods, Simulated Annealing, Genetic and Memetic Algorithms, Ant Colony Search algorithm, etc. (see; e.g., the books of Pardalos and Romeijn [43], Glover and Kochenberger [25], and Schaefer [57] for references). From a high-level point of view, in each time step t, these strategies transform a multiset P_t of candidate solutions to the global optimization problem, called the population (see Figure 1), and eventually – when some stop condition is met – return the best solution found. A single time step of these strategies is traditionally referred to as an *epoch*.

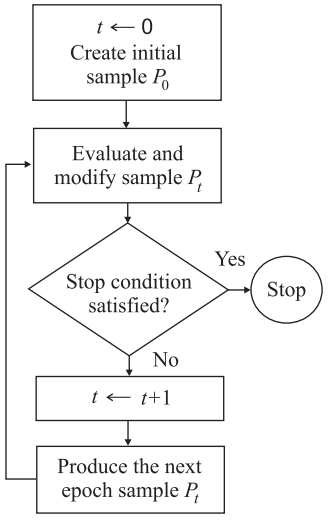

Figure 1. General scheme of the single-deme, population stochastic search.

Typically, the most computationally intensive part of these strategies is the evaluation of the current population P_t. It is especially costly in the case of solving inverse problems of the form (1), because computing the misfit value $f(u_o, u(\omega))$ for each single individual requires computing $u_{h,p}(\omega)$, by executing the hp–FEM algorithm.

Each population-based stochastic search algorithm is associated with a sequence of probabilistic measures $\{\rho^t\}_{t=1,2,3,\ldots} \subset \mathcal{M}(\mathcal{D})$ that govern the process of sampling the populations. That is, each population P_t is obtained by sampling a number of individuals from $\mathcal{D}$ according to measure ρ^t. The initial population P_0 is obtained by using ρ^0, which is typically the uniform distribution over the whole $\mathcal{D}$. Distributions ρ^t might be independent from each other, or ρ^t may depend on the earlier measures $\{\rho^\xi\}_{\xi<t}$. In the most-practical cases (genetic algorithms), the sequence of sampling measures depends on the size of the population, $\mu \in \mathbb{N}$. So, the sampling measures in the consecutive epochs are denoted by $\rho^t_\mu,\ t = 1, 2, 3, \ldots$.

The basic, single-deme stochastic search is convenient for theoretical investigations but is usually insufficient for solving difficult problems with many local minima and large plateau regions. The main directions for improving this strategy are: (1) to allow the population size to change throughout the computation; (2) to introduce a structure of demes within the population; or (3) to accept many demes to be only partially synchronized, allowing for variable search accuracy. Using these techniques typically leads to a loosely coupled computational scheme, locally governed and synchronized by software agents.

2.4. Adaptive inverse solvers

Let us now describe several adaptive strategies, well-suited for solving inverse problems.

2.4.1. Adaptive accuracy population-based searches and the Island Model

We start by mentioning some well-known, population-based strategies that involve the adaptivity paradigm. The type of adaptivity in the first two cases (the *Dynamic Parameter Encoding* [DPE] approach and the *Delta Coding* approach) is quite different than in the last one (the *Island Model* [IM]). The pioneering papers concerning DPE, Delta Coding, and IM are those of Schraudolph and Belew [65], Whitley, Mathias and Fitzhorn [81], and Whitley et al. [80, 82], respectively. For a synthetic description of these methods, we point the reader to Sections 5.3.8 and 5.4.2 of Schaefer's book [57].

Both DPE and Delta Coding algorithms use affine, binary encoding of the phenotypes located in the regular, compact set $\mathbf{D} \subset \mathbb{R}^N, N \geqslant 1$. Typically, it is assumed that, at the start of each algorithm, the phenotypes form a regular, coarse grid $\mathcal{D} \subset \mathbf{D}$. Both algorithms perform conventional evolutionary computations, as described in the previous Section 2.3. However, they monitor the evolution progress between the consecutive epochs, and when they encounter stagnation in the search process (e.g., when the mean fitness does not decrease sufficiently), they take appropriate remedial actions; different ones for DPE and different ones for Delta Coding.

DPE chooses an arbitrary number of best-fitted individuals that will constitute the seed of the forthcoming population. The accuracy of the search is improved by increasing the length of the binary code of individuals. After the accuracy improvement, the evolutionary search is continued until the next stagnation is observed or the global stopping criterion occurs.

On the other hand, the Delta Coding algorithm passes to the *Delta Phase*, in which the subdomain $\mathbf{D}_\Delta \subset \mathbf{D}$ containing the best-fitted individuals is identified. The new population search is started in the narrowed domain $\mathbf{D}_\Delta$, which is encoded using a finer spatial grid of phenotypes. The initial population is sampled uniformly. The best-fitted individuals are stored in memory at the start of each Delta Phase. Delta Phases are repeated until the global stopping criterion is met.

Both strategies follow a similar process to the standard branch-and-bound deterministic scheme (see; e.g., the book of Scholz [64]). They may speed up finding the global minimizer with a desired accuracy (if it is unique), but they provide no stochastic guarantees of doing so.

While the DPE and Delta Coding strategies improve the exploitation power of stochastic, population-based search algorithms, the adaptation mechanism implemented in the island model improves their exploration ability. The main idea behind IM is to perform several population-based stochastic searches (called "islands") to solve the same optimization problem in parallel. The sampling mechanisms might differ among the islands, but the individuals are encoded in the same way for each of them. The adaptation mechanism consists of temporarily exchanging the genetic material (groups of individuals) between the islands. Unfortunately, in spite of the popularity of IM systems in the initial decades of artificial evolution research, they brought relatively little to solving difficult, global optimization problems such as the inverse ones.

A significant contribution to the study of the island model was delivered by Skolicki and De Jong [68,69]. Schaefer, Byrski and Smołka [60] introduced a rigorous mathematical description of the island model dynamics, proving a theorem regarding its asymptotic guarantee of finding all local minimizers.

2.4.2. Common adaptation of accuracy and deme structure – HGS

Hierarchic Genetic Strategy (HGS) links both trends in stochastic search adaptation strategies, represented by the algorithms described in the previous section. Moreover, its low computational cost makes it suitable for solving difficult inverse problems, such as those formulated as (1).

To allow further study of the HGS computational cost and stopping condition, let us briefly describe the basic ideas and structures standing behind it. Our description is based on that from two of our recent papers [24,70] and from several papers referenced therein.

The HGS algorithm improves upon the exploratory and exploitative abilities of the standard evolutionary strategy by dynamically building a tree of demes. As we head from the tree's root to its leaves, each node is responsible for a more-accurate and more-focused search process. Since these search processes are independent, the algorithm can be implemented on parallel architectures in a very effective way.

The HGS algorithm operates as follows:

1. At the start, the *root deme* of the order one is created. It searches broadly with the lowest accuracy, which allows for the wide exploration of the admissible domain.

2. In a single HGS step, all live demes perform *metaepochs* composed of K genetic epochs (at most). An active deme may be stopped during a metaepoch if the observed progress in evolution is insufficient.
3. After each metaepoch, all live demes of the order less then m (which is the maximum depth of the tree) attempt to *sprout* child-demes. Each new child-deme is concentrated near the best-fitted individual currently found by its parental deme. Sprouting is performed conditionally; i.e., it is suspended if the sprouting region has already been explored by sibling-demes.
4. The demes of the same order arriving to the same region are reduced at the end of each metaepoch, in order to prevent search redundancy.

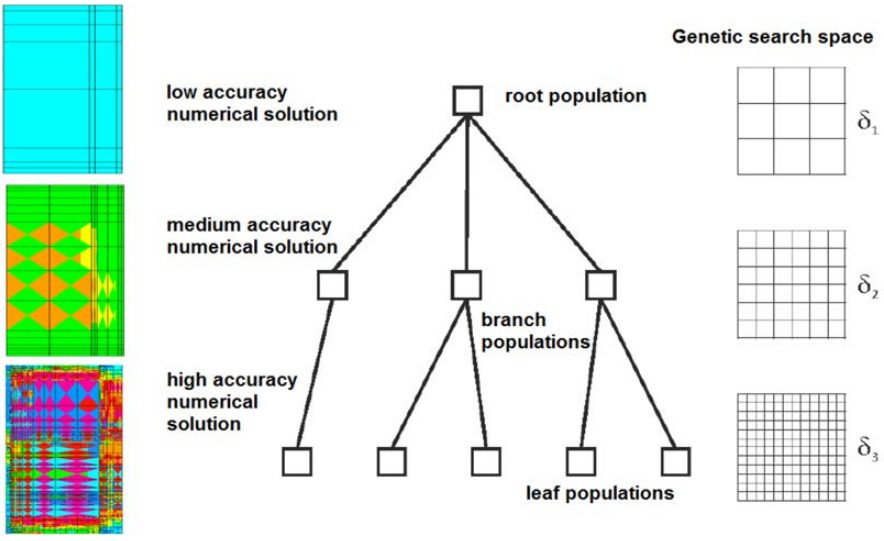

Figure 2. HGS tree and corresponding coding meshes (central and right parts of the picture). hp–FEM computational grids associated with different accuracy of direct problem solution (left part of the picture).

The HGS algorithm can use two types of hierarchies of encoding mappings associated with the deme hierarchy.

The first is the hierarchy involving the idea of binary affine encoding. At the start, we define the regular, densest grid of phenotypes $\mathcal{D} \subset \mathbf{D} \subset \mathbb{R}^N$, so that each phenotype $(x_1, \ldots, x_n)$ has coordinates belonging to the finite sets $x_i \in \{\xi_1^i, \ldots, \xi_{p_i^m}^i\}$ and $p_i^m = 2^{k_i^m}, i = 1, \ldots, N$. This grid is used for encoding leaf individuals, with the largest binary code of length $\sum_i^N p_i^m$. Each higher level deme of order $1 \leqslant w < m$ is associated with the grid of phenotypes obtained by dropping some points from the grid used by demes of the $(w+1)$-th order; but still, the number of available values of i-th coordinate is the power of two, $p_i^w = 2^{k_i^w}, k_i^w < k_i^{w+1} \leqslant k_i^m, i = 1, \ldots, N$. The length of the binary code at this level equals $\sum_i^N p_i^w$. If we denote by δ_w the maximum

diameter of the phenotype grid at the w-th level of the HGS tree, then we obtain $\delta_1 > \delta_2 >, \ldots, > \delta_m$. The accuracy improvement of the genetic search associated with the sprouting of a child-deme of order $w \leqslant m$ is obtained by extending the binary code representing each i-th coordinate by the $(p_i^w - p_i^{w-1})$-length suffix. The detailed description of the affine binary encoding and the hierarchy of binary encodings applied in HGS can be found in Sections 3.1.1 and 5.4.3 of Schaefer's book [57].

The second approach is a real-number encoding hierarchy. For simplicity, let us assume that the admissible domain is of the form $\mathcal{D} = \mathbf{D} = \prod_{i=1}^{N}[a_i, b_i] \subset \mathbb{R}^N$, where $a_i < b_i$. $\mathcal{D}$ is used as the genetic space (space of codes) for all leaf demes, allowing the most-detailed and most-accurate search that is available with the current computer arithmetic. The genetic spaces for higher-level demes, of order $1 \leqslant w < m$, are obtained by scaling, so $\mathcal{D}_w = \prod_{i=1}^{N}[0, \frac{b_i - a_i}{\eta_i}] \subset \mathbb{R}^N$, where $+\infty > \eta_1 \geqslant \eta_2 \geqslant \ldots \geqslant \eta_m = 1$ are the proper scaling coefficients. The maximum search accuracy δ_m achieved in leaves is then reduced to $\delta_w = \eta_w \, \delta_m$, for $j = 1, \ldots, m-1$, in demes of lower order w. The hierarchy of real-number codes was introduced by Wierzba [83]. Its broad description is available in Section 5.4.3 of Schaefer's book [57].

HGS was introduced by Kołodziej and Schaefer [62]. Their work also contains its partial, formal analysis. They proved that binary HGS has an asymptotic guarantee of success, and its computational cost can be estimated in comparison to SGA. The real-number HGS and its efficiency were discussed by Wierzba et al. [83] and by Schaefer and Barabasz [29]. The complete Markov model of HGS dynamics (excluding branch reduction mechanism) was described by Schaefer et al. [59].

2.4.3. Coupling with the Adaptive Direct Solvers: *hp*–HGS

In order to solve the particular class of inverse problems considered in this paper, HGS may be combined with the adaptive *hp*–FEM algorithm for solving the forward problems $A(u(\omega)) = 0$, to evaluate the misfit function $f(u_o, u(\omega))$. This strategy, called *hp*–HGS, consists of estimating the fitness value of each individual with the required accuracy depending on the level of the HGS tree. Forward problems at the root level are solved with the lowest accuracy and at the leaf level with the highest accuracy.

For some important cases (heat conduction, elasticity, AC and DC current in material continua), the following inequality holds:

$$Errf_i \leqslant ((e_{rel})_i)^{\alpha} + C \cdot \delta_i + Rest, \; i = 1, \ldots \; m, \; \alpha \geqslant 1, \tag{6}$$

where $Errf_i$ is the misfit evaluation error, $(e_{rel})_i$ is the relative FEM error (see Section 2.2), and δ_i stands for the inverse solution error at the i-th level of the HGS tree. The constants C and α depend on the particular inverse problems to be solved. *Rest* rapidly decreases when *hp*–FEM converges. So, in order to obtain the economic profile of computation, we can set:

$$(e_{rel})_i ::= Ratio_i \cdot (\delta_i)^{\frac{1}{\alpha}}, \; i = 1, \ldots \; m, \tag{7}$$

where $Ratio_i$ depends on the particular problem. Such a strategy allows for a significant decrease in computational cost for $i < m$ due to the rapid decrease of the hp–FEM computational cost when the restriction for the relative error $(e_{rel})_i$ is relaxed (see (8) in Section 2.2). The left part of Figure 2 shows sample hp–FEM computational grids associated with various accuracies of solving the forward problem at consecutive levels of the hp–HGS tree.

The hp–HGS strategy was introduced by Schaefer, Paszyński, and Barabasz [50]. Crucial mathematical results verifying the evaluation (6), the setting of $Ratio_i$, $i = 1 \ldots, m$ (see formula (7)), together with the computational examples for heat conduction problems, Young modulus restoring, and the inversion of logging measurements for DC and AC cases, was reported by Barabasz et al. [7], Barabasz et al. [6], Gajda-Zagórska et al. [24], and Smołka et al. [70], respectively. These papers also contain a detailed algorithmic description of hp–HGS.

The asymptotic guarantee of success, as well as some statistics concerning computational cost comparison of SGA, HGS, and hp–HGS for solving inverse problems, are provided by Barabasz and Schaefer [58].

3. Basic features and tools

In this section, we present the basic features and tools that may be useful when assessing the computational complexity of solving inverse problems. We analyze both results regarding the complexity of self-adaptive hp–FEM, running-time considerations for global optimization heuristics, and classic complexity theory results for these problems.

3.1. Computational cost versus relative error for hp–FEM

Let us consider a self-adaptive hp–FEM algorithm (see Section 2.2) applied to solving an "elliptic-like" forward problem of the form (2). It has been proven that the appropriate selection of element sizes h and polynomial orders of shape functions p leads to an exponential convergence of the numerical error with respect to the number of basis functions (degrees of freedom) [2,3,66]. In this process of convergence, the parameters h and p tend to 0 and $+\infty$, respectively, in an interdependent way. They both need to satisfy the syntactic rules of element mesh refinement, taking approximations appropriate to the mesh topology. In the computational practice, p rarely exceeds 10, and h is bounded from below by the arithmetic error of the particular implementation.

Exponential convergence of the self-adaptive hp–FEM is experimentally confirmed as the straight line $y = -ax + b$ in the system of coordinates where the horizontal axis represents the cube root of the number of degrees of freedom, $x = N^{1/3}$, and the vertical axis represents the logarithm of the relative error $y = log_{10}(\|e_{rel}\|)$, $\|e_{rel}\| < 1$, where $e_{rel} = \frac{u_{\frac{h}{2},p+1} - u_{h,p}}{u_{\frac{h}{1},p+1}}$ is the relative error, expressed as the difference between two consecutive approximate solutions, and $\|\cdot\|$ is a proper norm in the space of the problem's solutions. Constants a and b are

positive and problem dependent. This implies that $log_{10}(\|e_{rel}\|) = -a(N^{1/3}) + b$, which in turn entails $N = -c_1(log_{10}(c_2\,\|e_{rel}\|))^3$ for $\|e_{rel}\| < 1$, where the constants $c_1 = a^{-3}, c_2 = 10^{-b} > 0$ are problem-specific. The computational cost of solving the forward problem over a two-dimensional mesh depends on the structure of the *hp*-refined mesh. For a regular mesh, the cost is of the order $O\left(N^{3/2}\right)$. For meshes with point-wise singularities, the cost can be reduced down to be linear, $O\left(N\right)$. Finally,

$$cost = O\left(-c_1(log_{10}(c_2\,\|e_{rel}\|))^{3r}\right),\ \|e_{rel}\| < 1, \tag{8}$$

where $r \in [1, 3/2]$, and now $c_1 = a^{-3r}, c_2 = 10^{-b} > 0$.

The exponential convergence rate is illustrated in Figure 3 for a model L-shape domain problem. In this figure, we also compare the *hp*-adaptive algorithm with alternative algorithms and show that only *hp*-adaptivity provides an exponential convergence rate. In particular, we compare the algorithm with:

- *h*-uniform algorithm, which breaks all of the elements of the mesh into smaller elements but does not increase the polynomial order of approximation,
- *p*-uniform algorithm with 3 elements, which increases the polynomial order of approximation over the initial mesh with 3 elements,
- *p*-uniform algorithm with 12 elements, which increases the polynomial order of approximation over the initial mesh with 12 elements,
- *h*-adaptive algorithm, which is the restriction of the automatic *hp*-adaptive algorithm to the case where the polynomial orders of approximation remain constant.

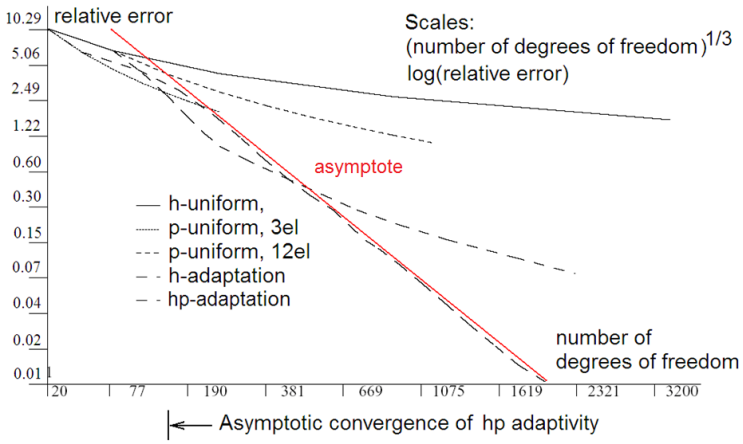

Figure 3. Comparison of different mesh-refinement strategies: *h*-uniform strategy, *p*-uniform strategy starting from 3 or 12 elements, automatic *h*-adaptation and automatic *hp*-adaptation strategy.

3.2. First Hitting Time (FHT) in stochastic search

Analyzing the so-called *First Hitting Time* (FHT) of a stochastic search algorithm is among the most-basic ways of evaluating its computational cost. The idea is as follows: first, we define some set $\mathcal{S} \subset \mathcal{D}$ of points in the search domain that are, in some sense, "close enough" to the global solutions. Second, we define a random variable – the first hitting time – that gives the number t of the first genetic epoch where the current population, P_t, contains at least one member of $\mathcal{S}$. Finally, we calculate the appropriate statistics of this random variable, such as the expected value. These statistics give us some level of understanding of the algorithm's efficiency. For example, if we assume that the size of the population in each epoch is constant and equal to μ, quantity $\mu \cdot E(\mathrm{FHT})$ is the expected computational cost. If sampling measures are constant ($\rho^t = \rho^0 \ \forall t > 0$) and $\rho^0(\mathcal{S}) > 0$ is known, then $E(\mathrm{FHT})$ can easily be computed using the binomial distribution (see; e.g., the work of Pardalos and Romeijn [43]). Analyzing the computational complexity of single- and multi-deme stochastic search algorithms through the first hitting time is an intensively studied area of research [8, 21, 33, 34, 54, 73]. In particular, the just-cited papers (and many similar ones) show rigorous results regarding the first hitting time of various algorithms. Unfortunately, these papers typically make quite restrictive assumptions, including those regarding the population size (single-individual populations or IMs composed of single-individual demes), regarding the objective function (e.g., assuming a globally convex objective), and others.

While considering the first hitting time provides a simple and precise notion of convergence of a stochastic search algorithm, it is not immediate how to use this notion to define a stopping condition for such algorithms. First, it is not easy to evaluate the measure $\rho^t(\mathcal{S})$, for each given $t \geqslant 0$, and usually only very rough lower bounds are available. Second, if the sequence of sampling measures is not constant, then computing the (expected) first hitting time requires precise knowledge regarding the stochastic dynamics of these measures (which is rarely available). In effect, estimating the expected first hitting time delivers only a very rough upper bound on the number of computational steps necessary in a stochastic search.

Summing up, for the case of complex metaheuristics applied to difficult inverse problems, using the first hitting time statistics to implement stopping conditions appears to be difficult. Algorithms using such stopping conditions would likely deliver unacceptable computational costs.

3.3. Convergence of genetic search along a heuristic

The concept of population-based stochastic search *heuristics* was introduced by Michael Vose [79] and applied to the Simple Genetic Algorithm (SGA). Some attempt at slightly generalizing the heuristic for other population-based searches can be found, for example, in the works of Schaefer [56, 57] and Schaefer and Jabłoński [61]. The existence of heuristics is conditioned through some simple principles:

1. The admissible set is finite, $\mathrm{card}(\mathcal{D}) = r < +\infty$.

2. Each population of an arbitrary finite size $\mu < +\infty$ (being the multisets of $\mathcal{D}$ elements) in unambiguously represented by an element of a compact, bounded set $\mathcal{E} \subset \mathbb{R}^r$.
3. There exists a bijection $\theta : \mathcal{E} \to \mathcal{M}(\mathcal{D})$; i.e., each population representative might be treated as the stochastic sampling measure over the admissible set.
4. The operations that lead to sampling (e.g., mutation, crossover) have to have the same stochastic characteristic at each epoch t of the search (e.g., by sampling each population $P_t, t = 1, 2, \ldots$).

Assumption 2 is satisfied; e.g., if $\mathcal{E}$ is the unit $r-1$ dimensional simplex in $\mathbb{R}^r$ containing frequency vectors of all populations. If the particular, finite population size μ has been established, then representations of such populations form a subset $X_\mu \subset \mathcal{E}$ such that:

$$n = \#X_\mu = \left(\begin{array}{c} r+\mu-1 \\ \mu-1 \end{array} \right) < +\infty \tag{9}$$

(see the work of Vose for details [79]).

Assumption 4 provides the possibility of modeling the search dynamics as a stationary Markov chain, with state space X_μ contained in $\mathcal{E}$ and the probability transition rule $\tau : \mathcal{E} \mapsto \mathcal{M}(\mathcal{E})$. The symbol τ is polymorphic and can be understood as the coherent family of transition rules for various population sizes $\mu < +\infty$, so that $\tau|X_\mu : X_\mu \to \mathcal{M}(X_\mu)$.

The realization of such a stochastic process is the sequence of μ-sized populations $P_0, P_1, P_2, \ldots$ or their unambiguous representations $x_\mu^0, x_\mu^1, x_\mu^2, \cdots \in X_\mu \subset \mathcal{E}$. We denote the starting probability distribution by $\pi_\mu^0 \in \mathcal{M}(X_\mu)$, and we denote the probability distributions in the consecutive epochs by $\pi_\mu^t \in \mathcal{M}(X_\mu), t = 1, 2, 3, \ldots$.

For each particular $\mu < +\infty$, function $\tau|X_\mu$ is fully characterized by the $n \times n$ transition probability matrix Q such that $Q_{xy} = \tau(x)(\{y\}),\ \forall x, y \in X_\mu$. Finally, from the Kolmogorov equation we have $\pi_\mu^t = Q^t\, \pi_\mu^0$, $t = 1, 2, \ldots$.

Definition 1. *Let us consider a particular class of population-based stochastic search algorithms satisfying Assumptions 1–4 with the common space of states $\mathcal{E}$ and Markov transition rule τ. The continuous mapping $\mathcal{H} : \mathcal{E} \to \mathcal{E}$ is called their heuristic if it satisfies:*

1. $\forall x \in \mathcal{E}\ \ \mathcal{H}(x) = E(\tau(x))$,
2. $\forall x \in \mathcal{E}\ \ \theta(\mathcal{H}(x)) \in \mathcal{M}(\mathcal{D})$ *is the sampling measure for the epoch following the epoch in which the population represented by $x \in \mathcal{E}$ appeared.*

Let us now denote by x_μ^t the unique representation of population P_t of cardinality μ in the t-th evolution epoch.

Remark 2. *From condition 1 of the above definition, it follows that, for an arbitrary $\mu \in \mathbb{N}$, expectation $E(\tau(x_\mu^t)) \in \mathcal{E}$ represents the expected population in the next epoch, $t+1$, provided that in the t-th epoch population represented by x_μ^t appeared. Moreover, Condition 2 from the definition says, that $\rho_\mu^{t+1} = \theta(\mathcal{H}(x_\mu^t))$. In other words, the following diagram (see Figure 4) commutes.*

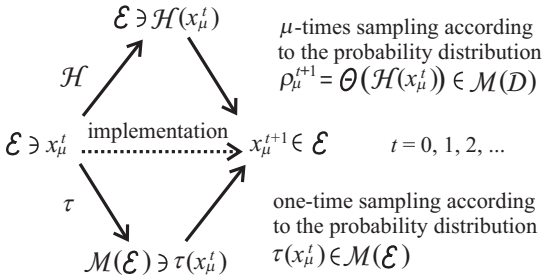

Figure 4. The stochastic schema of a population search with heuristics.

If the Simple Genetic Search with population size μ has heuristic $\mathcal{H}$, then the probability transition matrix Q can be computed using the following formula [79]:

$$Q_{xy} = \mu! \prod_{j=0}^{r-1} \frac{((\mathcal{H}(x))_j)^{\mu y_j}}{(\mu y_j)!} \quad \forall x, y \in X_\mu. \tag{10}$$

Definition 3. *Heuristic $\mathcal{H}$ is focusing if:*

$$\exists \mathcal{K} \subset \mathcal{E},\ \mathcal{K} \neq \emptyset;\ \forall x \in \mathcal{E}\ \exists z \in \mathcal{K};\ \mathcal{H}^p(x) \to z,\ p \to +\infty,$$

where $\mathcal{K}$ is the non-empty set of fixed points of $\mathcal{H}$ which attract all points from $\mathcal{E}$.

Remark 4. *If the space of states $\mathcal{E}$ is bounded and convex in some metric-topological space, then from Schauder's theorem [63], it follows that $\mathcal{K}$ is nonempty ($\mathcal{H}$ has fixed points).*

The following result was shown for the Simple Genetic Algorithm for which the space of states $\mathcal{E}$ is equal to the unit simplex (which is the closure of the set of frequency vectors of all populations of a finite cardinality $\overline{\bigcup_{\mu \in \mathbb{N}} X_\mu}$; see the work of Vose [79]).

Theorem 5 (Vose [79], Theorem 13.2)**.** $\forall K > 0, \forall \varepsilon > 0, \forall \nu < 1 \quad \exists N > 0$ *independent of $x_\mu^0 \in \mathcal{E}$ so that $\forall\ 0 \leqslant t \leqslant K$:*

$$\Pr\left\{\mu > N \Rightarrow \left\|x_\mu^t - \mathcal{H}^t(x_\mu^0)\right\| < \varepsilon\right\} > \nu \tag{11}$$

where $x_\mu^0 \in \mathcal{E}$ stands for the frequency vector of the initial population P_0.

The above theorem shows that, if the population is sufficiently large, then the deterministic trajectory of heuristic iterations $\{\mathcal{H}^t(x_\mu^0)\},\ t = 1, 2, 3, \ldots$, is followed arbitrarily closely by the trajectory of the SGA population, in the sense of inequality (11). Moreover, we may conclude that the trajectory of heuristic iterations represents the maximum search ability performed by the infinite $\mu \to +\infty$ population among all SGA searches represented by the same heuristic $\mathcal{H}$. Further, if the heuristic $\mathcal{H}$ is focusing (see Definition 3), then we may infer that its fixed points represent the limit search possibility of the aforementioned class of search algorithms. That is, the populations contained in $\mathcal{K}$ represent the maximum information about the solution among all populations represented in $\mathcal{E}$.

3.3.1. Approximating fixed points of a heuristic

We now consider the behavior of the class of Simple Genetic Algorithms governed by the same heuristic $\mathcal{H}$. Each SGA from this class has the same genetic space and the same selection and genetic operators (i.e., the parameters and the operator types are strictly the same), and they may differ only in the population cardinality μ.

Theorem 6 (Theorem 4.54 in the book of Schaefer [57], see also the works of Telega [75] and Telega, Schaefer and Cabib [76])**.** *Let us assume that genetic operator $\mathcal{H} : \mathcal{E} \to \mathcal{E}$ is focusing and that its set of fixed points is finite ($\#\mathcal{K} < +\infty$). Let us define:*

$$\mathcal{K}_\varepsilon = \{x \in \mathcal{E};\ \exists y \in \mathcal{K};\ d(x,y) < \varepsilon\}$$

to be the open ε-envelope of $\mathcal{K}$ in the $(r-1)$-dimensional hyperplane that contains $\mathcal{E}$, where $d(\cdot,\cdot)$ stands for the Euclidean distance in $\mathbb{R}^{r-1}$. If we assume that mutation is strictly positive ($p_m > 0$), then:

$$\forall \varepsilon > 0,\ \forall \eta > 0,\ \exists N \in \mathbb{N},\ \exists W(N) \in \mathbb{N};$$

$$\forall \mu > N,\ \forall k > W(N)\ \ \pi_\mu^k(\mathcal{K}_\varepsilon) > 1 - \eta,$$

where $\pi_\mu^k \in \mathcal{M}(\mathcal{E})$ is the measure associated with an arbitrary instance of the Simple Genetic Algorithms spanned by $\mathcal{H}$, with population size μ.

In other words, if $\mathcal{H}$ is focusing, then sufficiently large Simple Genetic Algorithm populations will concentrate close to the set $\mathcal{K}$ with an arbitrarily large probability $1 - \eta$ after sufficiently many genetic epochs.

The selection of the initial probability distribution $\pi_\mu^0 \in \mathcal{M}(\mathcal{E})$ does not affect the above result because the Markov chain of states in $X_\mu \subset \mathcal{E}$ and the transition probability rule τ associated with the Simple Genetic Algorithm is ergodic (provided that $p_m > 0$).

3.3.2. Well-tuning and convergence of sampling measures

We assume again that the set of phenotypes is embedded in a regular, compact set $\mathbf{D} \subset \mathbb{R}^N$. For each probabilistic measure $\rho \in \mathcal{M}(\mathcal{D})$, it is possible to define a measure with density $\overline{\rho} \in L^p(\mathbf{D})$ such that $\overline{\rho}$ is constant on the Voronoi neighborhood $\vartheta_\xi \subset \mathbf{D}$ associated with each $\xi \in \mathcal{D}$, and its value equals $\frac{\rho(\xi)}{\mathrm{meas}(\vartheta_\xi)}$ there (see Chapter 3 in Schaefer's book [57]).

The main advantage of such a definition is that, if sampling has a strict positive probability $\rho(\xi) > 0, \forall \xi \in \mathcal{D}$, then each set $A \subset \mathbf{D}$ with a positive Lebesgue's measure $\mathrm{meas}(A) > 0$ also has a strictly positive sampling measure $\int_A d\,\overline{\rho} > 0$. Moreover, it is easy to observe that $\overline{\rho}$ is the density of some probabilistic measure from $\mathcal{M}(\mathbf{D})$.

Let us denote by $\overline{\theta(x)} \in L^p(\mathbf{D})$ the density associated with the sampling measure $\theta(x) \in \mathcal{M}(\mathcal{D})$ for some population's representative $x \in \mathcal{E}$. The convergence result similar to Theorem 7 holds:

Theorem 7 (Theorem 4.66 in Schaefer's book [57]). *Let us assume that the SGA heuristic operator is focusing, that its set of fixed points is finite* ($\#\mathcal{K} < +\infty$), *and that the mutation probability is strictly positive. Then,* $\forall \varepsilon > 0$, $\forall \eta > 0$, $\exists N \in \mathbb{N}$, $\exists W(N) \in \mathbb{N}$, $\exists z \in \mathcal{K}$ *it holds that:*

$$\forall \mu > N,\ \forall k > W(N)\quad \Pr\left\{ \left\| \overline{\theta(x_\mu^k)} - \overline{\theta(z)} \right\|_{L^p(\mathcal{D})} < c\,\varepsilon \right\} > 1 - \eta, \tag{12}$$

where

$$c = \frac{\mathrm{meas}(\mathbf{D})^{\frac{1}{p}}}{\min_{\xi \in \mathcal{D}} \{\mathrm{meas}(\vartheta_\xi)\}},\ p \in [1, +\infty).$$

Let us denote by $\mathcal{W} \subset \mathbf{D}$ the finite set of local minimizers of the misfit function, and by $\{\mathcal{B}_{\xi^+}\}, \xi^+ \in \mathcal{W}$, the family of their basins of attraction. Roughly speaking, the basin of attraction $\mathcal{B}_{\xi^+} \subset \mathbf{D}$ of the local, isolated minimizer ξ^+ is a connected part of the maximum level set such that it contains ξ^+ and a local strictly descent method starting from each point in $\mathcal{B}_{\xi^+}$ converges to ξ^+ (see the work of Boender et al. [9] or Schaefer's book [57] for details).

Definition 8 (Definition 4.63 in Schaefer's book [57])). *We say that the class of stochastic population-based search algorithms spanned by heuristic* $\mathcal{H}$ *is well-tuned to the set of local minimizers* $\mathcal{W} \subset \mathbf{D}$ *if:*

1. $\mathcal{H}$ *is focusing and the set of its fixed points* $\mathcal{K}$ *is finite,*
2. $\forall \xi^+ \in \mathcal{W}$, $\exists C(\xi^+)$ *closed set in* $\mathbf{D}$ *so that* $\xi^+ \in C(\xi^+) \subset \mathcal{B}_{\xi^+}$, $\mathrm{meas}(C(\xi^+)) > 0$ *and*

$$\overline{\theta(z)} \geqslant \textit{threshold} \quad \textit{almost everywhere on } C(\xi^+),$$

$$\overline{\theta(z)} < \textit{threshold} \quad \textit{almost everywhere on } \mathbf{D} \setminus \bigcup_{\xi^+ \in \mathcal{W}} C(\xi^+),$$

where $z \in \mathcal{K}$ *is an arbitrary fixed point for* $\mathcal{H}$ *and threshold is some positive constant, which we treat as the definition's parameter.*

The main idea of well-tuning can be expressed as follows: the class of search algorithms spanned by the heuristic $\mathcal{H}$ is well-tuned to the set $\mathcal{W}$ of local misfit minimizers under interest, if $L^p(\mathbf{D})$-regular measure densities $\overline{\theta(z)}$, associated with all fixed points $z \in \mathcal{K}$ of $\mathcal{H}$ dominate almost everywhere on the central parts of their basins of attraction $C(x^+) \subset \mathcal{B}_{x^+}, x^+ \in \mathcal{W}$.

3.4. Complexity theory of local and global search

In this section, we consider the problem of finding local and global optima from the point of view of a classic complexity theory. In particular, we present two results bounding the complexity of these two problems; that is, we show that the problem of finding a local optimum of a function is NP-complete, whereas the problem of finding a global optimum is DP-complete. Both results follow through a relatively straightforward translation of our problems to the world of complexity theory, but are nonetheless interesting. Indeed, the complexity class DP is significantly larger than

the class NP (provided that NP $\neq$ coNP, which is a standard complexity-theoretic assumption – see; e.g., the textbook of Papadimitriou [40]), and so the fact that finding local optima is NP-complete and finding global optima is DP-complete shows that the latter is a significantly more computationally intensive problem. While this is an obvious fact from the point of view of every practitioner, it is reassuring that this also shows in the complexity-theoretic view of the problems.

3.4.1. Basic notions from complexity theory

In this section, we very quickly review the most-essential notions from complexity theory relevant to our study. We point those readers interested in a more detailed exposition to the classic textbook of Papadimitriou [40].

Most of the time, complexity theory is concerned with decision problems; i.e., with problems which ask yes/no questions. Consider the following classic problem:

Definition 9. *In the* SAT-3CNF *problem, we are given a Boolean formula F over variables $x_1, \ldots, x_n$, in conjunctive normal form*[1]*, with at most three variables per clause. We ask if there is a truth-assignment for the variables so that F evaluates to truth.*

For example, the following formula is in conjunctive normal form, contains at most three variables per clause, and is satisfiable:

$$(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee \overline{x_3}). \tag{13}$$

To see that the formula is satisfiable, note that it suffices to assign the value *true* to each variable. From now on, depending on the context, we will either speak of Boolean variables as having values true/false or as having values 0/1 (the latter will be convenient when we mix logical and algebraic expressions).

Decision problems are modeled as sets of those inputs for which the answer is *yes*. That is, we can think of SAT-3CNF as of a set of those formulas in conjunctive normal form with (at most) three variables per clause that are satisfiable. (The standard way of modeling decision problems is through formal languages; but for our purposes, there is no need to give such a low-level discussion.)

The two most important classes of decision problems are P and NP. A decision problem A belongs to class P if there is an algorithm that solves it in polynomial time. A decision problem B belongs to class NP if there is a polynomial-time algorithm that, given an instance of B and a certificate-of-correctness for this problem, decides if this certificate is indeed correct. The exact nature of the certificate depends on the problem at hand; the only requirement is that its length has to be polynomially bounded in the length of the input instance. For example, for SAT-3CNF, the certificates could simply be the valuations of all of the variables: Given a Boolean formula and the values of all of the variables, it is trivial to verify if the formula evaluates to the *truth*

[1] A formula is in conjunctive normal form if it is a conjunction of clauses, where each clause is a disjunction of variables or their negations.

or not. If it does, the certificate is correct. Thus, SAT-3CNF is in NP (but it is not known to be in P).

There is a natural partial order of hardness among decision problems, defined through the notion of a polynomial-time reduction.

Definition 10. *We say that decision problem A (polynomial-time many-one) reduces to decision problem B (denoted as $A \leqslant_m^p B$) if there exists a polynomial-time computable function x such that, for each possible input x for A, it holds that $x \in A \iff f(x) \in B$.*

In other words, a reduction converts its input from the format of one problem to the format of the other while maintaining the answer. Clearly, the problem to which we reduce has to be at least as hard as the problem from which we reduce.

A decision problem B is NP-complete if it belongs to NP and if every problem A from NP reduces to it. In other words, NP-complete problems are exactly the hardest problems in NP. It is well-known that SAT-3CNF is NP-complete [15].

In our discussion, we will occasionally refer to more advanced notions and results from complexity theory. In such cases (e.g., when discussing the class DP), we will provide necessary definitions as needed.

3.4.2. The problems of finding local and global optima

The main idea behind the results of this section is that we can encode various hard computational problems as a search for local and global optima. While doing so is quite easy, it requires some care at the formal level. We start by providing definitions of the problems of finding local/global optima. (We mention this for the ease of presentation, we consider one-dimensional problems only; it should be straightforward to see that all of our results directly translate to the multidimensional cases.)

Definition 11. *In the* LocalOptimum *problem, we are given function $f\colon (0,1) \to \mathbb{R}$, and our goal is to compute a local maximum x_0; that is, a value such that there is $\varepsilon > 0$ such that for each $x \in (x_0 - \varepsilon, x_0 + \varepsilon)$ it holds that $f(x_0) \geqslant f(x)$.*

Definition 12. *In the* GlobalOptimum *problem, we are given function $f\colon (0,1) \to \mathbb{R}$, and our goal is to compute a global maximum x_0; that is, a value such that for each $x \in (0,1)$ it holds that $f(x_0) \geqslant f(x)$.*

While intuitively clear, the two problems above are not well-defined. In particular, we need to decide how the input functions are encoded and how accurate we want our computations to be. Regarding accuracy, we assume that we are given accuracy parameter t and are looking for a solution in the set $\{\frac{i}{2^t} \mid 1 \leqslant i \leqslant 2^t - 1\}$. In other words, for simplicity, we are working in a t-bit fixed-point binary system. Regarding the issue of representing functions, we use the standard model of Boolean circuits (see; e.g., the text of Papadimitriou [40] regarding basic models of computation). Briefly put, a Boolean circuit that represents function f consists of t 0/1 inputs (modeling the t bits of f's input argument), $t' \geqslant t$ 0/1 outputs (modeling the t' bits of f's output; we assume that f's values can be greater than 1, but we implicitly assume that f is bounded), and a number of logical gates connecting inputs, outputs, and each other

(these logical gates include AND gates, OR gates, and NOT gates). It is well-known that every function $f\colon \{0,1\}^t \to \{0,1\}^{t'}$ that is computable in polynomial time with respect to $O(t+t')$ can be represented as a Boolean circuit of size polynomial in $O(t+t')$. This means that we can simply view input f of LocalOptimum and of GlobalOptimum as an arbitrary polynomial-time computable function.

To be formally correct and in sync with the discussion above, we should rephrase our definitions of LocalOptimum and GlobalOptimum to speak of functions encoded using Boolean circuits, and to explicitly take into account that these functions' inputs are t-bit integers, and their outputs are t'-bit integers. However, we choose to omit such technicalities in the hope that this will help keep our discussion clearer. We stress, however, that if one wished to go through such an exercise of being formally correct, all of our results would stay intact.

Nonetheless, we need to make one more adjustment to our problems. As we have said, classic complexity theory is geared towards working with decision problems, whereas LocalOptimum and GlobalOptimum are so-called search problems (i.e., problems that seek a value with some specified properties). We reformulate them as decision problems in a straightforward way.

Definition 13. *In the* DecisionLocalOptimum *problem, we are given a function* $f\colon (0,1) \to \mathbb{R}$ *(encoded as a Boolean circuit, with input precision* t *and output precision* t'*, both given as part of the input in the unary encoding), and a value* y_0*. We ask if there is a point* $x_0 \in \{\frac{i}{2^t} \mid 1 \leqslant i \leqslant 2^t - 1\}$ *such that* $f(x_0) \geqslant f(x_0 - \frac{1}{2^t})$*,* $f(x_0) \geqslant f(x_0 + \frac{1}{2^t})$*, and* $f(x_0) \geqslant y_0$*.*

Definition 14. *In the* DecisionGlobalOptimum *problem, we are given a function* $f\colon (0,1) \to \mathbb{R}$ *(encoded as a Boolean circuit, with input precision* t *and output precision* t'*, both given as part of the input in the unary encoding), and a value* y_0*. We ask if there is a point* $x_0 \in \{\frac{i}{2^t} \mid 1 \leqslant i \leqslant 2^t - 1\}$ *such that* $f(x_0) \geqslant y_0$*, and for every other point* x *in this set, it holds that* $f(x) \leqslant y_0$*.*

We mention that these are not the only natural ways of translating the intuitive variants of our problems into decision variants. For example, instead of asking about the existence of an optimum with at least a given value, we could be asking if there is an optimum of a given type in a particular open interval.

3.4.3. Finding a local optimum is NP-complete

We now show that the problem of finding a local optimum is NP-complete. To this end, we will give a reduction from SAT-3CNF. We will use the following notation: given an integer t, $0 \leqslant t \leqslant 2^n - 1$, we write $\langle t_1 t_2 \dots t_n \rangle$ to mean its n-bit binary representation (so each t_i, $1 \leqslant i \leqslant n$, is either 0 or 1, and we have $t = \sum_{i=1}^{n} t_i 2^{i-1}$). Given a nonnegative integer t and a Boolean formula F over variables $x_1, \dots, x_n$, we write $F(t)$ to mean the logical value of F evaluated for $x_1 = t_1, \dots, x_n = t_n$, where $\langle t_1 \dots t_n \rangle$ is the n-bit binary representation of t.

Theorem 15. DecisionLocalOptimum *is* NP*-complete.*

Proof. It is easy to see that the problem is in NP. Let f be our input function and t be the precision of the encoding used (we assume that t is represented in unary). It suffices to note that a nondeterministic algorithm can guess the point x_0 that is supposedly a local optimum (by convention, x_0 is of the form $\frac{i}{2^t}$, $1 \leqslant i \leqslant 2^t - 1$) and check that indeed we have $f(x_0) \geqslant f(x_0 - \frac{1}{2^t})$, $f(x_0) \geqslant f(x_0 + \frac{1}{2^t})$, and $f(x_0) \geqslant y_0$.

We now give a reduction from SAT-3CNF to LocalOptimum. Let F be our input formula over variables $x_1, \ldots x_n$. For each i, $0 \leqslant i \leqslant 2^{n-1}$, we define the following two functions:

$$g_i^n(x) = \begin{cases} \frac{1}{2}x, & \text{if } x \in [\frac{i}{2^n}, \frac{i+1}{2^n}), \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

$$h_i^n(x) = \begin{cases} (2^{n+1} - i)x - 2i + \frac{i^2}{2^n} + \frac{i}{2^{n+1}}, & \text{if } x \in [\frac{i}{2^n}, \frac{i}{2^n} + \frac{1}{2^{n+1}}), \\ (-2^{n+1} + 2i + 2)x + \frac{-2i^2 - 3i - 1}{2^n} + 2i + 2, & \text{if } x \in [\frac{i}{2^n} + \frac{1}{2^{n+1}}, \frac{i+1}{2^n}), \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

While a bit contrived, these two functions are very simple. First, let us note that, for each i, $0 \leqslant i \leqslant 2^n - 1$, both g_i and h_i are nonzero only on the interval $[\frac{i}{2^n}, \frac{i+1}{2^n})$ (with the exception of $i = 0$, where the interval is open on both ends). Second, h_i is, in essence, just a sequence of parts of a linear function, and indeed, we have that

$$g(x) = \sum_{i=0}^{2^n - 1} g_i^n(x)$$

is simply $\frac{1}{2}x$ over the interval $[0, 1)$. Third, for each i, $0 \leqslant i \leqslant 2^n - 1$, we have that h_i^n over interval $[\frac{i}{2^n})$ for the first half of the interval is a straight line that increases from $\frac{1}{2} \cdot \frac{i}{2^n}$ to 1 (which it reaches exactly at point $x = \frac{i}{2^n} + \frac{1}{2^{n+1}}$), and then it becomes a straight line that moves toward value $\frac{1}{2} \cdot \frac{i+1}{2^n}$.

Given the families of functions g_i^n and h_i^n, $0 \leqslant i \leqslant 2^n - 1$, we define our target function as follows:

$$f(x) = \sum_{i=0}^{2^n - 1} \left(F(i)h_i^n(x) + (1 - F(i))g_i^n(x)\right).$$

Intuitively put, $f(x)$ is a linear function $\frac{1}{2}x$ that occasionally, on intervals $[\frac{i}{2^n}, \frac{i+1}{2^n})$ such that the n-bit binary representation of i corresponds to a satisfying truth assignment for F, spikes up to value 1 and then goes down to become the linear function again. It is easy to verify that $f(x)$ is a continuous function, and if one insisted, one could even define f to be infinitely differentiable (but then its definition would be much more involved).

Finally, it is easy to see that $f(x)$ is computable in time polynomial with respect each chosen precision t and the value n. Thus, it can be represented as a Boolean circuit.

Our reduction outputs the circuit-encoded function f, input precision $t = n + 1$, output precision $t' = t$, and value $y_0 = 1$. By the very definition of f and the fact that our input precision is $t = n + 1$, it is easy to see that our function f has a local optimum with value at least 1 if and only if F is satisfiable. □

The above result should not come as a surprise. It is well-known that various optimization problems indeed are NP-complete. Its value stems from two facts: first, it is expressed in the language of searching for local optima of continuous functions. Second, we will use it as a baseline for the comparison regarding the complexity of global search problems.

3.4.4. Finding a Global Optimum Is DP-Complete

Let us now move on to the case of the DecisionGlobalOptimum problem. Intuitively, finding a global optimum is much harder than finding a local one, because given a function f, it requires one to find a point $x_0 \in (0, 1)$ such that, for all $x \in (0, 1)$, we have $f(x_0) \geqslant f(x)$; on the other hand, when looking for a local optimum, it suffices to find a point whose value was greater than that of its two neighbors. In this section, we give a formal argument that this intuition is correct by showing that DecisionGlobalOptimum is DP-complete.

The complexity class DP is not very well known outside the structural complexity theory literature, and so we will now recall its definition and provide a very brief discussion of its properties.

Definition 16 (Papadimitriou and Yannakakis [42])**.** *We say that a decision problem L belongs to the class* DP *if there are two decision problems $A, B \in$ NP such that $L = A - B$.*

In other words, a problem L belongs to the class DP if there are two problems, A and B, from the class NP such that L accepts all of the instances accepted by A, except those accepted by B. In effect, it is easy to see that NP $\subseteq$ DP. If A belongs to NP, then by taking $B = \emptyset$ (which, of course, belongs to NP), we have that $A \in$ DP. On the other hand, we also have that coNP $\subseteq$ DP (coNP is the set of all decision problems from NP with the answer reversed; it is widely believed that NP $\neq$ coNP.) In effect, unless NP = coNP (which is not believed to be true), we have that DP is a strictly larger class than NP. This also means that DP-complete problems are strictly harder than NP-complete ones. Yet, DP-complete problems are not *much* harder than NP-complete ones. This follows from the fact that DP is a subclass of the second level of the polynomial hierarchy:

$$\text{NP} \subseteq \text{DP} \subseteq \text{P}^{\text{NP}} \subseteq \text{PH}.$$

(We point the readers not familiar with the polynomial hierarchy either to the original research paper of Stockmeyer [71], or to the textbook of Papadimitriou [40], or to the textbook of Hemaspaandra and Ogihara [26] for a general overview of a number of complexity classes, their properties, and their complete problems.)

The class DP has many natural complete problems, many of which indeed model some form of a search for a global optimum. In particular, we will use the following one:

Definition 17. *In the* MaxClique *problem, we are given a graph $G = (V, E)$, an integer k, and we ask if the largest clique in G (that is, the largest complete subgraph of G) consists of exactly k vertices.*

Using the fact that MaxClique is DP complete, we will show that GlobalOptimum is DP-complete as well.

Theorem 18. DecisionGlobalOptimum *is* DP*-complete.*

Proof. We write $\langle f, y_0 \rangle$ to denote an input for the DecisionGlobalOptimum, where f is a function (encoded as a Boolean circuit, with t and t' being the input/output accuracies, encoded in unary, and available implicitly as part of f's description), and where y_0 is a value.

We first show membership of DecisionGlobalOptimum in DP. To this end, we define the following two problems:

1. In the MatchOptimum we are given input $\langle f, y_0 \rangle$ and we ask if there is x_0 such that $f(x_0) \geqslant y_0$.
2. In the ExceedOptimum we are given input $\langle f, y_0 \rangle$ and we ask if there is a point x such that $f(x) > y_0$.

It is straightforward to see that the following holds:

$$\langle f, y_0 \rangle \in \text{DecisionGlobalOptimum} \iff \langle f, y_0 \rangle \in \text{MatchOptimum} \wedge \langle f, y_0 \rangle \notin \text{ExceedOptimum}$$

This means that:

$$\text{DecisionGlobalOptimum} = \text{MatchOptimum} - \text{ExceedOptimum}.$$

Since it is easy to see that both MatchOptimum and ExceedOptimum are in NP, we have that DecisionGlobalOptimum is in DP.

To show that the problem is DP-hard, we will give a reduction from the MaxClique problem. Consider an input instance of MaxClique with graph $G = (V, E)$ and number k. We ask if G's largest clique has exactly k vertices. For ease of notation, we rename G's vertices so that $V = \{1, \ldots, n\}$.

Recall from Section 3.4.3 that we write $\langle i_1, \ldots, i_n \rangle$ to mean the binary representation of a number i, $0 \leqslant i \leqslant 2^n - 1$. We define a function $F(i)$, $0 \leqslant i \leqslant 2^n - 1$, as follows:

$$F(i) = \begin{cases} \sum_{j=1}^{n} i_j, & \text{if for each } j, j' \text{ such that } i_j = 1 \text{ and } i_{j'} = 1 \text{ there} \\ & \text{is an edge connecting vertices } j \text{ and } j' \text{ in } G, \\ 0, & \text{otherwise.} \end{cases}$$

In other words, if the binary encoding of i corresponds to a clique in G, then $F(i)$ is the size of this clique, and otherwise it is 0. We will also need the following family of functions (for each function h_i^n, we have that i is an integer, $0 \leqslant i \leqslant 2^n - 1$):

$$h_i^n(x) = \begin{cases} 2^{n+1}(x - \frac{i}{2^n}), & \text{if } x \in [\frac{i}{2^n}, \frac{i}{2^n} + \frac{1}{2^{n+1}}), \\ 2^{n+1}(\frac{i+1}{2^n} - x), & \text{if } x \in [\frac{i}{2^n} + \frac{1}{2^{n+1}}, \frac{i+1}{2^n}), \\ 0, & \text{otherwise} \end{cases}$$

Note that, for each i, $0 \leqslant i \leqslant 2^{n-1}$, we have that $h_i^n(x)$ is simply a function that is 0 on the whole interval $(0, 1)$ except that on the subinterval $(\frac{i}{2^n}, \frac{i+1}{2^n})$ it linearly spikes up to 1 (which it reaches at $\frac{i}{2^n} + \frac{1}{2^{n+1}}$), and then linearly moves down to 0.

We define our DecisionGlobalOptimum function f as follows:

$$f(x) = x + \sum_{i=0}^{2^n-1} F(i) h_i^n(i).$$

The basic idea behind function f is that it behaves like a linear function (the x component), except that, if for a given i the encoding $\langle i_1, \ldots, i_n \rangle$ of i describes a clique of size k, then the function moves up to $x + k$ and then moves back down to x. It is thus clear that a global optimum of f corresponds to the largest clique of G. To put it formally, our reduction outputs $\langle f, y_0 \rangle$, where f is the just-defined function, $y_0 = k$, and the input precision is $t = n + 1$. (Note that the setting of the input precision is extremely important for the correctness of the proof.) The output precision is irrelevant as long as we can output integers with values between 0 and $n + 1$.

To see that the reduction is correct, it suffices to see that if an integer i describes a clique of size k in G, then $k \leqslant f(\frac{i}{2^n} + \frac{1}{2^{n+1}}) < k + 1$.

It is also quite clear that one can derive a Boolean circuit describing function f in polynomial time. This concludes the proof. □

This result requires some discussion. First, the reader may question the usefulness of the DecisionGlobalOptimum problem compared to its search variant, GlobalOptimum, because the former requires us to specify the value of the global optimum and simply verifies if, indeed, there is a global optimum with a given value, whereas the latter provides the point which is the global optimum. However, our main goal in this discussion is to formally justify that the problem of finding a global optimum is much harder than the problem of finding a local one. Our result shows that, even in a much-simplified setting where we do have a good guess of the value of the global optimum, the problem of verifying our guess has higher computational complexity (DP-completeness) than the problem of finding a local optimum (NP-completeness).

4. Research directions and first results

In this section, we present some preliminary ideas for establishing the computational cost of global search algorithms. In particular, we focus on the first hitting time for

well-tuned genetic search algorithms, we consider the first hitting time for hierarchic search strategies, and we show that, from the point of view of complexity theory, finding all global optima is significantly more difficult than finding a single one.

4.1. Evaluation of FHT for well-tuned genetic searches

4.1.1. Single-step evaluations

Let us assume that $S \subset \mathcal{D}$ is a set of points that might be called "solutions" to the global phase of the inverse problem. S might be a set of phenotypes located in the basins of attraction of the misfit minimizers.

Let $\rho^0 \in \mathcal{M}(\mathcal{D})$ be the sampling measure used for creating the initial population P_0. Applying the Bernoulli sampling rule for μ–times sampling with return with probability of success $\rho^0(S)$, we obtain:

Observation 1. *The probability of sampling at least one individual from S to the initial population equals $p_0 = 1 - (1 - \rho^0(S))^\mu$, while the probability of sampling the initial population without solution individuals is $p'_0 = (1 - \rho^0(S))^\mu$.*

Analogously, if the particular population vector $x \in X_\mu$ appeared in a t-th epoch, then:

Observation 2. *The probability of sampling at least one individual from S to the population in the $(t+1)$-th epoch equals $p_x = 1-(1-\theta(\mathcal{H}(x))(S))^\mu$ while the probability of sampling this population without any individuals from S is $p'_x = (1-\theta(\mathcal{H}(x))(S))^\mu$, where x is the population vector for the t-th epoch.*

Next, using Bayes' rule, we get:

Observation 3. *Assuming the initial probability distribution is π^0_μ, the probability of sampling at least one individual from S in the $(t+1)$-th epoch is:*

$$\sum_{x,y \in X_\mu} (Q^t)_{xy} \, (\pi^0_\mu)_y \, p_x, \tag{16}$$

whereas the probability that the $(t+1)$-th epoch does not contain any individual from S is:

$$1 - \sum_{x,y \in X_\mu} (Q^t)_{xy} \, (\pi^0_\mu)_y \, p_x = \sum_{x,y \in X_\mu} (Q^t)_{xy} \, (\pi^0_\mu)_y \, p'_x. \tag{17}$$

Note that all components of formulas (16) and (17) can be computed using heuristic $\mathcal{H}$.

4.1.2. Asymptotic property

We consider the set of all populations from X_μ that contains solutions:

$$X^S_\mu = \{x \in X_\mu : x \text{ contains individuals from } S\}. \tag{18}$$

Let A_k be the event that $\bigcup_{t=0}^k P_t \cap X^S_\mu \neq \emptyset$, where $P_0, P_1, \ldots, P_k$ is the sequence of μ–sized populations generated by the stochastic search algorithm with heuristic $\mathcal{H}$. We assume that the heuristic has a single fixed point $\hat{x} \in \mathcal{E}$.

Assuming some small $\varepsilon > 0$, we can evaluate $\Pr(A_t)$ using Bayes' formula:

$$\begin{aligned} \Pr(A_t) &= \Pr(A_t | x_\mu^t \in K_\varepsilon(\hat{x})) \cdot \Pr(x_\mu^t \in K_\varepsilon(\hat{x})) \\ &\quad + \Pr(A_t | x_\mu^t \in \mathcal{E} \setminus K_\varepsilon(\hat{x})) \cdot \Pr(x_\mu^t \in \mathcal{E} \setminus K_\varepsilon(\hat{x})). \end{aligned} \tag{19}$$

Respecting the assumptions of Theorem 6, according to the genetic search process, we have:

$$\Pr(x_\mu^t \in K_\varepsilon(\hat{x})) > 1 - \eta, \;\; \Pr(x_\mu^t \in \mathcal{E} \setminus K_\varepsilon(\hat{x})) \leqslant 1 - \eta, \tag{20}$$

where $\eta > 0$ is arbitrary small and μ and t are sufficiently large.

The following additional assumptions are reasonable for the Simple Genetic Processes well-tuned to the set of local minimizers $\mathcal{K} \subset S$ (see Definition 8):

1. $\varepsilon_1 < \varepsilon_2 \Rightarrow \Pr(A_t | x_\mu^t \in K_{\varepsilon_1}(\hat{x})) \geqslant \Pr(A_t | x_\mu^t \in K_{\varepsilon_2}(\hat{x}))$,
2. $\exists \varepsilon_0 > 0;\ \forall \varepsilon \leqslant \varepsilon_0,\ \forall t > 0 \;\; \Pr(A_t | x_\mu^t \in K_\varepsilon(\hat{x})) = 1$, or, in other words $K_\varepsilon(\hat{x})) \subset X_\mu^S \subset X_\mu$, where X_μ^S is the set of population vectors such that each population represented by its element contains and individual from S.

We make the following observation:

Observation 4. *Let us consider the class of Simple Genetic Processes spanned with focusing heuristic $\mathcal{H}$, well-tuned to the set of local minimizers $\mathcal{K} \subset S$. If Assumption 2 (see a few lines above) holds, then for a sufficiently small $\varepsilon > 0$ we have:*

$$Pr(A_t) \geqslant Pr(x_\mu^t \in K_\varepsilon(\hat{x})). \tag{21}$$

Moreover, if μ and t are sufficiently large, then $Pr(A_t)$ is arbitrary close to $Pr(x_\mu^t \in K_\varepsilon(\hat{x}))$, while $Pr(A_t | x_\mu^t \in \mathcal{E} \setminus K_\varepsilon(\hat{x})) \cdot Pr(x_\mu^t \in \mathcal{E} \setminus K_\varepsilon(\hat{x}))$ vanishes.

4.1.3. First hitting time

The probability of hitting S in exactly time step t equals:

$$\begin{aligned} &\Pr\left(x_t \in X_\mu^S \middle| \; x_0 \notin X_\mu^S, \ldots, x_{t-1} \notin X_\mu^S\right) = \\ &\textstyle\sum_{y_t \in X_\mu^S} \sum_{y_0, \ldots, y_{t-1} \notin X_\mu^S} Q_{y_0 y_1} \cdots Q_{y_{t-2} y_{t-1}} Q_{y_{t-1} y_t} (\pi_\mu^0)_{y_0}. \end{aligned} \tag{22}$$

The first hitting time of S is the random variable:

$$H^S = \inf\left\{t \geqslant 0 : x_t \in X_\mu^S\right\}. \tag{23}$$

From (22) it follows that:

$$\Pr\left(H^S = t\right) = \sum_{y_t \in X_\mu^S} \sum_{y_0, \ldots, y_{t-1} \notin X_\mu^S} Q_{y_0 y_1} \cdots Q_{y_{t-2} y_{t-1}} Q_{y_{t-1} y_t} (\pi_\mu^0)_{y_0}. \tag{24}$$

The statistics of H^S can be computed by means of (24). In particular, its mean value is given by the following formula:

$$E(H^S) = \begin{cases} +\infty & \text{if } \Pr\left(x_t \notin X_\mu^S \text{ for all } t\right) > 0, \\ \sum_{t=0}^{\infty} t \Pr\left(H^S = t\right) & \text{otherwise.} \end{cases} \tag{25}$$

For ergodic chains the first case of (25) cannot hold. In such a case, from (22), we have:

$$E(H^S) = \sum_{t=0}^{\infty} t \sum_{y_t \in X_\mu^S} \sum_{y_0,\ldots,y_{t-1} \notin X_\mu^S} Q_{y_0 y_1} \ldots Q_{y_{t-2} y_{t-1}} Q_{y_{t-1} y_t} (\pi_\mu^0)_{y_0}. \tag{26}$$

In particular, when the initial distribution is concentrated in a single individual x_0, the above expected value equals:

$$E_{x_0}(H^S) = \sum_{t=0}^{\infty} t \sum_{y_t \in X_\mu^S} \sum_{y_1,\ldots,y_{t-1} \notin X_\mu^S} Q_{x_0 y_1} \ldots Q_{y_{t-2} y_{t-1}} Q_{y_{t-1} y_t}. \tag{27}$$

It can be easily proven (cf. [38]) that $E_{x_0}(H^S)$ is the unique nonnegative solution to the following linear system:

$$\begin{cases} E_x(H^S) = 0 & \text{for } x \in X_\mu^S \\ E_x(H^S) = 1 + \sum_{y \in X_\mu} Q_{xy} E_y(H^S) & \text{for } x \notin X_\mu^S. \end{cases} \tag{28}$$

Note that, since X_μ is finite, $E(H^S)$ can be computed using (28) for any initial distribution π_μ^0.

4.2. Generalization of FHT evaluation for hierarchic strategies

The formulation of the HGS dynamics Markov model [59] gives us the opportunity to compute the first hitting time for this complex strategy as well. Since the description of the model is rather complicated, we recall here only some of its elements in a form that is simplified but suitable for derivation of the first hitting time for this strategy. See the work of Schaefer et al. [59] for the general formulation and all necessary details.

4.2.1. HGS Tree

The deme structure of the HGS strategy is mapped into graph $HGSTREE = \langle V, E, F \rangle$ (see Section 4.1 in the work Schaefer et al. [59]), where: (1) V is the set of nodes representing all possible demes allowed by the applied encoding and additional restrictions introduced in the HGS instance under consideration. (2) The set of edges E imposes a tree structure of depth m. (3) The set of labels F contains integer vectors of length m that form the complete collection of paths to each node from the root.

The number of children of the *HGSTREE* nodes may vary through levels, but on each level, i is bounded by $k_i, i = 1, \ldots, m-1$. The root deme may have, at most, k_1 child-demes, and each deme at each i-th level has, at most, k_i child-demes. The maximum, total number of demes at the i-th level equals: 1 for $i = 1$ and $g_i = \prod_{s=1}^{i-1} k_s$ for $1 < i < m$.

We assume later that we will use a discrete encoding, the same for all demes at each particular level of the *HGSTREE*. Moreover, all demes at the particular i-th level have the same size μ_i. Of course, the encoding and μ_i may vary among *HGSTREE* levels.

4.2.2. The space of states

Roughly speaking, the state of the Markov chain modeling HGS is a tree of states of all potentially existing demes associated to the nodes of the *HGSTREE* (see Section 4.2 in the work Schaefer et al. [59]).

Let us recall of the notation of deme representation introduced in Section 3.3. Respecting the imposed assumptions, we denote by $X^i_{\mu_i}$ the set of state vectors of all demes at the i-th level of the *HGSTREE*. Consequently, the evolution progress of such a deme is characterized by vector $x \in X^i_{\mu_i}$.

Moreover, each deme (except for the root) may take the following status: *inactive*, *new*, *active*, *stopped*. The root deme may take only two of them, *active* or *stopped*. It becomes *active* just after the start of HGS and is switched to *stopped* after the global stopping condition is satisfied. The status of each branch or leaf deme is:

- *inactive* from the start of HGS until it is sprouted by the parental deme;
- *new*, immediately after it has been sprouted; moreover, the sprouting operation sets the initial value of the deme's state vector (creates an initial multiset of individuals); just after, during the forthcoming metaepoch, the state *new* is switched to *active*, or to *stopped* if the efficiency stopping condition is satisfied;
- *active*, if it was previously *new* or *active* and did not meet the efficiency stopping condition;
- *stopped*, if the efficiency stopping condition was satisfied in the current metaepoch, or has been satisfied earlier; moreover, the branch demes are switched to the state *stopped* if the number of its active or stopped child demes reached the prescribed maximum; the state of a deme once stopped cannot be changed.

For the sake of completeness of the stochastic model, the deme state vectors of all branches and leaves (except the root node) are filled by an arbitrary values at the start of HGS. These starting values do not affect the computation results and are removed just after the particular deme becomes *new*.

The state space of the HGS Markov model is a subset of the space:

$$\begin{aligned} X \quad = \quad & \{active, stopped\} \times X^1_{\mu_1} \times \\ & \times \prod_{i=2}^{m} \left(\prod_{p=1}^{g_i} \left(\{inactive, new, active, stopped\} \times X^i_{\mu_i} \right) \right). \end{aligned} \tag{29}$$

Let us introduce:

$$deme_p^i : X \longrightarrow X_{\mu_i}^i \tag{30}$$

to denote the projection that maps an HGS state to its coordinates related to the state of deme p on level j, and let us write:

$$status_p^i : X \longrightarrow \{inactive, new, active, stopped\} \tag{31}$$

to denote the projection extracting the status of the p-th deme on j-th level from the whole system state.

It was shown by Schaefer et al. [59] that the passage between two consecutive HGS states might be described by Markov transition probability function $\tau : X \to \mathcal{M}(X)$. This mapping considers all stochastic operations performed by the state passage; i.e., evolution in all active demes (selection, crossover, mutation, and succession), efficiency stopping condition evaluation (possibly changing deme status to *stopped*), and conditional sprouting. Since state space X is finite, the Markov transition probability function is characterized by probability transition matrix Q.

4.2.3. Computing first hitting time for HGS

Let us once again write $S \subset \mathcal{D}$ to denote all points called "solutions," as in the case of single-population search algorithms (see Section 4.1.1). There are at least two possible ways of defining an interesting subset of the HGS states related to S that the hierarchic search is intended to reach.

The first, simpler one is to take those states that contain at least one leaf deme that incorporates at least one individual with a phenotype located in S.

$$X_1^S = \big\{x \in X |\ \exists p \in \{1, \ldots, g_m\} : deme_p^m(x) \text{ contains individuals from } S\ , \\ status_p^m(x) \in \{active, stopped\}\big\} \tag{32}$$

In the second case (more-suitable for multi-modal misfit functions), S is the union of disjoint connected components $S_1, \ldots, S_w$. Each S_i may be a level set located in the central part of the basin of attraction of a single local/global minimizer for the misfit function. In this case, the target subset of HGS states are such that, for every set $S_i, i = 1, \ldots, w$, there exists at least one leaf deme containing at least one individual with a phenotype located in S_i. In both cases, the leaves might be *active* or *stopped*.

$$X_{all}^S = \big\{x \in X |\ \forall i \in \{1, \ldots, w\} \exists p_i \in \{1, \ldots, g_m\} : \\ deme_{p_i}^m(x) \text{ contains individuals from } S_i, \\ status_{p_i}^m(x) \in \{active, stopped\}\big\} \tag{33}$$

Accordingly, in the case of HGS, we can consider two variants of the first hitting time for S. Namely, we define the following two random variables:

$$H_1^S = \inf\left\{t \geqslant 0 : x_t \in X_1^S\right\}, \tag{34}$$

$$H_{all}^S = \inf\left\{t \geqslant 0 : x_t \in X_{all}^S\right\}. \tag{35}$$

For both, we can repeat the results of Section 4.1.3. First of all:

$$\begin{aligned}&\Pr\left(x_t \in X_*^S |\ x_0 \notin X_*^S, \ldots, x_{t-1} \notin X_*^S\right) = \\ &\textstyle\sum_{y_t \in X_*^S} \sum_{y_0,\ldots,y_{t-1} \notin X_*^S} Q_{y_0 y_1} \cdots Q_{y_{t-2} y_{t-1}} Q_{y_{t-1} y_t} \pi^0_{y_0},\end{aligned} \tag{36}$$

where $*$ stands for 1 or *all* and π^0 is the initial distribution. Therefore, we have:

$$\Pr\left(H_*^S = t\right) = \sum_{y_t \in X_*^S} \sum_{y_0,\ldots,y_{t-1} \notin X_*^S} Q_{y_0 y_1} \cdots Q_{y_{t-2} y_{t-1}} Q_{y_{t-1} y_t} \pi^0_{y_0}. \tag{37}$$

The above equality allows us to compute the statistics of H_*^S. In particular, we can compute its expected value as follows:

$$E(H_*^S) = \begin{cases} +\infty & \text{if } \Pr\left(x_t \notin X_*^S \text{ for all } t\right) > 0, \\ \displaystyle\sum_{t=0}^{\infty} t \Pr\left(H_*^S = t\right) & \text{otherwise.} \end{cases} \tag{38}$$

Observation 5. *It is quite easy to see that the HGS Markov chain cannot be ergodic. Nevertheless, due to the possible SGA/SEA ergodicity, each leaf (in fact, each deme) is ergodic, provided the mutation rate is positive. Therefore, in the H_1^S case, if we guarantee that all levels are sprouted at least once, then we can be sure that the first clause of* (38) *does not hold. This is because we shall obtain at least one leaf, and that leaf shall reach S in a finite time. The H_{all}^S case is more complicated, as we have to force multiple sprouting to obtain at least one leaf per connected component of S.*

In any case, when the initial distribution is concentrated in a single point x_0, from (37), we have:

$$E_{x_0}(H_*^S) = \begin{cases} +\infty & \text{if } \Pr\left(x_t \notin X_*^S\right) > 0, \\ \displaystyle\sum_{t=0}^{\infty} t \sum_{y_t \in X_*^S, y_1, \ldots, y_{t-1} \notin X_*^S} Q_{x_0 y_1} Q_{y_1 y_2} \cdots Q_{y_{t-1} y_t} & \text{otherwise.} \end{cases} \tag{39}$$

And again, as in Section 4.1.3, it can be proven that $E_{x_0}(H^S)$ is the unique nonnegative solution to the following linear system:

$$\begin{cases} E_x(H_*^S) = 0 & \text{for } x \in X_*^S \\ E_x(H_*^S) = 1 + \sum_{y \in X} Q_{xy} E_y(H_*^S) & \text{for } x \notin X_*^S. \end{cases} \tag{40}$$

Note that, since X is finite, $E(H_*^S)$ can be computed using (40) for any initial distribution π^0.

4.3. Complexity theory of finding all global optima

Let us now step back to the ideas from Section 3.4 regarding the computational complexity of global and local searches. One of the important aspects of memetic

search algorithms in general (and of HGS specifically) is their ability to find many (or all) global optima. In this section, we show that, from the point of view of complexity theory, this task is much more demanding than the task of finding a single optimum point.

There are many ways in which one can model the problem of finding all global optima. Perhaps the most natural one is to require the solver to output a list of all of them. From the point of view of theoretical analysis, such an approach is, however, difficult. The reason is that, in our complexity-theoretic view of representing functions (as Boolean circuits, with a given precision), it is perfectly possible that a function might have exponentially many optima (compared to the length of its encoding). Then, even the time needed to output them all (short of doing the actual computation) would be exponential, and all complexity-theoretic analysis would be meaningless. Thus, to avoid this problem, we consider the problem of counting how many global optima there are.

Definition 19. *In the* #GlobalOptimum *problem, we are given a function* $f\colon (0,1) \to \mathbb{R}$*, and our goal is to count how many points* x_0 *there are such that, for each* $x \in (0,1)$*, it holds that* $f(x_0) \geqslant f(x)$*.*

How difficult is #GlobalOptimum? To answer this question, we need to consider complexity theory of counting functions and its most prominent complexity class, #P [67]. Intuitively speaking, #P is a counting variant of class NP. Formally, it is defined as follows: let A be some problem from class NP associated with some scheme of providing certificates of membership (for example, recall that for SAT-3CNF, a certificate was a valuation of a given formula's variables under which the formula evaluates to truth). Problem $\#A$, the counting variant of A, is the problem where, given an input for A, we ask how many different certificates there are for it. Class #P is the class of all counting problems associated with the problems from NP[2]. However, it is much more natural to think of #P in terms of its complete problems, of which one of the best-known is the counting variant of the SAT-3CNF problem.

Definition 20. *In* #SAT-3CNF *we are given Boolean formula* F*, over variables* $x_1, \ldots, x_n$*, in conjunctive normal form, with at most three variables per clause. We ask how many satisfying truth-assignments there are for* F*.*

#SAT-3CNF is #P-complete in the strictest possible sense, that is, with respect to parsimonious reductions (a discussion of various types of reductions between counting problems is included (e.g., in the work of Faliszewski and Ogihara [23]); the classic discussion of different reducibility types for decision problems is due to Ladner, Lynch, and Selman [32]).

Definition 21. *Let* F *and* G *be two functions (that output nonnegative integers). We say that* F *parsimoniously reduces to* G *if there is a polynomial-time computable function* ϕ *such that, for each* F*'s input* x*, it holds that* $F(x) = G(\phi(x))$*.*

[2]The formal definition of the class is more technical, and introducing it here is not necessary for our discussion.

Using exactly the same reduction as from SAT-3CNF to LocalOptimum (Theorem 15), it is easy to show that #SAT-3CNF reduces (in the parsimonious sense) to #GlobalOptimum. In effect, we get the following result (a problem is #P-hard if every problem in #P reduces to it):

Theorem 22. #GlobalOptimum *is* #P*-hard.*

This result requires discussion. First, it seems difficult to obtain #P-completeness rather than #P-hardness (note that, for #P-completeness, we would need both #P-hardness and membership in #P). For the membership problem, it seems that we would need a polynomial-time algorithm for verifying if a given point is indeed a global optimum. However, this problem is coNP-hard, and so a polynomial-time algorithm is unlikely.

Second, this theorem can be seen as yet another argument that the problem of computing all global optima (or even just counting them) is much more difficult than the problem of finding some local optimum. Even though one might think that #P is "simply a counting variant of NP" and, thus, is of the same "difficulty," it seems that this is not the case. The celebrated Toda's theorem [77, 78] says that the following holds:

$$\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}.$$

This statement means that any problem in the whole of the polynomial hierarchy (which seems to be much bigger than NP and, in particular, includes DP) could be solved in polynomial time, provided that, on each input, one could compute the value of some #P-complete function for one argument. In our context, one could interpret this as follows: if we were effectively able to count global optima, we could solve all NP-complete problems, all coNP-complete problems, all DP-complete problems, and many even more difficult ones. This means that being able to effectively count global optima would give us much greater power than being able to, say, compute global or local optima, and – in effect – counting global optima is a much harder task.

Naturally all of these results apply directly to HGS, and we can claim that either HGS will have a worst-case exponential running time or we cannot guarantee its complete success on every instance. However, we believe that a finer look is necessary. The fact that counting global optima is #P-hard is formally correct, but the proof relies on exploiting a rather unrealistic scenario. Indeed, in typical domains of application of HGS, we can expect only several global optima (or, at best, polynomially many of them), but not necessarily exponentially many (as is the case in the proof). Thus, the complexity of counting them would rather fall into the class #FewP and not #P, which is much easier (#FewP is a counting variant of the FewP class of Allender and Rubinstein [1]). However, analysis of #FewP is more difficult because it is a promise class and, to the best of our knowledge, does not have complete functions.

On the other hand, there is a very different set of complexity classes for local search problems, including the class PLS [28, 41], which models the typical heuristic process of searching for a local optimum. Currently, we do not have such classes to

describe algorithms seeking global optima. Perhaps the model of HGS computation could be used in this way.

5. Conclusions

The goal of this paper is to promote research in a rather understudied area of analyzing the computational cost of complex stochastic strategies for solving parametric inverse problems. This area has two main components: solving global optimization problems, and solving forward problems (to evaluate the misfit function that we try to minimize).

For the first component, we provide a formal description of population-based algorithms that manage single or multiple dependent demes (sub-populations). We pay particular attention to genetic algorithms with heuristics, which can be modeled as ergodic Markov chains. For example, we recall the conditions that may be useful for defining stopping conditions (see Theorems 6 and 7 and Definition 8), as well as provide some formulas that allow one to evaluate the probability of hitting the solution in a single step of such strategies (see Observations 1, 2, and 3), and provide a useful inequality that allows one to associate the probability of hitting a solution with the notion of a convergence along a heuristic (see Observation 4).

We also review the simple method for evaluating the first hitting time for the single-deme algorithm modeled by the ergodic Markov chain (see Section 4.1.3), and we extend it to the case of HGS, a multi-deme hierarchic strategy (see Section 4.2). We focus on the case in which at least the demes in the leaves are well-tuned (see; again, Definition 8).

Finally, we also express the problems of finding local and global optima in terms of a classic complexity theory. We formulate the natural result that finding a local optimum of a function (expressed in a certain natural way) is an NP-complete task, and we argue that finding a global optimum is a much harder, DP-complete, task. Further, we argue that finding all global optima is, possibly, even harder (a #P-hard task). These results provide a reassuring theoretical confirmation of the intuitively clear relation between the hardness of these problems.

Regarding the second component of solving parametric inverse problems (the forward problem solvers), we discuss the computational cost of *hp*-adaptive Finite Element solvers and their rates of convergence with respect to the growing number of degrees of freedom (see Section 3.1).

The presented results provide some useful taxonomy of problems and methods of studying the computational cost and complexity of various strategies for solving inverse parametric problems. Yet, we stress that our goal was not to deliver detailed evaluations for particular algorithms applied to particular inverse problem, but rather to try to identify possible ways of obtaining such results. We plan to extend this research by detailed analysis of particular, important examples in the future.

Acknowledgements

The work presented in this paper has been partially supported by Polish National Science Center grant DEC-2011/03/B/ST6/01393.

References

[1] Allender E., Rubinstein R.: P-Printable Sets. *SIAM Journal on Computing*, vol. 17(6), pp. 1193–1202, 1988.

[2] Babuška I., Guo B.: The *hp*-version of the finite element method, Part I: The basic approximation results. *Computational Mechanics*, vol. 1, pp. 21–41, 1986.

[3] Babuška I., Guo B.: The *hp*-version of the finite element method, Part II: General results and applications. *Computational Mechanics*, vol. 1, pp. 203–220, 1986.

[4] Banks H.T., Kunisch K.: *Estimation Techniques for Distributed Parameter Systems*. Birkhäuser, Boston, 1989.

[5] Barabasz B., Gajda E., Migórski S., Paszyński M., Schaefer R.: Studying inverse problems in elasticity by hierarchic genetic search. In: *ECCOMAS thematic conference on Inverse Problems in Mechanics of Structures and Materials*, pp. 9–10, 2011.

[6] Barabasz B., Gajda-Zagórska E., Migórski S., Paszyński M., Schaefer R., Smołka M.: A hybrid algorithm for solving inverse problems in elasticity. *International Journal of Applied Mathematics and Computer Science*, vol. 24(4), pp. 865–886, 2014.

[7] Barabasz B., Migórski S., Schaefer R., Paszyński M.: Multi-deme, twin adaptive strategy hp-HGS. *Inverse Problems in Science and Engineering*, vol. 19(1), pp. 3–16, 2011.

[8] Beume N., Laumanns M., Rudolph G.: Convergence Rates of (1+1) Evolutionary Multiobjective Optimization Algorithms. In: R. Schaefer, C. Cotta, J. Kołodziej, G. Rudolph, eds., *Parallel Problem Solving from Nature – PPSN XI*, *Lecture Notes in Computer Science*, vol. 6238, pp. 597–606, Springer, 2010.

[9] Boender C., Rinnoy Kan A., Stougie L., Timmer G.: A Stochastic Method for Global Optimization. *Mathematical Programming*, vol. 22, pp. 125–140, 1982.

[10] Burczyński T., Beluch W.: The Identification of Cracks Using Boundary Elements and Evolutionary Algorithms. *Engineering Analysis with Boundary Elements*, vol. 25(4–5), pp. 313–322, 2001.

[11] Burczyński T., Kuś W., Długosz A., Orantek P.: Optimization and defect identification using distributed evolutionary algorithms. *Engineering Applications of Artificial Intelligence*, vol. 17(4), pp. 337–344, 2004.

[12] Cabib E., Davini C., Chong-Quing R.: A problem in the optimal design of networks under transverse loading. *Quarterly of Applied Mathematics*, vol. 48(2), pp. 251–263, 1990.

[13] Caicedo J.M., Yun G.: A novel evolutionary algorithm for identifying multiple alternative solutions in model updating. *Structural Health Monitoring*, vol. 10, pp. 491–501, 2011.

[14] Ciarlet P.G.: *The Finite Element Method for Elliptic Problems*. North-Holland, 1978.

[15] Cook S.: The complexity of theorem-proving procedures. In: *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pp. 151–158, ACM Press, 1971.

[16] Demkowicz L.: *Computing with hp-Adaptive Finite Elements, Vol. I. One and Two Dimensional Elliptic and Maxwell Problems*. Chapman and Hall/CRC Applied Mathematics and Nonlinear Science, 2006.

[17] Demkowicz L., Kurtz J., Pardo P., Paszyński M., Rachowicz W., Zdunek A.: *Computing with hp-Adaptive Finite Elements, Vol. II. Frontiers: Three-Dimensional Elliptic and Maxwell Problems with Applications*. Chapman and Hall/CRC Applied Mathematics and Nonlinear Science, 2007.

[18] Denkowski Z., Migórski S., Papageorgiou N.: *An Introduction to Nonlinear Analysis: Applications*. Kluwer Academic/Plenum, 2003.

[19] Denkowski Z., Migórski S., Papageorgiou N.: *An Introduction to Nonlinear Analysis: Theory*. Kluwer Academic/Plenum, 2003.

[20] Descloux J.: *Méthode Des Éléments Finis*. Ecole Polytechnique Fédérale de Lausanne, Lausanne, 1973.

[21] Doerr B., Jansen T., Sudholt D., Winzen C., Zarges C.: Optimizing Monotone Functions Can Be Difficult. In: R. Schaefer, C. Cotta, J. Kołodziej, G. Rudolph, eds., *Parallel Problem Solving from Nature – PPSN XI*, *Lecture Notes in Computer Science*, vol. 6238, pp. 42–51, Springer, 2010.

[22] Engl H., Hanke M., Neubauer A.: *Regularization of Inverse Problems*, *Mathematics and its Applications*, vol. 375. Springer-Verlag, Berlin Heidelberg, 1996.

[23] Faliszewski P., Ogihara M.: On the Autoreducibility of Functions. *Theory of Computing Systems*, vol. 46(2), pp. 222–245, 2010.

[24] Gajda-Zagórska E., Schaefer R., Smołka M., Paszyński M., Pardo D.: A hybrid method for inversion of 3D DC logging measurements. *Natural Computing*, vol. 14(3), pp. 355–374, 2015.

[25] Glover F., Kochenberger G.: *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2002.

[26] Hemaspaandra L., Ogihara M.: *The Complexity Theory Companion*. Springer-Verlag, 2002.

[27] Isakov V.: *Inverse Problems for Partial Differential Equations*. Springer, 2006.

[28] Johnson D., Papadimitriou C., Yannakakis M.: How Easy is Local Search. *Journal of Computer and System Sciences*, vol. 37(1), pp. 79–100, 1988.

[29] Kołodziej J., Jakubiec W., Starczak M., Schaefer R.: Identification of the CMM Parametric Errors by Hierarchical Genetic Strategy. In: *IUTAM Symposium on Evolutionary Methods in Mechanics*, pp. 187–196, Springer, 2004.

[30] Kołodziej J., Schaefer R., Paszyńska A.: Hierarchical genetic computation in optimal design. *Journal of Theoretical and Applied Mechanics, Computational Intelligence*, vol. 42(3), pp. 519–539, 2004.

[31] Koper K., Wysession M., Wiens D.: Multimodal function optimization with a niching genetic algorithm: A seismological example. *Bulletin of the Seismological Society of America*, vol. 89(4), pp. 978–988, 1999.

[32] Ladner R., Lynch N., Selman A.: A Comparison of Polynomial Time Reducibilities. *Theoretical Computer Science*, vol. 1(2), pp. 103–124, 1975.

[33] Lässig J., Sudholt D.: Experimental Supplements to the Theoretical Analysis of Migration in the Island Model. In: R. Schaefer, C. Cotta, J. Kołodziej, G. Rudolph, eds., *Parallel Problem Solving from Nature – PPSN XI*, *Lecture Notes in Computer Science*, vol. 6238, pp. 224–233, Springer, 2010.

[34] Lässig J., Sudholt D.: General Scheme for Analyzing Running Times of Parallel Evolutionary Algorithms. In: R. Schaefer, C. Cotta, J. Kołodziej, G. Rudolph, eds., *Parallel Problem Solving from Nature – PPSN XI*, *Lecture Notes in Computer Science*, vol. 6238, pp. 234–243, Springer, 2010.

[35] Mahfoud S.W.: Niching Methods. In: T. Back, D.B. Fogel, Z. Michalewicz, eds., *Handbook of Evolutionary Computations*, chap. C6.1, pp. C6.1:1–C6.1:4, IOP Publishing and Oxford University Press, 1997.

[36] Meruane V., Heylen W.: Damage Detection with Parallel Genetic Algorithms and Operational Modes. *Structural Health Monitoring*, vol. 9, pp. 481–496, 2009.

[37] Neri F., Cotta C., Moscato P., eds.: *Handbook of Memetic Algorithms*, *Studies in Computational Intelligence*, vol. 379. Springer-Verlag, Berlin Heidelberg, 2012.

[38] Norris J.R.: *Markov Chains*. Cambridge University Press, Cambridge, 1997.

[39] Osman I., Kelly J.: *Meta-Heuristics: Theory and Applications*. Kluwer Academic Publishers, 1996.

[40] Papadimitriou C.: *Computational Complexity*. Addison-Wesley, 1994.

[41] Papadimitriou C., Schäffer A., Yannakakis M.: On the complexity of local search. In: *Proceedings of the 22nd ACM Symposium on Theory of Computing*, pp. 84–94, ACM Press, 1990.

[42] Papadimitriou C., Yannakakis M.: The Complexity of Facets (and some Facets of Complexity). *Journal of Computer and System Sciences*, vol. 28(2), pp. 244–259, 1984.

[43] Pardalos P., Romeijn H.: *Handbook of Global Optimization (Nonconvex Optimization and its Applications)*, vol. 2. Kluwer, 1995.

[44] Paszyńska A., Grabska E., Paszyński M.: A Graph Grammar Model of the hp Adaptive Three Dimensional Finite Element Method. Part I. *Fundamenta Informaticae*, vol. 114(2), pp. 149–182, 2012.

[45] Paszyńska A., Grabska E., Paszyński M.: A Graph Grammar Model of the hp Adaptive Three Dimensional Finite Element Method. Part II. *Fundamenta Informaticae*, vol. 114(2), pp. 183–201, 2012.

[46] Paszyńska A., Paszyński M., Grabska E.: Graph Transformations for Modeling

hp-Adaptive Finite Element Method with Triangular Elements. *Lecture Notes in Computer Science*, vol. 5103, pp. 604–613, 2008.

[47] Paszyńska A., Paszyński M., Grabska E.: Graph Transformations for Modeling hp-Adaptive Finite Element Method with Mixed Triangular and Rectangular Elements. *Lecture Notes in Computer Science*, vol. 5545, pp. 875–884, 2009.

[48] Paszyński M.: On the Parallelization of Self-Adaptive hp-Finite Element Methods: Part I: Composite Programmable Graph Grammar Model. *Fundamenta Informaticae*, vol. 93(4), pp. 411–434, 2009.

[49] Paszyński M.: On the Parallelization of Self-Adaptive hp-Finite Element Methods: Part II: Partitioning Communication Agglomeration Mapping (PCAM) Analysis. *Fundamenta Informaticae*, vol. 93(4), pp. 435–457, 2009.

[50] Paszyński M., Barabasz B., Schaefer R.: Efficient adaptive strategy for solving inverse problems. In: *Computational Science – ICCS 2007*, vol. 4487, pp. 342–349, Springer, 2007.

[51] Paszyński M., Demkowicz L.: Parallel Fully Automatic hp-Adaptive 3D Finite Element Package. *Engineering with Computers*, vol. 22(3–4), pp. 255–276, 2006.

[52] Paszyński M., Schaefer R.: Graph grammar-driven parallel partial differential equation solver. *Concurrency and Computation: Practice and Experience*, vol. 22(9), pp. 1063–1097, 2010.

[53] Rachowicz W., Pardo D., Demkowicz L.: Fully Automatic hp-Adaptivity in Three Dimensions. *Computer Methods in Applied Mechanics and Engineering (J.H. Argyris Memorial Issue)*, vol. 37–40, pp. 4816–4842, 1995.

[54] Rudolph G.: Takeover Time in Parallel Populations with Migration. In: *Proceedings of the Second International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2006)*, pp. 63–72, Josef Stefan Institute, Ljubljana, 2006.

[55] Ryszka I., Paszyńska A., Grabska E., Sieniek M., Paszyński M.: Graph Transformation Systems for Modeling Three Dimensional Finite Element Method: Part I. *Fundamenta Informaticae*, vol. 140(2), pp. 129–172, 2015.

[56] Schaefer R.: The role of heuristics in serial and parallel genetic search. In: *Abstract Book of the 3rd Conference on Numerical Analysis, Krynica, Poland*, pp. 16–17, 2002, ISBN 978-3-642-15843-8.

[57] Schaefer R.: *Foundation of Genetic Global Optimization, with Chapter 6 by Telega H.*, *Studies in Computational Intelligence Series*, vol. 74. Springer, 2007.

[58] Schaefer R., Barabasz B.: Asymptotic Behavior of hp-HGS (hp-Adaptive Finite Element Method Coupled with the Hierarchic Genetic Strategy) by Solving Inverse Problems. In: *Computational Science — ICCS 2008*, *Lecture Notes in Computer Science*, vol. 5103, pp. 682–691, Springer, 2008.

[59] Schaefer R., Byrski A., Kołodziej J., Smołka M.: An agent-based model of hierarchic genetic search. *Computers and Mathematics with Applications*, vol. 64(12), pp. 3763–3776, 2012.

[60] Schaefer R., Byrski A., Smołka M.: Island Model as Markov Dynamic System.

International Journal of Applied Mathematics and Computer Science, vol. 22(4), pp. 971–984, 2012.

[61] Schaefer R., Jabłoński Z.: On the convergence of sampling measures in the global genetic search. In: *Parallel Processing and Applied Mathematics – PPAM IV*, *Lecture Notes in Computer Science*, vol. 2328, pp. 593–600, Springer, 2002.

[62] Schaefer R., Kołodziej J.: Genetic search reinforced by the population hierarchy. In: K. De Jong, R. Poli, J. Rowe, eds., *Foundations of Genetic Algorithms 7*, pp. 383–399, Morgan Kaufman, 2003.

[63] Schauder J.: Der Fixpunktsatz in Funktionalräumen. *Studia Mathematica*, vol. 2, pp. 171–180, 1930.

[64] Scholz D.: *Deterministic Global Optimization. Geometric Branch-and-bound Methods and their Applications*. Springer Optimization and Application Series 63, Springer, 2013.

[65] Schraudolph N., Belew R.: Dynamic parameter encoding for genetic algorithms. *Machine Learning Journal*, vol. 9(1), pp. 9–21, 1992.

[66] Schwab C.: *P and hp Finite Element Methods*. Oxford University Press, 1998.

[67] Simon J.: *On Some Central Problems in Computational Complexity*. Ph.D. thesis, Cornell University, Ithaca, N.Y., 1975, available as Cornell Department of Computer Science Technical Report TR75-224.

[68] Skolicki Z., Jong K.D.: Improving Evolutionary Algorithms with Multi-representation Island Models. In: *Proceedings of 8th International Conference on Parallel Problem Solving from Nature — PPSN VIII*, vol. 3242, pp. 420–429, Springer, 2004.

[69] Skolicki Z., Jong K.D.: The Influence of Migration Sizes and Intervals on Island Models. In: *Proceedings of Genetic and Evolutionary Computation Conference — GECCO-2005*, pp. 1295–1302, ACM Press, 2005.

[70] Smołka M., Gajda-Zagórska E., Schaefer R., Paszyński M., Pardo D.: A Hybrid Method for Inversion of 3D AC Resistivity Logging Measurements. *Applied Soft Computing*, vol. 36, pp. 442–456, 2015.

[71] Stockmeyer L.: The Polynomial-Time Hierarchy. *Theoretical Computer Science*, vol. 3(1), pp. 1–22, 1976.

[72] Strug B., Paszyńska A., Paszyński M., Grabska E.: Using a Graph Grammar System in the Finite Element Method. *Applied Mathematics and Computer Science*, vol. 23(4), pp. 839–853, 2013.

[73] Sudholt D.: General Lower Bounds for the Running Time of Evolutionary Algorithms. In: R. Schaefer, C. Cotta, J. Kołodziej, G. Rudolph, eds., *Parallel Problem Solving from Nature – PPSN XI*, *Lecture Notes in Computer Science*, vol. 6238, pp. 124–133, Springer, 2010.

[74] Tarantola A.: *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, 2005.

[75] Telega H.: *Parallel Algorithms for Solving Selected Inverse Problems*. PhD Thesis, AGH University of Science and Technology, 1999.

[76] Telega H., Schaefer R., Cabib E.: A Parallel Genetic Clustering for Inverse Problems. In: *Applied Parallel Computing: Large Scale Scientific and Industrial Problems*, *Lecture Notes in Computer Science*, vol. 1541, pp. 551–556, Springer, 1998.

[77] Toda S.: PP Is as Hard as the Polynomial-Time Hierarchy. *SIAM Journal on Computing*, vol. 20(5), pp. 865–877, 1991.

[78] Toda S., Ogiwara M.: Counting Classes are at Least as Hard as the Polynomial-Time Hierarchy. *SIAM Journal on Computing*, vol. 21(2), pp. 316–328, 1992.

[79] Vose M.: *The Simple Genetic Algorithm*. MIT Press, 1999.

[80] Whitley D., Gordon V.: Serial and Parallel Genetic Algorithms as Function Optimizers. In: S. Forrest, ed., *Proceedings of ICGA'97*, pp. 177–218, Morgan Kaufman, 1993.

[81] Whitley D., Mathias K., Fitzhorn P.: Delta Coding: An Iterative search Strategy. In: R. Belew, L. Booker, eds., *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 77–84, Morgan Kaufman, 1991.

[82] Whitley D., Soraya S., Heckerdorn R.: Island Model Genetic Algorithms. In: *Proceedings of AISB'97 Workshop on Evolutionary Computing*, pp. 112–129, 1997.

[83] Wierzba B., Semczuk A., Kołodziej J., Schaefer R.: Hierarchical Genetic Strategy with Real Number Encoding. In: *Proceedings of the 6th Conference on Evolutionary Algorithms and Global Optimization*, pp. 231–237, 2003.

Affiliations

Piotr Faliszewski
AGH University of Science and Technology, Krakow, Poland, faliszew@agh.edu.pl

Maciej Smołka
AGH University of Science and Technology, Krakow, Poland, smolka@agh.edu.pl

Robert Schaefer
AGH University of Science and Technology, Krakow, Poland, schaefer@agh.edu.pl

Maciej Paszyński
AGH University of Science and Technology, Krakow, Poland, paszynsk@agh.edu.pl

Received: 20.05.2015
Revised: 24.11.2015
Accepted: 24.11.2015