Volume 12, Number 1, 2019. pp. 17-36

Jordan Journal of Physics

ARTICLE

A Genetic Algorithm for Addressing Computationally Expensive Optimization Problems in Optical Engineering

A. Mayer [a] and **Michaël Lobet** [a,b]

[a] *Department of Physics, University of Namur, Rue de Bruxelles 61, 5000 Namur, Belgium.*

[b] *John A. Paulson School of Engineering and Applied Sciences, Harvard University, 9 Oxford Street, 02138 Cambridge, MA, USA.*

Received on: 2/7/2018; Accepted on: 14/11/2018

Abstract: We present a genetic algorithm that we developed in order to address computationally expensive optimization problems in optical engineering. The idea consists of working with a population of individuals representing possible solutions to the problem. The best individuals are selected. They generate new individuals for the next generation. Random mutations in the coding of parameters are introduced. This strategy is repeated from generation to generation until the algorithm converges to the global optimum of the problem considered. For computationally expensive problems, one can analyze the data collected by the algorithm in order to infer more rapidly the final solution. The use of a mutation operator that acts on randomly-shifted Gray codes helps the genetic algorithm escape local optima and enables a wider diversity of displacements. These techniques reduce the computational cost of optical engineering problems, where the design parameters have a finite resolution and are limited to a realistic range. We demonstrate the performance of this algorithm by considering a set of 22 benchmark problems in 5, 10 and 20 dimensions that reflect the conditions of these engineering problems. We finally show how these techniques accelerate the determination of optimal structures for the broadband absorption of electromagnetic radiations.

Keywords: Genetic Algorithm, Optical Engineering, Optimization, Quadratic Approximation, Gray Codes, Metamaterials.

PACS: 02.60.Pn, 42.15.Eq, 42.25.Bs, 78.67.Pt.

Introduction

The design of optical devices requires at some point the search for optimal parameters in order to achieve maximal performances. With genetic algorithms (GAs), natural selection is mimicked in order to determine this set of optimal parameters. The idea consists of working with a virtual population of individuals representing possible solutions to the problem. The initial population consists of random individuals. The best individuals are then selected. They generate new individuals for the next generation. Random mutations in the coding of parameters are finally introduced. When repeated from generation to generation, this strategy enables the determination of a globally optimal set of parameters [1-6].

Optical engineering problems are typically computationally expensive due to the numerous degrees of freedom and the CPU time involved by the numerical modeling. It is therefore desirable to solve the optimization problem ideally by a single run of the GA and with a reduced number of fitness evaluations. The fitness is defined as the objective function to be optimized. When the time required by

Corresponding Author: Alexandre Mayer **Email**: alexandre.mayer@unamur.be

the fitness evaluations is largely superior to the time required by the GA itself, it makes sense to establish a record with all fitness evaluations in order to avoid any duplication of these calculations. The GA also gains at being organized in a way that enables all fitness calculations in a given generation to be addressed at the same time. This allows indeed a massive parallelization of these calculations on modern supercalculators. The genetic algorithm finally gains at being combined with a mathematical analysis of the collected data in order to accelerate convergence to the final solution. The objective is to determine the global optimum as quickly as possible (reduced number of generations) and with a reduced number of fitness evaluations.

One can guide the algorithm to promising directions and accelerate the refinement of the final solution by coupling the genetic algorithm with a local optimizer (memetic algorithm) [7-13]. A first approach consists of applying a local optimization procedure on the solutions established by the genetic algorithm, either regularly (starting from best-so-far solutions established at each generation by the GA) or after the GA has converged (starting from the final best solutions established by the GA) [5, 8]. This approach requires however an extra budget of fitness evaluations. Another approach consists of working on the data already collected by the genetic algorithm in order to avoid an increase in the number of fitness evaluations. An idea consists of establishing different approximations of the fitness (reduced models) in order to implement this local optimization [14-19], improve the genetic operators [20, 21], estimate the robustness of solutions [22] or avoid unnecessary evaluations of the fitness [23-25]. The data collected by the GA can actually be analyzed by a variety of mathematical methods. Methods based on the Singular Values Decomposition were used to estimate the evolution direction and increase the population diversity [26]. This technique was also used to qualify potential candidates for the next generation [27]. Recent papers finally considered training neural networks in order to guide the genetic algorithm [20, 28-31]. A neural network is then trained on the data collected by the GA in order to establish reduced models of the fitness and suggest promising solutions.

In optical engineering problems, the physical parameters to determine have a finite resolution due to physical or experimental limitations in the fabrication of a device [32-37]. The decision variables have therefore a finite number of possible values (typically of the order of 1000). A binary encoding of these decision variables offers the advantage to account for this discrete set of possible values at all stages of the algorithm. Optical engineering problems that rely on numerical simulations for the evaluation of the fitness have also as specificity the fact that the fitness is generally accurate to only three or four significant digits. Optimizing the fitness beyond this limited accuracy does not make any sense. The genetic algorithm on the contrary gains at being tuned to achieve a target accuracy that is both realistic and appropriate for these applications (typically $\Delta f_{target} \sim 10^{-4}$).

We present in this article an algorithm that we developed in order to account for these different issues when addressing optical engineering problems. Our approach consists of establishing at each generation a quadratic approximation of the fitness in the close neighborhood of the best-so-far individual in order to infer more rapidly the global optimum. We also consider randomly-shifted Gray codes when applying mutations in order to improve exploration and escape local optima. These modifications of the well-known genetic algorithm reduce the computational cost of optical engineering problems, where the design parameters have a finite resolution and are limited to a realistic range. This article is organized as follows. The main lines of our algorithm are presented in the next section. Then, we apply our algorithm to typical benchmark problems in 5, 10 and 20 dimensions in order to demonstrate its performance. Then, we provide a real optical engineering application. Finally, we conclude this article.

Description of the Genetic Algorithm

The genetic algorithm described in this section aims at determining the global optimum (depending on the application, it will be the global minimum or the global maximum) of an objective function $f =$

$f(x_1, \dots x_n)$, where n is the number of decision variables. $x_i \in [x_i^{\min}, x_i^{\max}]$, with a discretization step Δx_i. The boundaries $x_i^{\min}$ and $x_i^{\max}$ must be specified at the beginning of the search. Δx_i accounts for the experimental resolution of each decision variable. The variables x_i are represented by sequences of binary digits (genes). We use the Gray code to interpret the bit content of these genes [5, 38]. The decision variables are then given by $x_i = x_i^{\min} + \langle \text{gene } i \rangle \times \Delta_i$,where $\langle \text{gene } i \rangle \in [0, 2^{n_i} - 1]$ refers to the value of the gene. The bit length n_i of each gene is the first integer for which $x_i^{\min} + (2^{n_i} - 1) \times \Delta_i \geq x_i^{\max}$. $n_{bits} = \sum_{i=1}^{n} n_i$ refers to the total number of bits in a DNA; i.e., the set of genes used for coding the n decision variables.

A detailed pseudocode of our algorithm can be found in Appendix A. We present here only the main ideas of this algorithm, which are as follows: We consider a population of n_{pop} = 50 individuals. We start with a random population. We evaluate the fitness $f(x_1, \dots x_n)$ of each individual and sort the population from the best individual to the worst one. We save the computed $\{\vec{x}, f(\vec{x})\}$ data in a record. We compute the genetic similarity s of the population; s corresponds to the fraction of bits in the population whose value is identical to the best individual [32, 38]. We then define a progress indicator $p = |s - 0.5|/0.5$, which takes values between 0 and 1. The worst n_{rand} = even $[0.1 \times n_{\text{pop}} \times (1\text{-}p)]$ individuals of the population are then replaced by random individuals (even [.] stands for the nearest even integer). These random individuals are transferred to the next generation. The remaining $N = n_{\text{pop}} - n_{\text{rand}}$ individuals of the current population participate to the usual steps of selection, crossover and mutation. We hence select N parents in this subset of N individuals by a rank-based roulette wheel selection, noting that a given individual can be selected several times [5, 32]. For any pair of parents, we define two children for the next generation either (i) by a one-point crossover of the parents' DNA (probability of 70%) or (ii) by a simple replication of the parents. The children obtained by crossover are subjected to a modified mutation operator that acts on randomly-shifted Gray codes (see Appendix B), using $m = 0.95/n_{\text{bits}}$ as mutation rate for individual bit flips. We apply at this point a local optimization procedure on the $\{\vec{x}, f(\vec{x})\}$ data collected so far by the genetic algorithm in order to guess the final solution (see Appendix C). If the result of this local optimization can be accepted, it replaces the last individual already scheduled for the next generation (a random individual if $n_{\text{rand}} > 0$). Before evaluating the fitness of the individuals finally scheduled for the next generation, we check the records in order to avoid any duplication of these evaluations. We then evaluate the fitness of the individuals scheduled for the next generation for which no $\{\vec{x}, f(\vec{x})\}$ data was found. We sort the new population and apply elitism in order to make sure that the best solution achieved so far is not lost when going from one generation to the next [5]. We apply these different steps from generation to generation until a termination criterion is met.

The organization of the algorithm ensures that all fitness calculations in a given generation can be evaluated in parallel, since there is only one round of fitness evaluations per generation. In this implementation, the parents are not transferred automatically to the next generation, since this leads to premature convergence to solutions that are not globally optimal. We found in previous, unpublished work that a crossover rate of 70% maintains a good balance between the conservation of good solutions (individuals transferred to the next generation without any modification) and the exploration of new solutions (individuals modified by the operators of crossover and mutation). The mutation rate $m = 0.95/n_{\text{bits}}$ is settled automatically by the number of bits used for the representation of the decision variables. We found in previous work that the optimal mutation rate decreases with the dimension of the problem. Maintaining $m \times n_{\text{bits}} < 1$ is also motivated by biological evidence [39]. This condition ensures indeed that the best individuals in the population have a chance to be unaffected by mutations. We confirmed empirically that this improves in the long term the quality of the solutions established by the genetic algorithm. The use of a mutation operator that acts on randomly-shifted Gray codes helps the genetic algorithm escape local optima, since the displacements generated by this mutation operator have a wider diversity (see Appendix B). This improves also the exploration of the decision variable space. The local optimization procedure finally provides a useful guidance to the genetic algorithm by indicating, generation after generation, directions to consider based on collected data. The technical parameters of this algorithm were tuned on test problems in 5, 10 and 20 dimensions, for conditions that reflect those encountered in optical engineering problems [40]. We demonstrate the performance of this algorithm on an extended set of test problems in the next section.

Application to Test Problems in 5, 10 and 20 Dimensions

In optical engineering problems that stimulated this work [32-34], the decision variables x_i must be determined only up to a precision Δx_i due to experimental limitations in the fabrication of a device. We will therefore consider in this section test problems for which $\Delta x_i = (x_i^{\max} - x_i^{\min})/4096$ in order to reflect the conditions of these applications. This corresponds to n_i=12 bits per gene, since $2^{12} = 4096$. We will also consider that the global minimum of the test problems considered in this section is found if the objective function is within a margin Δf_{target} of 10^{-4} compared to the exact solution. This reflects again the accuracy with which solutions should be established in these optical engineering applications. Our objective was to determine the global minimum of this type of problems with a high chance of success in one run and with a reduced number of fitness evaluations (since we accept a margin Δf_{target} on the global minimum, technically we actually seek determining a "global Δf_{target}-optimal solution". Since the algorithm is stochastic, there is of course no guarantee on optimality).

The 22 benchmark functions considered in this work are given in Table 1. The boundaries $[x_i^{\min}, x_i^{\max}]$ considered for each function are provided as well as the number of bits n_i used for the representation of each decision variable (n_i=12, except for Schwefel 7, where n_i=16) [41]. With this setting of the experiment, all gene values can be accepted and there is a point in the grid for which the target Δf_{target} of 10^{-4} can actually be reached. In order to make sure that our results do not depend on a specific encoding of the decision variables and in order to break easy symmetries, we consider for each instance of the genetic algorithm a random shift of the domain $[x_i^{\min}, x_i^{\max}]$ considered for each decision variable. This randomization of the boundaries is limited to integer multiples of $\Delta x_i = (x_i^{\max} - x_i^{\min})/2^{n_i}$ in order to make sure that the point for which the target Δf_{target} of 10^{-4} can actually be reached remains on the grid. The limits considered for this randomization of the boundaries are given in the fourth column of Table 1.

When running the genetic algorithm on a given function $f(\vec{x})$ in order to determine its global minimum, we consider that the target Δf_{target} is reached if $|f(\vec{x}_{\text{best}}) - f_{\text{opt}}^*)| \leq \Delta f_{\text{target}}$, where $f(\vec{x}_{\text{best}})$ is the best-so-far solution found by the genetic algorithm and f_{opt}^* the exact global minimum. By running the genetic algorithm #run times on each test function, we can measure the probability $P(\Delta f_{\text{target}})$ with which the target Δf_{target} is reached by a given run of the algorithm. This quantity is calculated by $P(\Delta f_{\text{target}})$=#success/#run, where #success refers to the number of successful runs. We can also measure the average number of fitness evaluations required to reach Δf_{target}. This quantity is calculated by $\langle n_{\text{eval}} \rangle$=#eval(target not reached)/#success, where #eval(target not reached) is the number of fitness evaluations in all generations for which the target Δf_{target} was not reached (summing over the #run executions of the GA) [11]. $\langle n_{\text{eval}}\rangle$ includes fitness evaluations in runs that failed to meet the target. Accounting for failed attempts makes sense, since they must be paid in real-world applications. They consume indeed CPU time and cause a delay in the resolution of a problem. Our efforts to tune the genetic algorithm therefore focusses on $\langle n_{\text{eval}}\rangle$ as a measure for the computational cost associated with a given target Δf_{target}. Another measure commonly used in the literature is $\langle n^*_{\text{eval}}\rangle$, the average number of fitness evaluations required to reach Δf_{target} when this target is actually reached. $\langle n^*_{\text{eval}}\rangle$ does not account for failed attempts. Similarly, $\langle n^*_{\text{gen}}\rangle$ measures the average number of generations required to reach a Δf_{target} for runs that actually reach this target. $\langle n^*_{\text{gen}}\rangle$ is representative of how fast a solution is found, if found.

TABLE 1. List of test functions with the boundaries $[x_i^{\min}, x_i^{\max}]$ considered for the decision variables and the number of bits n_i used for the representation of each gene. The fourth column indicates the limits considered for the randomization of the boundaries when running a given instance of the genetic algorithm. Names: Sphere (#1), Rotated Hyper-Ellipsoid (#2), Rosenbrock (#3), Modified Dixon-Price (#4), Mayer (#5), Schwefel 7 (#6), Levy (#7), Rastrigin (#8), Ackley (#9), Griewank (#10), Cosine Mixture (#11), Exponential (#12), Levy and Montalvo 1 (#13), Levy and Montalvo 2 (#14), Zakharov (#15), Schwefel 3 (#16), Brown 3 (#17), Cigar (#18), Sinusoidal (#19), Trigonometric 1 (#20), Pinter (#21) and Whitley (#22).

#	Formula	$[x_i^{\min}, x_i^{\max}]$	rand shift	n_i
1	$f(\vec{x}) = \sum_{i=1}^n x_i^2$	$[-5.12, 5.12]$	$[-0.5, 0.5]$	12
2	$f(\vec{x}) = \sum_{i=1}^n (\sum_{j=1}^i x_j)^2$	$[-65.5, 65.5]$	[-5,5]	12
3	$f(\vec{x}) = \sum_{i=1}^{n-1} [100\ (x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	$[-2, 2]$	[-0.2,0.2]	12
4	$f(\vec{x}) = n(x_1 - 1)^2 + \sum_{i=2}^n (2x_i^2 - x_{i-1})^2$	$[0, 10.24]$	[0,0.25]	12
5	$f(\vec{x}) = -\Pi_{i=1}^n \cos(x_i)^2 \exp(-x_i^2/10)$	$[-5, 5]$	[-0.5,0.5]	12
6	$f(\vec{x}) = 418.98288727243 \times n - \sum_{i=1}^n x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]$	[-5,10]	16
7	$f(\vec{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{n-1} (w_i - 1)^2 [1 + 10 \sin^2(\pi w_i + 1)]$ $+ (w_n - 1)^2 [1 + \sin^2(2\pi w_n)],\ w_i = 1 + (x_i - 1)/4$	$[-10.24, 10.24]$	[-1,1]	12
8	$f(\vec{x}) = 10\ n + \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i))$	$[-5.12, 5.12]$	[-0.5,0.5]	12
9	$f(\vec{x}) = -a \exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^n \cos(cx_i)\right)$ $+a + e,\ a = 20, b = 0.2, c = 2\pi$	$[-32, 32]$	[-3,3]	12
10	$f(\vec{x}) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos(x_i/\sqrt{i})$	$[-600, 600]$	[-50,50]	12
11	$f(\vec{x}) = 0.1 \times n + \sum_{i=1}^n x_i^2 - 0.1 \sum_{i=1}^n \cos(5\pi x_i)$	$[-1, 1]$	[-0.1,0.1]	12
12	$f(\vec{x}) = 1 - \exp\left(-0.5 \sum_{i=1}^n x_i^2\right)$	$[-1, 1]$	[-0.1,0.1]	12
13	$f(\vec{x}) = \frac{\pi}{n}\left(10 \sin^2(\pi w_1) + \sum_{i=1}^{n-1} (w_i - 1)^2 [1 + 10 \sin^2(\pi w_{i+1})]\right.$ $\left.+ (w_n - 1)^2\right),\ w_i = 1 + \frac{x_i + 1}{4}$	$[-10.24, 10.14]$	[-1,1]	12
14	$f(\vec{x}) = 0.1 \left(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})]\right.$ $\left.+ (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\right)$	$[-5.12, 5.12]$	[-0.5,0.5]	12
15	$f(\vec{x}) = \sum_{i=1}^n x_i^2 + \left(\frac{1}{2}\sum_{i=1}^n ix_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^n ix_i\right)^4$	$[-5.12, 5.12]$	[-0.5,0.5]	12
16	$f(\vec{x}) = \sum_{i=1}^n \lvert x_i \rvert + \prod_{i=1}^n \lvert x_i \rvert$	$[-10, 10]$	[-1,1]	12
17	$f(\vec{x}) = \sum_{i=1}^{n-1} \left((x_i^2)^{x_{i+1}^2 + 1} + (x_{i+1}^2)^{x_i^2 + 1}\right)$	$[-1, 4]$	[-0.1,0.4]	12
18	$f(\vec{x}) = x_1^2 + 100,000 \sum_{i=2}^n x_i^2$	$[-10, 10]$	[-1,1]	12
19	$f(\vec{x}) = 3.5 - 2.5 \prod_{i=1}^n \sin\left(x_i - \frac{\pi}{6}\right) - \prod_{i=1}^n \sin\left(5(x_i - \frac{\pi}{6})\right)$	$[0, 3.1415]$	[-0.1,0.2]	12
20	$f(\vec{x}) = \sum_{i=1}^n \left[n - \sum_{j=1}^n \cos x_j + i(1 - \cos x_i - \sin x_i)\right]^2$	$[0, 3.1415]$	[-0.3,0]	12
21	$f(\vec{x}) = \sum_{i=1}^n ix_i^2 + \sum_{i=1}^n 20i \sin^2 A + \sum_{i=1}^n i \log_{10}\left(1 + iB^2\right),$ $A = x_{i-1} \sin x_i + \sin x_{i+1},\ x_0 = x_n,\ x_{n+1} = x_1,$ $B = x_{i-1}^2 - 2x_i + 3x_{i+1} - \cos x_i + 1$	$[-10, 10]$	[-1,1]	12
22	$f(\vec{x}) = \sum_{i=1}^n \sum_{j=1}^n \left[\frac{(100(x_i^2 - x_j)^2 + (1 - x_j)^2)^2}{4000}\right.$ $\left.- \cos(100(x_i^2 - x_j)^2 + (1 - x_j)^2) + 1\right]$	$[-10.24, 10.24]$	[-1,1]	12

The results obtained with our algorithm, when considering the benchmark problems of Table 1 for n = 5, 10 and 20 dimensions, are summarized in Table 2. Tables 3, 4 and 5 provide the $P(\Delta f_{\text{target}})$, $\langle n_{\text{eval}} \rangle$, $\langle n^*_{\text{eval}} \rangle$ and $\langle n^*_{\text{gen}} \rangle$ values obtained for individual functions when considering a target Δf_{target} of 10^{-4}. For these benchmark problems, we consider a maximum of $30\times n_{\text{bits}}$ generations for a given run of the algorithm. The algorithm is interrupted if (i) there is no improvement of the best fitness in the last $1.5\times n_{\text{bits}}$ generations, (ii) the mean value of the genetic similarity s over the last $1.5\times n_{\text{bits}}$ generations is higher than $1\text{-}3m$, (iii) $s\geq 1\text{-}m$ or (iv) the number of fitness evaluations exceeds $10000\times n$. The different columns of Table 2 show the results obtained when considering/not considering (i) local optimizations based on quadratic approximations of the fitness and (ii) a mutation operator that acts on randomly-shifted Gray codes. The table provides the probability of success in one run $P(\Delta f_{\text{target}})$ and the average number of fitness evaluations $\langle n_{\text{eval}} \rangle$ for different values of Δf_{target}. It also specifies the number of functions for which the target was reached at least once in ten runs. This comparison between different versions of our algorithm proves the advantage of using a mutation operator that acts on randomly-shifted Gray codes and a local optimization procedure that works on the data collected by the algorithm (see Appendix B and Appendix C).

By using the local optimization procedure and a mutation operator that acts on randomly-shifted Gray codes, we achieve a probability of success in one run $P(\Delta f_{\text{target}})$ of 94.9% for n=5 dimensions, 92.3% for n=10 dimensions and 89.0% for n=20 dimensions when considering a target Δf_{target} of 10^{-4} (these values correspond to an average over the 22 benchmark problems; the values obtained for individual functions can be found in Tables 3, 4 and 5). The average number of fitness evaluations $\langle n_{\text{eval}} \rangle$ required to reach this target is 1724 for n=5 dimensions, 5104 for n=10 dimensions and 19870 for n=20 dimensions. This corresponds to $\langle n_{\text{eval}} \rangle / n$ ratios of 345 for n =5 dimensions, 510 for n =10 dimensions and 993 for n =20 dimensions. We meet therefore our objective to determine the global minimum of these test problems with a high probability of success in one run ($P(\Delta f_{\text{target}})$=89-95%), while keeping to a budget of fitness evaluations $\langle n_{\text{eval}} \rangle$ of the order of $\sim 1000 \times n$. In contrast, when the techniques presented in the two Appendices are not used, the probability of success in one run $P(\Delta f_{\text{target}})$ is reduced to 75.6% for n=5 dimensions, 62.5% for n=10 dimensions and 46.7% for n=20 dimensions. The number of functions for which the global minimum is determined at least once in ten runs decreases rapidly with the dimension of the problem, going from 18 functions out of 22 for problems in 5 dimensions to only 15 functions out of 22 for problems in 20 dimensions. The average number of fitness evaluations required to reach a given target is also significantly higher.

The local optimization procedure improves significantly the ability of the genetic algorithm to determine the global minimum (a global Δf_{target} -optimal solution) of the functions considered (increase of $P(\Delta f_{\text{target}})$). This conclusion was tested for statistical significance [42]. This technique also accelerates the algorithm by reducing the number of fitness evaluations (decrease of $\langle n_{\text{eval}} \rangle$). Although originally intended to accelerate the refinement of the final solution, this technique actually provides a useful guidance to the genetic algorithm by indicating, generation after generation, directions to consider based on collected data. This is especially useful for functions that require displacements in preferential directions, like the function #3 (Rosenbrock). It is also useful for functions whose large-scale structure leads to the global minimum despite the presence of many local minima, like the function #10 (Griewank). For functions that have a single minimum, like the function #1 (Sphere) and the function #2 (Rotated Hyper-Ellipsoid), the procedure is actually able to finalize the minimization as soon as a sufficient number of data points have been collected. Other functions, like the function #12 (Exponential), the function #17 (Brown 3) and the function #18 (Cigar), have their global minimum also much more rapidly determined.

TABLE 2. Results obtained for test problems in 5, 10 and 20 dimensions. The different columns correspond to results obtained when considering/not considering (i) local optimizations based on quadratic approximations of the fitness and (ii) a mutation operator that acts on randomly-shifted Gray codes. $P(\Delta f_{\text{target}})$ represents the probability to reach a target Δf_{target} by a single run of the GA. $\langle n_{\text{eval}} \rangle$ is the average number of fitness evaluations required to reach this target, counting runs that fail to meet the target. #fct(P≥10%) is the number of functions for which the target was reached at least once in ten runs. The last column provides for comparison the results obtained with CMA-ES when using the same population size of 50 individuals. These statistics were generated by running the genetic algorithm 100 times on each test function.

5 dimensions

Δf_{target}	Local Optim.	no	no	yes	yes	CMA-ES
	Shifted Gray	no	yes	no	yes	
10^{-4}	$P(\Delta f_{\text{target}})$	75.6%	83.2%	87.0%	94.9%	83.7%
	$\langle n_{\text{eval}} \rangle$	9051	7867	1866	1724	3287
	$\#\text{fct}(P \geq 10\%)$	18/22	20/22	21/22	all	21/22
10^{-3}	$P(\Delta f_{\text{target}})$	75.8%	83.4%	87.1%	94.9%	83.7%
	$\langle n_{\text{eval}} \rangle$	8256	7153	1751	1622	3017
	$\#\text{fct}(P \geq 10\%)$	18/22	20/22	21/22	all	21/22
10^{-2}	$P(\Delta f_{\text{target}})$	78.8%	86.4%	87.2%	94.9%	85.4%
	$\langle n_{\text{eval}} \rangle$	7037	5990	1604	1488	2630
	$\#\text{fct}(P \geq 10\%)$	19/22	20/22	21/22	all	21/22
10^{-1}	$P(\Delta f_{\text{target}})$	83.5%	91.6%	88.5%	95.5%	87.4%
	$\langle n_{\text{eval}} \rangle$	5389	4436	1248	1173	2170
	$\#\text{fct}(P \geq 10\%)$	21/22	all	21/22	all	21/22

10 dimensions

Δf_{target}	Local Optim.	no	no	yes	yes	CMA-ES
	Shifted Gray	no	yes	no	yes	
10^{-4}	$P(\Delta f_{\text{target}})$	62.5%	78.3%	77.8%	92.3%	81.7%
	$\langle n_{\text{eval}} \rangle$	30884	22981	5928	5104	6891
	$\#\text{fct}(P \geq 10\%)$	17/22	18/22	19/22	21/22	19/22
10^{-3}	$P(\Delta f_{\text{target}})$	63.4%	78.5%	78.1%	92.5%	81.7%
	$\langle n_{\text{eval}} \rangle$	28144	20834	5698	4892	6399
	$\#\text{fct}(P \geq 10\%)$	17/22	18/22	19/22	all	19/22
10^{-2}	$P(\Delta f_{\text{target}})$	67.0%	82.9%	78.5%	92.6%	81.9%
	$\langle n_{\text{eval}} \rangle$	24387	17323	5302	4541	5865
	$\#\text{fct}(P \geq 10\%)$	18/22	20/22	19/22	all	19/22
10^{-1}	$P(\Delta f_{\text{target}})$	71.8%	87.6%	79.2%	92.6%	82.0%
	$\langle n_{\text{eval}} \rangle$	19331	13371	4462	3896	5289
	$\#\text{fct}(P \geq 10\%)$	20/22	21/22	19/22	all	19/22

20 dimensions

Δf_{target}	Local Optim.	no	no	yes	yes	CMA-ES
	Shifted Gray	no	yes	no	yes	
10^{-4}	$P(\Delta f_{\text{target}})$	46.7%	76.1%	62.2%	89.0%	72.0%
	$\langle n_{\text{eval}} \rangle$	108941	61676	29643	19870	16081
	$\#\text{fct}(P \geq 10\%)$	15/22	18/22	17/22	20/22	18/22
10^{-3}	$P(\Delta f_{\text{target}})$	48.4%	76.4%	62.7%	89.3%	72.0%
	$\langle n_{\text{eval}} \rangle$	98986	56276	27280	18325	15242
	$\#\text{fct}(P \geq 10\%)$	15/22	18/22	17/22	20/22	18/22
10^{-2}	$P(\Delta f_{\text{target}})$	50.1%	79.5%	63.1%	89.5%	72.0%
	$\langle n_{\text{eval}} \rangle$	90715	49375	25404	16989	14377
	$\#\text{fct}(P \geq 10\%)$	15/22	19/22	17/22	20/22	18/22
10^{-1}	$P(\Delta f_{\text{target}})$	59.4%	84.9%	64.9%	89.6%	72.0%
	$\langle n_{\text{eval}} \rangle$	68611	39038	22229	15133	13426
	$\#\text{fct}(P \geq 10\%)$	16/22	20/22	17/22	20/22	18/22

TABLE 3. Results obtained for each test function when considering a target Δf_{target} of 10^{-4} for problems in 5 dimensions. The local optimization procedure as well as a mutation operator that acts on randomly-shifted Gray codes are used by the genetic algorithm. The quantities represented are the probability of success in one run ($P(\Delta f_{\text{target}})$), the average number of fitness evaluations required to reach the target counting runs that fail to meet the target ($\langle n_{\text{eval}}\rangle$), the average number of fitness evaluations required to reach the target counting only runs that reach the target ($\langle n^*_{\text{eval}}\rangle$) and the average number of generations required to reach the target counting only runs that reach the target ($\langle n^*_{\text{gen}}\rangle$). $\langle n^*_{\text{gen}}\rangle$ corresponds to the number of generations beyond that associated with the initial population. The standard deviation (std) of $\langle n_{\text{eval}}\rangle$, $\langle n^*_{\text{eval}}\rangle$ and $\langle n^*_{\text{gen}}\rangle$ is also indicated. These statistics were generated by running the genetic algorithm 100 times on each test function.

	n=5 dimensions			
#	P	$\langle n_{\text{eval}}\rangle$	$\langle n^*_{\text{eval}}\rangle$	$\langle n^*_{\text{gen}}\rangle$
1	100%	85±4	85±4	1±0
2	100%	86±5	86±5	1±0
3	99%	6250±1147	6156±1102	187±34
4	100%	1333±166	1333±166	38±5
5	100%	953±209	953±209	26±7
6	100%	1403±603	1403±603	42±22
7	100%	1235±213	1235±213	35±7
8	95%	2754±733	2565±694	85±27
9	100%	1823±343	1823±343	55±11
10	95%	1186±736	1004±474	30±17
11	100%	513±79	513±79	13±2
12	100%	270±55	270±55	6±2
13	100%	1040±215	1040±215	30±7
14	100%	1133±221	1133±221	33±7
15	100%	1893±392	1893±392	55±12
16	100%	1777±300	1777±300	54±10
17	100%	654±91	654±91	17±3
18	100%	1199±478	1199±478	41±18
19	98%	991±427	928±298	26±11
20	87%	2571±1392	1978±1152	60±42
21	99%	1341±669	1295±585	37±18
22	14%	43640±6590	2727±490	85±16

The use of a mutation operator that acts on randomly-shifted Gray codes provides a further boost to our results. Table 2 reveals indeed that the probability to determine the global minimum (a global Δf_{target}-optimal solution) of the functions considered by a single run of the genetic algorithm is improved by this technique. This conclusion was also tested for statistical significance [43]. It applies whether the local optimization procedure is used or not. Table 2 reveals consistently that the number of fitness evaluations required to determine the global minimum of the functions considered is reduced by this technique. The use of randomly-shifted Gray codes when applying mutations helps the genetic algorithm escape local minima, since the displacements generated by these mutations have a wider diversity (see Appendix A). This is especially useful for functions with many local minima, like the function #6 (Schwefel), the function #8 (Rastrigin), the function #11 (Cosine Mixture), the function #13 (Levy and Montalvo 1), the function #14 (Levy and Montalvo 2) and the function #21 (Pinter). The wider variety of displacements generated by the use of randomly-shifted Gray codes improves exploration of the decision variable space, which results in a higher probability to detect the global minimum of the functions considered. This technique represents a useful complement to the local optimization procedure used in this work.

TABLE 4. Results obtained for each test function when considering a target Δf_{target} of 10^{-4} for problems in 10 dimensions. The local optimization procedure as well as a mutation operator that acts on randomly-shifted Gray codes are used by the genetic algorithm. The quantities represented are the probability of success in one run ($P(\Delta f_{\text{target}})$), the average number of fitness evaluations required to reach the target counting runs that fail to meet the target ($\langle n_{\text{eval}} \rangle$), the average number of fitness evaluations required to reach the target counting only runs that reach the target ($\langle n^*_{\text{eval}} \rangle$) and the average number of generations required to reach the target counting only runs that reach the target ($\langle n^*_{\text{gen}} \rangle$). $\langle n^*_{\text{gen}} \rangle$ corresponds to the number of generations beyond that associated with the initial population. The standard deviation (std) of $\langle n_{\text{eval}} \rangle$, $\langle n^*_{\text{eval}} \rangle$ and $\langle n^*_{\text{gen}} \rangle$ is also indicated. These statistics were generated by running the genetic algorithm 100 times on each test function.

	n=10 dimensions			
#	P	$\langle n_{\text{eval}} \rangle$	$\langle n^*_{\text{eval}} \rangle$	$\langle n^*_{\text{gen}} \rangle$
1	100%	121±6	121±6	2±0
2	100%	120±7	120±7	2±0
3	99%	16962±3162	16766±3151	522±102
4	100%	3016±341	3016±341	90±11
5	100%	2407±372	2407±372	72±12
6	99%	4298±1445	4214±1383	136±51
7	100%	3164±414	3164±414	95±13
8	94%	7800±1720	7264±1673	251±67
9	100%	4952±666	4952±666	155±22
10	100%	458±580	458±580	12±19
11	100%	1079±196	1079±196	30±6
12	100%	585±58	585±58	15±2
13	100%	2352±361	2352±361	70±11
14	100%	2598±567	2598±567	78±19
15	100%	11596±2080	11596±2080	356±65
16	100%	6067±1041	6067±1041	197±37
17	100%	1495±219	1495±219	43±7
18	100%	2063±878	2063±878	75±35
19	98%	2449±722	2315±396	68±12
20	34%	26531±4548	9785±3652	308±126
21	100%	3541±1105	3541±1105	108±35
22	7%	260736±24993	11907±5253	397±189

The genetic algorithm presented in this work generally achieves good results on the test problems considered. The functions #20 (Trigonometric 1) and #22 (Whitley) remain however challenging. It is interesting at this point to compare our results with those provided by the reference algorithm CMA-ES [44-46]. CMA-ES, for Covariance-Matrix Adaptation-Evolution Strategy, is a genetic algorithm that relies on a real-value encoding of the decision variables. Mutations consist of random normally-distributed perturbations of the decision variables. The covariance matrix that actually controls the distribution of these mutations is adapted along the optimization. When applying CMA-ES to our test problems with the same population size of 50 individuals, it actually achieves a probability of success in one run $P(\Delta f_{\text{target}} = 10^{-4})$ of 84.1% for n=5 dimensions, 81.7% for n=10 dimensions and 72.0% for n=20 dimensions [47]. These results are included in Table 2. A detailed analysis of the results achieved with CMA-ES on individual test functions for n=20 dimensions can be found in Table 6. The comparison with Table 5 shows that the algorithm presented in this work achieves respectable performances for the class of problems considered. The use of a mutation operator that acts on randomly-shifted Gray codes enables indeed our genetic algorithm to escape local optima more easily. This improves its ability to determine the true global minimum of the multimodal functions considered in this work.

TABLE 5. Results obtained for each test function when considering a target Δf_{target} of 10^{-4} for problems in 20 dimensions. The local optimization procedure as well as a mutation operator that acts on randomly-shifted Gray codes are used by the genetic algorithm. The quantities represented are the probability of success in one run ($P(\Delta f_{\text{target}})$), the average number of fitness evaluations required to reach the target counting runs that fail to meet the target ($\langle n_{\text{eval}} \rangle$), the average number of fitness evaluations required to reach the target counting only runs that reach the target ($\langle n^*_{\text{eval}} \rangle$) and the average number of generations required to reach the target counting only runs that reach the target ($\langle n^*_{\text{gen}} \rangle$). $\langle n^*_{\text{gen}} \rangle$ corresponds to the number of generations beyond that associated with the initial population. The standard deviation (std) of $\langle n_{\text{eval}} \rangle$, $\langle n^*_{\text{eval}} \rangle$ and $\langle n^*_{\text{gen}} \rangle$ is also indicated. These statistics were generated by running the genetic algorithm 100 times on each test function.

	n=20 dimensions			
#	P	$\langle n_{\text{eval}} \rangle$	$\langle n^*_{\text{eval}} \rangle$	$\langle n^*_{\text{gen}} \rangle$
1	100%	284±11	284±11	6±0
2	100%	283±11	283±11	6±0
3	98%	74216±17673	72168±17207	2340±572
4	100%	9700±1271	9700±1271	303±41
5	100%	6313±757	6313±757	194±24
6	99%	11840±2766	11624±2586	377±98
7	100%	8380±818	8380±818	259±26
8	97%	18950±3498	18269±3430	631±138
9	100%	13833±1458	13833±1458	444±50
10	100%	623±1102	623±1102	18±37
11	100%	3893±759	3893±759	118±24
12	100%	2228±323	2228±323	65±10
13	100%	6264±809	6264±809	193±27
14	100%	8142±1519	8142±1519	255±56
15	95%	89761±11592	84907±11222	2707±367
16	100%	18387±2402	18387±2402	620±87
17	100%	3986±442	3986±442	120±14
18	100%	3715±1389	3715±1389	143±56
19	91%	7866±2975	6354±815	195±28
20	4%	677342±48424	49436±19695	1590±674
21	73%	40537±22437	24540±15579	799±526
22	1%	5108453	44272	1509

TABLE 6. Results obtained with CMA-ES for each test function when considering a target Δf_{target} of 10^{-4} for problems in 20 dimensions. CMA-ES is used with a population size of 50 individuals. The quantities represented are the probability of success in one run ($P(\Delta f_{\text{target}})$), the average number of fitness evaluations required to reach the target counting runs that fail to meet the target ($\langle n_{\text{eval}} \rangle$), the average number of fitness evaluations required to reach the target counting only runs that reach the target ($\langle n^*_{\text{eval}} \rangle$) and the average number of generations required to reach the target counting only runs that reach the target ($\langle n^*_{\text{gen}} \rangle$). $\langle n^*_{\text{gen}} \rangle$ corresponds to the number of generations beyond that associated with the initial population. The standard deviation (std) of $\langle n_{\text{eval}} \rangle$, $\langle n^*_{\text{eval}} \rangle$ and $\langle n^*_{\text{gen}} \rangle$ is also indicated. These statistics were generated by running CMA-ES 100 times on each test function.

	n=20 dimensions (CMA-ES)			
#	P	$\langle n_{\text{eval}} \rangle$	$\langle n^*_{\text{eval}} \rangle$	$\langle n^*_{\text{gen}} \rangle$
1	100%	4582±214	4582±214	92±4
2	100%	9000±314	9000±314	180±6
3	100%	34559±1151	34559±1151	691±23
4	100%	6662±310	6662±310	133±6
5	4%	22162±15607	5688±278	114±6
6	0%	/	/	/
7	100%	5704±263	5704±263	114±5
8	0%	/	/	/
9	96%	9261±1882	8622±378	172±8
10	100%	7250±293	7250±293	145±6
11	99%	4804±741	4685±211	94±4
12	100%	3399±191	3399±191	68±4
13	100%	5010±302	5010±302	100±6
14	100%	5022±316	5022±316	100±6
15	100%	8199±313	8199±313	164±6
16	100%	9232±341	9232±341	185±7
17	100%	5532±327	5532±327	111±7
18	100%	14186±451	14186±451	284±9
19	28%	104618±47707	6322±3318	126±66
20	23%	67967±3169	13818±1450	276±29
21	35%	39899±6176	9197±605	184±12
22	0%	/	/	/

Application in Optical Engineering

In order to provide a real-world application in optical engineering, we consider the maximization of broadband absorption by a metamaterial. The structures considered in this work consist of 2-D periodic arrays of truncated square-based pyramids made of 3 stacks of titanium/poly (methyl methacrylate) (Ti/PMMA) layers (see Fig. 1). These pyramids stand on a flat support that consists of successive uniform layers of Au (60 nm), Cr (5 nm) and amorphous Si (1 micron). Previous work has shown that periodic arrays of truncated square-based pyramids made of successive stacks of metal/dielectric layers can lead to the quasi-perfect absorption of electromagnetic radiations over a wide wavelength range. By considering pyramids made of 20 stacks of Au/Ge layers, Lobet et al. could indeed achieve an integrated absorptance of 98% of incident light over a 0.2-5.8 µm wavelength range [48, 49]. This ultra-broadband absorption is essentially due to (i) an efficient anti-reflection property of these pyramidal structures [50, 51] and (ii) a well-designed coupling between the localized surface plasmons found at the metal/dielectric interfaces of each stack [52-55].

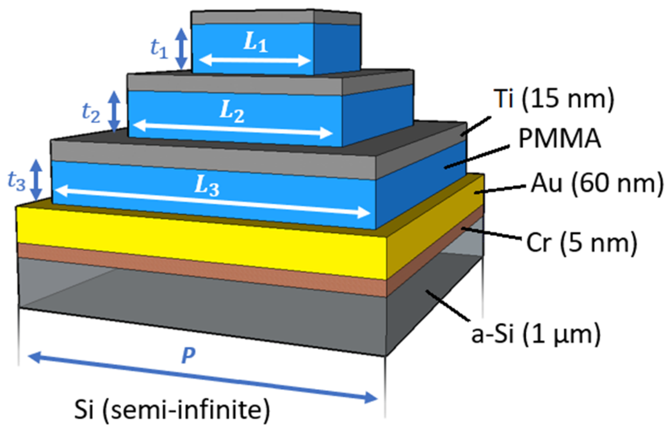

FIG. 1. Square-based pyramids made of 3 stacks of Ti/PMMA layers. The support of the pyramids consists of uniform layers of Au (60 nm), Cr (5 nm) and a-Si (1 micron). We assume an infinite substrate of Si (ε=16).

In order to reduce the difficulty of fabricating structures made of many different layers, we will consider in this work pyramids that consist of only three stacks of Ti/PMMA layers (see Fig. 1 again). Our objective is to maximize the absorption of incident radiations in the wavelength range 420-1600 nm by tuning the geometrical parameters of the system. The objective function (fitness) for this problem is therefore defined by $\eta(\%) = 100 \times \frac{\int_{\lambda_{\min}}^{\lambda_{\max}} A(\lambda) d\lambda}{\lambda_{\max} - \lambda_{\min}}$, where $\lambda_{\min}$=420 nm and $\lambda_{\max}$=1600 nm. $A(\lambda)$ refers to the absorptance of normally incident radiations at the wavelength λ. It is calculated by a Rigorous Coupled Waves Analysis (RCWA) method [56, 57]. This method solves Maxwell's equations numerically in laterally periodic systems. We used this method with 11×11 plane waves and reported values for the refractive indices [58-60]. The parameters to determine in order to maximize the figure of merit η are (i) the lateral period P of the system, (ii) the lateral dimensions L_1, L_2 and L_3 of the three stacks of Ti/PMMA layers and (iii) the thicknesses t_1, t_2 and t_3 of the three PMMA layers (the subscripts 1, 2 and 3 refer respectively to the top, medium and bottom stacks of the nanopyramids). The thickness of each Ti layer is fixed at 15 nm. In order to reduce the search to a realistic range, we actually consider P values between 50 and 500 nm, L_1, L_2 and L_3 values between 50 and 500 nm and t_1, t_2, t_3 values between 50 and 250 nm. We account for the experimental resolution with which these structures can possibly be fabricated by considering a discretization step of 1 nm for these different quantities. In order to obtain pyramidal structures, we finally impose that the genetic algorithm only considers solutions for which $L_1 < L_2 < L_3 \leq P$ [61]. With these specifications, we hence have seven decision variables to determine and 1.3×10^{16} possible parameter combinations! Each simulation takes approximately one hour of CPU time. We are therefore in conditions where it is impossible to test all parameter combinations. We are also in conditions where the time required by the fitness evaluations is largely superior to the time required for running the genetic algorithm.

In order to show the advantage of using the techniques developed in Appendix B and Appendix C, we represent in Fig. 2 the fitness (figure of merit η) of the best individual as a function of the number of generations. When using a mutation operator that acts on randomly-shifted Gray codes (Appendix B) and a local optimization procedure that analyzes the collected data (Appendix C), the genetic algorithm determines after 167 generations and 4628 fitness evaluations the final solution (global optimum associated with a figure of merit η=99.757%; the parameters found by the GA are the following: L_1=155 nm, t_1=124 nm, L_2=285 nm, t_2=126 nm, L_3=416 nm, t_3=98 nm and P=416 nm). If all fitness calculations in a given generation run in parallel, this solution is actually obtained after 7 days. When the techniques described in Appendix B and Appendix C are not used, the genetic algorithm stops after 266 generations and 6275 fitness evaluations without finding the global optimum (the solution found in this case corresponds to a figure of merit η=99.726%; the parameters associated with this solution are the following: L_1=161 nm, t_1=125 nm, L_2=295 nm, t_2=126 nm, L_3=431 nm, t_3=97 nm and P=431 nm). The GA stopped in this case, because the mean value of the genetic similarity s over the last $1.5\times n_{\text{bits}}$ generations was higher than $1\text{-}3m$, where the total number of bits n_{bits} is 60 and the mutation rate $m=0.95/n_{\text{bits}}$ is 1.6% for this application. If all fitness calculations in a given generation run in parallel, this sub-optimal solution is obtained after 11 days. As shown in the previous section, several runs are typically necessary on difficult problems when the techniques of Appendix B and Appendix C are not used. This would be the case here. Fig. 2 shows that the modified version of the genetic algorithm (techniques of Appendix B and Appendix C used) actually outperforms the classical version of the genetic algorithm (techniques of Appendix B and Appendix C not used) after only 50 generations.

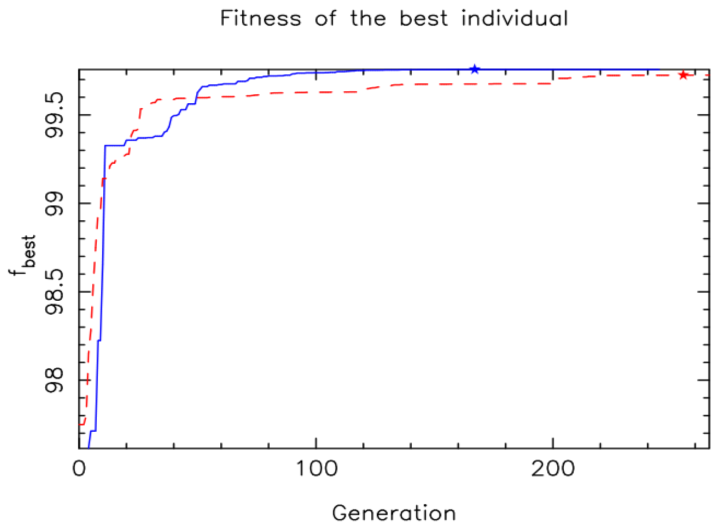

FIG. 2. Best fitness (figure of merit η) when optimizing a structure made of three stacks of Ti/PMMA layers. Solid: the GA is used with a mutation operator that acts on randomly-shifted Gray codes (Appendix B) and a local optimization procedure (Appendix C). Dashed: the GA does not use the techniques developed in Appendix B and Appendix C. The stars indicate when the best solution is found.

Conclusions

This article describes a genetic algorithm that we developed in order to address computationally expensive optimization problems in optical engineering. For these problems, the decision variables are characterized by a finite set of possible values due to experimental limitations in the fabrication of a device. A target accuracy of 10^{-4} on the objective function is also sufficient for these applications. The technical parameters of our algorithm were tuned to address these conditions. The organization of the algorithm enables a massive parallelization of the fitness calculations. The data collected by the genetic algorithm is analyzed by a local optimization procedure in order to infer more rapidly the final solution. This procedure, which relies on quadratic approximations of the fitness in the close neighborhood of the best-so-far solution, provides a useful guidance to the genetic algorithm by indicating, generation after generation, directions to consider based on these collected data. We also use a mutation operator that acts on randomly-shifted Gray codes. This helps the genetic algorithm escape local optima. It also improves the exploration of the decision variable space by enabling a wider diversity of displacements. We applied this algorithm to a set of 22 benchmark problems in 5, 10 and 20 dimensions in order to demonstrate its performance. The results prove that the techniques presented in this work improve significantly the ability of the genetic algorithm to determine the global minimum of these problems. The average number of fitness evaluations required to determine these solutions is also significantly reduced. This algorithm was already applied successfully to a variety of computationally expensive optimization problems in optical engineering. We showed in this article how these techniques accelerate the optimization of square-based pyramidal structures for the broadband absorption of electromagnetic radiations.

Appendix A: Pseudocode of the Genetic Algorithm

```
Initialize a Population of $n_{pop}$ random individuals.
Compute the fitness $f(\vec{x})$ of each individual in the Population.
Save the calculated $\{\vec{x}, f(\vec{x})\}$ data in the Records.
Sort the Population from best to worst individuals.
Save $\{\vec{x}_{best}, f_{best}\}$=best-so-far solution.

For $k$ ranging from 1 to $n_{gen}$:
  Compute genetic similarity $s$ of the Population.
  Set $p = |s - 0.5|/0.5$,
    $n_{rand}$=even[0.1×$n_{pop}$×(1-$p$)] and $N$=$n_{pop}$-$n_{rand}$.
  Define, for the modified mutation operator, a random shift$_i \in [0, 2^{n_i} - 1]$ for each gene $i \in [1, n]$.

  Pool($N$+1:$n_{pop}$) = $n_{rand}$ random individuals.
  For $i$ ranging from 1 to $N$/2:
    Select Parent$_1$ in Population(1:$N$) by a rank-based roulette wheel selection.
    Select Parent$_2$ in Population(1:$N$) by a rank-based roulette wheel selection.
    If rnd ≤ 0.7:
      {Child$_1$,Child$_2$}=1-point crossover between {Parent$_1$,Parent$_2$}.
      Apply_Mutation=True.
    Else:
      {Child$_1$,Child$_2$}={Parent$_1$,Parent$_2$}.
      Apply_Mutation=False.
    If Apply_Mutation:
      Apply modified mutation operator on Child$_1$ (see Appendix B).
      Apply modified mutation operator on Child$_2$ (see Appendix B).
    Pool(1+($i$-1)*2)=Child$_1$.
    Pool(2+($i$-1)*2)=Child$_2$.
  Guess=Local Optimization using $\{\vec{x}, f(\vec{x})\}$ data in the Records (see Appendix C).
  If Guess can be accepted:
    Pool($N$)=Guess.

  Check the Records to avoid any duplication in the fitness evaluations.
  Compute the fitness $f(\vec{x})$ of each new individual in the Pool.
  Save the calculated $\{\vec{x}, f(\vec{x})\}$ data in the Records.
  Sort the Pool from best to worst individuals.
  Set new Population=Pool.

  If best individual in new Population not as good as previous $\{\vec{x}_{best}, f_{best}\}$:
    Choose random integer $i \in [1, n_{pop}]$.
    Population($i$)= $\vec{x}_{best}$.
    Update sorting of Population.
  Save $\{\vec{x}_{best}, f_{best}\}$=best-so-far solution.
  Exit if a stopping criterion is met.
```

Appendix B: Modified Mutation Operator Based on Randomly-Shifted Gray Codes

The decision variables are represented by $x_i = x_i^{\min} + \langle \text{gene } i \rangle \times \Delta_i$, where $\langle \text{gene } i \rangle \in [0, 2^{n_i} - 1]$ stands for the value coded by the n_i binary digits of the gene. We use the Gray code to interpret the value of this gene [5, 39]. A Gray code is characterized by the fact that successive numbers differ only by one bit (see Table 7). It is therefore always possible to move from x_i to $x_i + \Delta x_i$ by changing a single bit. This is an advantage compared to standard binary, where several bit changes are typically necessary [62]. The use of Gray codes enables thus mutations to perform a fine tuning of the decision variables. By changing the n_i-2 other bits of the gene, mutations will generate wider displacements in the decision variable space. These wider displacements are important for exploration. The displacements generated by mutations depend however artificially on the coding considered and this is a limit to exploration.

TABLE 7. Comparison between decimal, standard binary, the original Gray code and a shifted version of the Gray code (circular permutation by 3 steps).

Decimal	Binary	Gray	Gray+3
0	000	000	010
1	001	001	110
2	010	011	111
3	011	010	101
4	100	110	100
5	101	111	000
6	110	101	001
7	111	100	011

The idea to improve the mutation operator is hence to apply this operator to the encoding obtained with shifted versions of the Gray code. It consists actually of a circular permutation of the original encoding; see last column of Table 7 [62-64]. At each generation, a random shift in the range $[0, 2^{n_i} - 1]$ is attributed to each gene. This shift is specific to the gene. It is identical for all individuals of the current generation. Its value is reset at each generation. A possible implementation of the modified mutation operator is given in Table 8. This modified mutation operator receives genes that are expressed in the original Gray code. Before applying mutations, the original chain of binary digits $\langle \text{gene } i \rangle$ is translated from the original Gray code to the shifted Gray code (in Table 7, this comes to moving from column 3 to column 4 on the line associated with the original encoding). Mutations are then applied on the modified encoding. The result is finally translated back from the shifted Gray code to the original Gray code (in Table 7, this comes to moving back from column 4 to column 3 on the line associated with the modified version of the gene). Since the result of this modified mutation operator is expressed in the original Gray code (reference encoding used in the rest of the algorithm), adaptation related to this reference encoding can still take place.

TABLE 8. Possible implementation of the modified mutation operator. Operations 1, 2 and 3 transform $\langle \text{gene } i \rangle$ from the original Gray code to the shifted Gray code. Operation 4 introduces mutations on the encoding obtained with this shifted Gray code. Operations 5, 6 and 7 transform the modified gene from the shifted Gray code to the original Gray code. The shift assigned to each gene is the same for all individuals in the population. It is reset randomly at each generation.

```
Input : $\langle \text{gene}_i \rangle$ (Gray code)

Operations:
1. Decode the gene : $\langle \text{gene}_i \rangle$ (Gray code) $\rightarrow$ integer
2. Apply the shift : integer $\rightarrow$ mod(integer+shift,$2^{n_i}$)
3. Get Gray code representation : integer $\rightarrow$ Gray code
4. Apply bit-wise mutations
5. Decode the gene : Gray code $\rightarrow$ integer
6. Remove the shift : integer $\rightarrow$ mod(integer-shift,$2^{n_i}$)
7. Recode the gene : integer $\rightarrow$ $\langle \text{gene}_i \rangle$ (Gray code)

Ouput : $\langle \text{gene}_i \rangle$ (Gray code) with mutations
```

Illustrative example: Let us consider the number "3" (010 in the original Gray code; see third column of Table 7). Individual bit flips can lead to "2" (011), "4" (110) and "0" (000). This possible transition between "3" and "0" is specific to the original Gray code. There is no direct transition to the other entries of the table. If we consider a circular permutation by three steps of the original Gray code (last column of Table 7), the number "3" is now encoded by "101". Individual bit flips lead now to "2" (111), "4" (100) and "6" (001). There is a possible transition between "3" and "6" (instead of "3" and "0"). By changing the shift introduced in the Gray code at each generation, we reset the transitions generated by individual bit flips.

Illustration with Rastrigin's function: Rastrigin's function (fct#8 in Table 1) provides a good illustration for the benefit of using randomly-shifted Gray codes when applying mutations. This function has many local minima. The global minimum is for x_i=0 (i=1,... n). When searching for the global minimum of Rastrigin's function in n=10 dimensions, it turns out that the algorithm described in Sec. 2 fails most of the times at finding this global minimum if the mutation operator does not shift the Gray code. The reason is that x_i=0 is represented by 110000000000 in our case if we work in the original domain [-5.12, 5.12] (we have indeed $x_i^{\min}$=-5.12 and Δx_i=0.0025; a gene value of 2048 is represented by 110000000000 in the original Gray code). The closest local minimum is at x_i= 0.995, which is represented by 110101001001. There is a difference of four bits between these two encodings and the genetic algorithm has a hard time finding the appropriate bit changes once trapped in this local minimum. Fig. 3 shows that there is a poor diversity in the displacements generated by mutations if no shifting of the Gray code is considered. By considering randomly-shifted versions of the Gray code when applying mutations, we increase the diversity of the displacements generated by these mutations. This helps the genetic algorithm escape the local minimum to eventually find the global minimum. The second part of Fig. 3 shows that there is indeed a wider diversity in the displacements generated by mutations when considering randomly-shifted Gray codes.

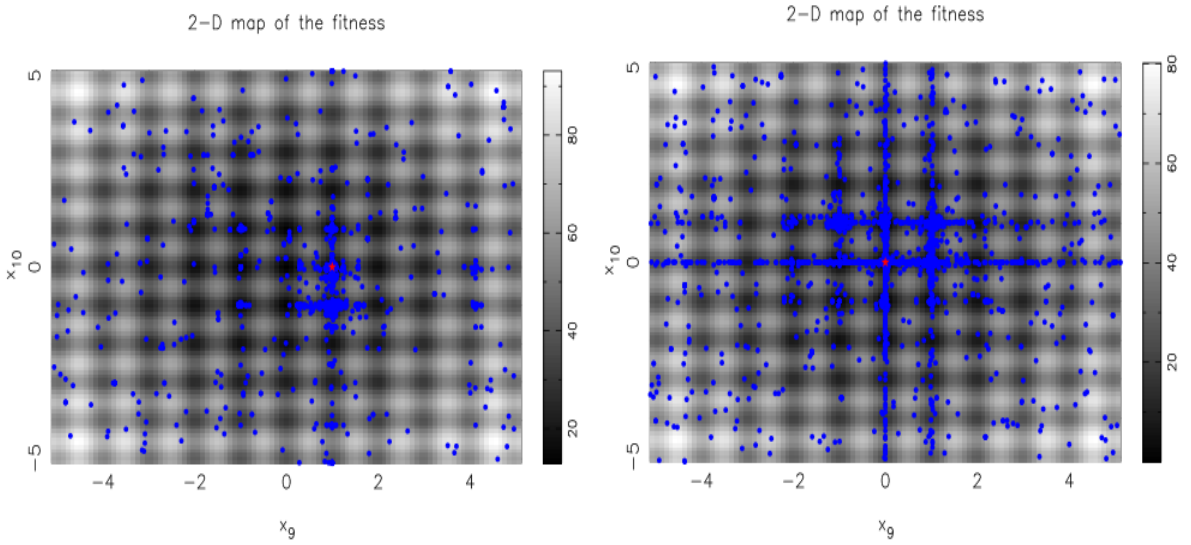

FIG. 3. Application of the genetic algorithm to Rastrigin's function in 10 dimensions. The blue dots represent individuals considered by the genetic algorithm. The star represents the best solution found by the algorithm. The algorithm was interrupted after 10000 evaluations of the fitness. Top: There is no shift of the Gray code when applying mutations; the genetic algorithm is trapped in a local minimum. Bottom: Mutations are applied to randomly-shifted versions of the Gray code; the algorithm finds the global minimum.

Appendix C: Local Optimization Based on a Quadratic Approximation of the Fitness

The data collected by the genetic algorithm can be analyzed, generation after generation, in order to infer more rapidly the final solution. The idea consists of establishing a quadratic approximation of the fitness in the close neighborhood of the best-so-far solution. We then inject in the population an individual that corresponds to the optimum of this approximation [65]. We chose as reference point ($\vec{x}_{\mathrm{ref}}$) the best-so-far solution found by the genetic algorithm. In order to establish the quadratic approximation, we will use N_{select} distinct data points from the records established by the genetic algorithm. A data point $\vec{x}$ is selected if $\max_i \frac{|x_i - x_{i,\mathrm{ref}}|}{\Delta x_i} \leq W$, where W specifies the width of the selection, in units of Δx_i. We take W=5 as initial value each time we start this procedure.

The expression to establish has the form:

$$f(\vec{x}) = a_0 + \vec{A}_1.\vec{X} + \tfrac{1}{2}\vec{X}.A_2\vec{X}, \qquad (1)$$

where $\vec{X} = \Delta^{-1}(\vec{x} - \vec{x}_{\mathrm{ref}})$ with $\Delta = \mathrm{diag}[\Delta x_1, \ldots \Delta x_n]/\max_i \Delta x_i$ a diagonal matrix that contains appropriate scaling factors. a_0 is a scalar, $\vec{A}_1$ is a vector of size n and A_2 is a symmetric matrix of size $n \times n$. Since A_2 is symmetric, there is a total of $N_{\mathrm{coeff}}=1+n+n.(n+1)/2$ coefficients to determine. We must ensure at this point that $N_{\mathrm{select}} \geq 2N_{\mathrm{coeff}}$, by increasing W if needed. To establish the quadratic approximation, we define a vector $\vec{f}$ of size N_{select} that contains the $f(\vec{x})$ values of the selected data points and a vector $\vec{A}$ of size N_{coeff} that contains the unknown coefficients in a_0, $\vec{A}_1$ and A_2. The equation to solve can then be written as: $\vec{f} = M\vec{A}$, where M is an $N_{\mathrm{select}} \times N_{\mathrm{coeff}}$ matrix with coefficients defined from Eq. (1). Since the system $\vec{f} = M\vec{A}$ is overdetermined, we actually require that $\|\vec{f} - M\vec{A}\|^2$ be minimized (by an appropriate choice of $\vec{A}$). We compute for this purpose the singular values decomposition (SVD) of the matrix M [66]. This gives $M = U\Sigma V^t$, where U is an orthonormal matrix of size $N_{\mathrm{select}} \times N_{\mathrm{coeff}}$ and V is an orthonormal matrix of size $N_{\mathrm{coeff}} \times N_{\mathrm{coeff}}$. Σ is a diagonal matrix of size $N_{\mathrm{coeff}} \times N_{\mathrm{coeff}}$ that contains the singular values σ_k of the matrix M. The solution of $\min\|\vec{f} - M\vec{A}\|^2$ is then given by $\vec{A} = V\Sigma^+ U^t \vec{f}$, where Σ^+ is a diagonal matrix of size $N_{\mathrm{coeff}} \times N_{\mathrm{coeff}}$ whose diagonal elements are defined by σ_k^{-1} if $\sigma_k \geq \varepsilon \times \sigma_{\max}$ (with $\sigma_{\max} = \max_k \sigma_k$) and 0 otherwise. ε accounts for the relative accuracy of $f(\vec{x})$.

Once the quadratic approximation has been established, the solution of $\vec{\nabla} f = 0$ is given formally by $\vec{x}^* = \vec{x}_{\mathrm{ref}} - \Delta A_2^{-1}\vec{A}_1$. Since the matrix A_2 may be non-invertible, we use an approach based on the spectral decomposition of A_2. Since the matrix A_2 is symmetric, its eigensystem $A_2\vec{x}_k = \lambda_k \vec{x}_k$ is characterized by real eigenvalues λ_k and its eigenvectors $\vec{x}_k$ form an orthonormal basis. It is useful at this point to define $\lambda_{\max} = \max_k |\lambda_k|$ and $\lambda_{\min} = \min_k |\lambda_k|$. The solution of $\vec{\nabla} f = 0$ can then be expressed as:

$$\vec{x}^* = \vec{x}_{\mathrm{ref}} - \Delta \sum_k \frac{\vec{x}_k.\vec{A}_1}{\lambda_k} \vec{x}_k, \qquad (2)$$

where the sum is restricted to the eigenvalues λ_k that satisfy $|\lambda_k| \geq \varepsilon_{\mathrm{inv}} \times \lambda_{\max}$ in order to avoid numerical instabilities. For analytical functions, we take ε=10^{-10} and $\varepsilon_{\mathrm{inv}} = 10\frac{\lambda_{\max}}{\lambda_{\min}}\epsilon$. For problems in which the fitness has an accuracy limited to three significant digits, we recommend using $\varepsilon_{\mathrm{inv}} = \varepsilon =$ 10^{-3}. If the solution $\vec{x}^*$ provided by this approach can be accepted, it replaces the last individual scheduled for the next generation. We repeat otherwise this procedure up to three times by increasing the width of the selection ($W \to W + 2$).

Acknowledgments

Alexandre Mayer is funded by the Fund for Scientific Research (F.R.S.-FNRS) of Belgium. He is member of NaXys, Namur Institute for Complex Systems, University of Namur, Belgium. This work was performed while Michaël Lobet was a recipient of a fellowship of the Belgian American Educational Foundation. The authors acknowledge Benoit Hackens, Nicolas Reckinger, Luc Henrard and Sarah Griesse-Nascimento for useful discussions on the nanopyramids. This research used resources of the "Plateforme Technologique de Calcul Intensif (PTCI)" (http://www.ptci.unamur.be) located at the University of Namur, Belgium, which is supported by the F.R.S.-FNRS under the convention No. 2.5020.11. The PTCI is member of the "Consortium des Equipements de Calcul Intensif (CECI)" (http://www.ceci-hpc.be).

References

[1] Holland, J., "Adaptation in Natural and Artificial Systems", (University of Michigan Press, Ann Arbor, Mich., 1975).

[2] De Jong, K., Ph.D. Thesis, University of Michigan, (1975), Ann Arbor, Mich.

[3] Goldberg, D., "Genetic Algorithms in Search, Optimization and Machine Learning", (Addison-Wesley, Reading, Mass., 1989).

[4] Haupt, R. and Werner, D., "Genetic Algorithms in Electromagnetics", (J. Wiley and Sons, Hoboken, NJ, 2007).

[5] Eiben, A. and Smith, J., "Introduction to Evolutionary Computing", 2nd Edn. (Springer-Verlag, Berlin, 2007).

[6] Eiben, A. and Smith, J., Nature, 521 (2015) 476.

[7] Hinton, G. and Nowlan, S., Complex Systems, 1 (1987) 495.

[8] Krasnogor, N. and Smith, J., IEEE T. Evolut. Comput., 9 (2005) 474.

[9] Chen, X., Ong, Y.-S., Lim, M.-H. and Tan, K., IEEE T. Evolut. Comput., 15 (2011) 591.

[10] Neri, F., Cotta, C. and Moscato, P., "Handbook of Memetic Algorithms", (Springer, Berlin, 2011).

[11] Posik, P., Huyer, W. and Pal, L., Evol. Comput., 20 (2012) 509.

[12] Sapin, E., De Jong, K. and Shehu, A., Proceedings of the Genetic and Evolutionary Computation Conference, Denver (2016), 85.

[13] Nguyen, P. and Sudholt, D., Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto (2018), 1071.

[14] Powell, M., Large-Scale Non-linear Optimization, 83 (2006) 255.

[15] Wanner, E., Guimaraes, F., Takahashi, R. and Fleming, P., IEEE C. Evolut. Comput., (2007) 677.

[16] Wanner, E., Guimaraes, F., Takahashi, R. and Fleming, P., Evol. Comput., 16 (2008) 185.

[17] Deep, K. and Das, K., Appl. Math. Comput., 203 (2008) 86.

[18] da Cruz, A., Wanner, E., Cardoso, R. and Takahashi, R., IEEE C. Evol. Computat., (2011) 1217.

[19] Fonseca, C. and Wanner, E., IEEE C. Evol. Computat., (2016) 4911.

[20] Rasheed, K., Ni, X. and Vattam, S., Soft Comput., 9 (2005) 29.

[21] Regis, R. and Shoemaker, C., IEEE T. Evolut. Comput., 8 (2004) 490.

[22] Paenke, I., Branke, J. and Jin, Y., IEEE T. Evolut. Comput., 10 (2006) 405.

[23] Jones, D.R., J. Global Optim., 21 (2001) 345.

[24] Jin, Y., Swarm Evol. Comput., 1 (2011) 61.

[25] Forrester, A., Sobester, A. and Keane, A., "Engineering Design *via* Surrogate Modelling: A Practical Guide", (J. Wiley and Sons, Chichester, UK, 2008).

[26] De Lucia, A., M., D.P., Oliveto, R. and Panichella, A., Proceedings of the Genetic and Evolutionary Computation Conference, Philadelphia (2012), 617.

[27] Martin, J. and Rasheed, K., Proceedings of the 2003 Congress on Evolutionary Computation, Canberra (2003), 1612.

[28] Marim, L., Lemes, M. and Dal Pino, A., Phys. Rev. A, 67 (2003) 033203.

[29] Javadi, A., Farmani, R. and Tan, T., Adv. Eng. Inform., 19 (2005) 255.

[30] Patra, T., Meenakshisundaram, V., Hung, J.-H. and Simmons, D., ACS Comb. Sci., 19 (2017) 96.

[31] Garciarena, U., Santana, R. and Mendiburu, A., Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto (2018), 849.

[32] Mayer, A. and Bay, A., J. Opt., 17 (2015) 025002.

[33] Mayer, A., Gaouyat, L., Nicolay, D., Carletti, T. and Deparis, O., Opt. Express, 22 (2014) A1641.

[34] Mayer, A., Muller, J., Herman, A. and Deparis, O., Proc. SPIE 9546, San Diego (2015), 95461N.

[35] Lin, A. and Phillips, J., Sol. Energ. Mat. Sol. C., 92 (2008) 1689.

[36] Wang, C., Yu, S., Chen, W. and Sun, C., Sci. Rep., 3 (2013) 1.

[37] Yu, S., Wang, C., Sun, C. and Chen, W., Struct. Multidisc. Optim., 50 (2014) 367.

[38] Judson, R., Reviews in Computational Chemistry, 10 (1997) 1.

[39] Smith, J., "Evolutionary Genetics", 2nd Edn., (Oxford University Press, 1998).

[40] Mayer, A., Proceedings of the Genetic and Evolutionary Computation Conference Companion, Berlin (2017), 195.

[41] For Schwefel 7 (fct #6 in Table 1), we actually consider n_i=16 bits per gene, since we can otherwise not get sufficiently close to the exact solution x_i=420.96874636.

[42] The hypothesis H$_0$="$P(\Delta f_{\text{target}})$ not improved by the local optimization procedure" is rejected at a confidence level α=0.005 by a right-tailed z-test, where $\mu_0 = P(\Delta f_{\text{target}})$ for the reference model (no local optimization) and $s_0^2 = \#\text{run} * \mu_0 * (1 - \mu_0)$. We used Δf_{target}=10^{-4} and #run=100 in our numerical experiment. This confidence level holds for problems in 5, 10 and 20 dimensions and whether shifted Gray codes are used or not.

[43] The hypothesis H$_0$="$P(\Delta f_{\text{target}})$ not improved by shifted Gray codes" is rejected at a confidence level α=0.05 by a right-tailed z-test, where $\mu_0 = P(\Delta f_{\text{target}})$ for the reference model (no shift of the Gray code) and $s_0^2 = \#\text{run} * \mu_0 * (1 - \mu_0)$. We used Δf_{target}=10^{-4} and #run=100 in our numerical experiment. This confidence level holds for problems in 5, 10 and 20 dimensions and whether the local optimization procedure was used or not.

[44] Hansen, N. and Ostermeier, A., Evol. Comput., 9 (2001) 159.

[45] Hansen, N., "Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms", (Springer, Berlin, 2006), 75.

[46] Hansen, N., Proceedings of the Genetic and Evolutionary Computation Conference, Montreal (2009), 2389.

[47] CMA-ES accounts for the boundaries $[x_i^{\min}, x_i^{\max}]$ specified in Table 1. The starting point $\langle \vec{x} \rangle_0$ used by CMA-ES is a random position in the search domain. We finally take $\sigma^{(0)} = (x_i^{\max} - x_i^{\min})/3$ as recommended.

[48] Lobet, M., Lard, M., Sarrazin, M., Deparis, O. and Henrard, L., Opt. Express, 22 (2014) 12678.

[49] Lobet, M. and Henrard, L., 8th International Congress on Advanced Electromagnetic Materials in Microwaves and Optics, Copenhagen (2014), 190.

[50] Clapham, P. and Hutley, M., Nature, 244 (1973) 281.

[51] Deparis, O., Vigneron, J.-P., Agustsson, O. and Decroupet, D., J. Appl. Phys., 106 (2009) 094505.

[52] Prodan, E., Radlo, C., Halas, N. and Nordlander, P., Science, 302 (2003) 419.

[53] Christ, A., Zentgraf, T., Tikhodeev, S., Gippius, N., Kuhl, J. and Giessen, H., Phys. Rev. B, 74 (2006) 155435.

[54] Liu, N., Guo, H., Fu, L., Kaiser, S., Schweizer, H. and Giessen, H., Adv. Mater., 19 (2007) 3628.

[55] Pu, M., Feng, Q., Hu, C. and Luo, X., Plasmonics, 7 (2012) 733.

[56] Moharam, M. and Gaylord, T., J. Opt. Soc. Am. A, 71 (1981) 811.

[57] Lobet, M. and Deparis, O., Proc. SPIE 8425, Brussels (2012), 842509.

[58] Johnson, P. and Christy, R., Phys. Rev. B, 9 (1974) 5056.

[59] Ordal, M., Bell, R., Alexander, R., Newquist, L. and Querry, M., Appl. Opt., 27 (1988) 1203.

[60] Beadie, G., Brindza, M., Flynn, R., Rosenberg, A. and Shirk, J., Appl. Optics, 54 (2015) 139.

[61] For applications with constraints on acceptable gene values, the genetic algorithm will only consider individuals that match these constraints. The crossover operator makes in this case n_{bits}-1 attempts to generate children with acceptable gene values. If these attempts fail, children will be simple copies of the parents. The mutation operator is repeated from scratch on the input DNA until it generates a DNA with acceptable gene values.

[62] Rowe, J., Whitley, D., Barbulescu, L. and Watson, J.-P., Evol. Comput., 12 (2004) 47.

[63] Barbulescu, L., Watson, J.-P. and Whitley, D., 17th National Conference on Artificial Intelligence, Austin (2000), 879.

[64] Whitley, D., Information and Software Technology, 43 (2001) 817.

[65] We use a quadratic approximation of the fitness, because establishing this approximation and its optimum is easily tractable for problems in up to 20 dimensions as in this work and because it is indeed appropriate to describe the local behavior of the fitness in the region of interest. Advanced methods are available for higher dimensions or for situations in which the time required by this analysis is no more negligible compared to that required for evaluating the fitness [14-16,19].

[66] Golub, G. and Kahan, W., J. Soc. Ind. Appl. Math. Ser. B Numer. Anal., 2 (1965) 205.