




# Genetic algorithm for multilayer shield optimization with a custom parallel computing architecture

F. Cordella<sup>1,2,a</sup> , M. Cappelli<sup>1</sup>, M. Ciotti<sup>1</sup>, G. Claps<sup>1,2</sup>, V. De Leo<sup>1,2</sup>, C. Mazzotta<sup>1</sup>, D. Pacella<sup>1,2</sup>, A. Tamburrino<sup>1,2,3</sup>, F. Panza<sup>1,4</sup>

<sup>1</sup> ENEA–Frascati Research Center, via E. Fermi 45, 00044 Frascati, Italy

<sup>2</sup> INFN–Frascati National Laboratories (LNF), via E. Fermi 40, 00044 Frascati, Italy

<sup>3</sup> DIAEE - Department of Astronautical, Electrical, and Energetic Engineering, “La Sapienza” University of Rome, Piazzale Aldo Moro 5, 00185 Rome, Italy

<sup>4</sup> INFN–Genoa, via Dodecaneso 33, 16146 Genova, Italy

Received: 6 October 2023 / Accepted: 26 December 2023

© The Author(s) 2024

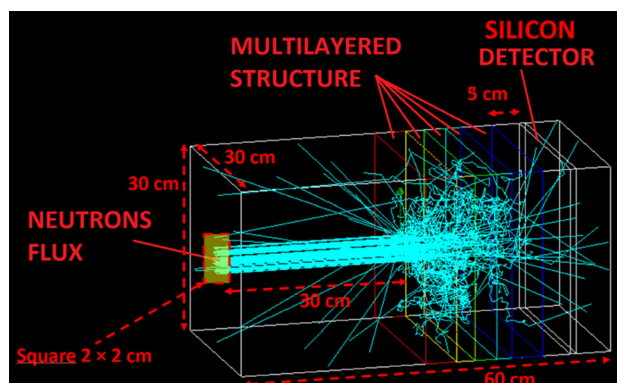
**Abstract** This paper introduces a novel architecture for optimizing radiation shielding using a genetic algorithm with dynamic penalties and a custom parallel computing architecture. A practical example focuses on minimizing the Total Ionizing Dose for a silicon slab, considering only the layer number and the total thickness (additional constraints, e.g., cost and density, can be easily added). Genetic algorithm coupled with Geant4 simulations in a custom parallel computing architecture demonstrates convergence for the Total Ionizing Dose values. To address genetic algorithm issues (premature convergence, not perfectly fitted search parameters), a Total Ionizing Dose Database Vault object was introduced to enhance search speed (data persistence) and to preserve all solutions’ details independently. The Total Ionizing Dose Database Vault analysis highlights boron carbide as the best material for the first layer for neutron shielding and high-Z material (e.g., Tungsten) for the last layers to stop secondary gammas. A validation point between Geant4 and MCNP was conducted for specific simulation conditions. The advantages of the custom parallel computing architecture introduced here, are discussed in terms of resilience, scalability, autonomy, flexibility, and efficiency, with the benefit of saving computational time. The proposed genetic algorithm-based approach optimizes radiation shielding materials and configurations efficiently benefiting space exploration, medical devices, nuclear facilities, radioactive sources, and radiogenic devices.

## 1 Introduction

Radiation shielding plays a fundamental role in many applications, dealing with ionizing radiations and radiogenic devices (e.g., optimized multilayer shields for satellite electronics [1], a novel thin radiation shield inspired by the morpho butterfly wing structure in medical applications [2], a genetic algorithm to optimize a tin-copper graded shielding system in plutonium safeguards measurements [3], safety considerations for the DTT device, addressing radiation fluxes, shielding, fueling, pumping, and cooling systems [4]). The optimization of radiation shielding structures poses a complex challenge due to the wide range of potential materials and configurations (sometimes only implicitly defined [5]). In other papers, like [1], the application is restricted to satellite applications; this work aims to provide a comprehensive architecture applicable to many different research areas instead. The presented test use case is dedicated to electronic component protection from neutrons in harsh environments, such as fusion nuclear facilities like ITER (the International Thermonuclear Experimental Reactor, [6]) and DTT (the Divertor Tokamak Test facility, [7]) where the neutron fluxes, in some electronics placement positions, can reach very high values. Moreover, shielding requirements for fusion reactors are complicated by geometry constraints that make the disposition of fully effective shielding a complex task [8]. For fission facilities, the electronics neutron shielding is usually less challenging than for fusion devices (some values of the neutron fluxes measured at different distances are reported in [9]), but if a closer to the core placement of the electronics is needed, also for fission plants the shielding may become necessary. The state-of-the-art neutron shielding materials for storing Spent Nuclear Fuel have been reviewed in a recent study [10] and for a recent review of the shielding design technology for nuclear reactors [11]. In particular, we are interested in minimizing the Total Ionizing Dose (TID) effect on electronic components while considering the constraints of the number of layers and the total thickness of the materials used. To this end, due to the unfeasibility of an exhaustive search approach (Sect. 3), it has been presented the utilization of a custom parallel computing architecture in Python for a genetic algorithm (GA)-based material selection approach to optimize multilayered structures for radiation shielding (for a GA selection of multi-materials, [5]). The GA, written from scratch without external libraries, is designed to optimize the shielding properties of

<sup>a</sup> e-mail: [francesco.cordella@enea.it](mailto:francesco.cordella@enea.it) (corresponding author)

**Fig. 1** Geant4 simulation of a beam of parallel 14 MeV neutrons (color aqua) impinging on a multilayer shield (from red to blue color) protecting a silicon slab (gray color) simulating a detector. The snapshot was drawn using the OpenGL library with 100 impinging neutrons



multilayered structures by minimizing the TID, while considering the number of layers and the total thickness of the materials used (on how they can be implemented, [12]). These layers can consist of light and heavy materials strategically incorporated to slow down and absorb neutrons within the shielding structure.

Monte Carlo simulations (Fig. 1) embedded in GA loops [13] were performed using the Geant4 (G4) toolkit, and the obtained results were validated through code comparison between G4 [14] and MCNP [15]. The decision to implement a genetic algorithm from scratch instead of using existing libraries can be attributed to several reasons. Firstly, implementing a genetic algorithm from scratch provides a deeper understanding of the algorithm's inner workings and allows for greater flexibility in customizing the algorithm to suit specific needs (for example, implementing a job to be recovered from dump files due to a sudden interruption). Secondly, existing libraries may not provide the necessary functionality required for the specific use case (for instance, the sub-process pipe execution of the Monte Carlo's code.exe file, the contextual modifications of the.txt input files or the dynamic writing of SQL queries). Finally, the implementation of a genetic algorithm from scratch can be more useful for an execution time investigation where for the particular Monte Carlo simulations setup used here, more than 99% of the computational time is dedicated to the Monte Carlo simulation itself. A GA is a strategy that guides heuristics to find adequate solutions to complex optimization problems, especially when computational resources are limited. It explores a subset of potential solutions that are too large to fully enumerate. However, GAs may not always find the optimal solution, but rather satisfactory ones.

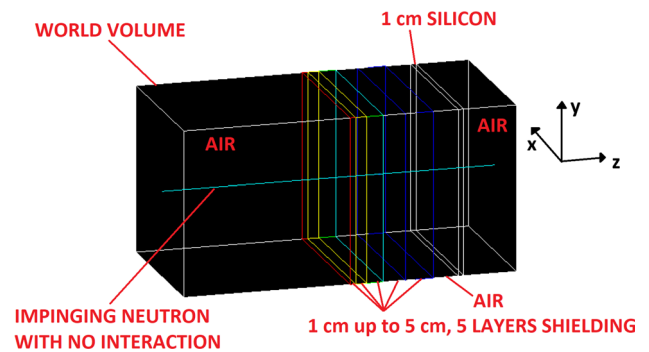
## 2 Monte Carlo simulations

Monte Carlo methods are a class of computational algorithms that rely on repeated random sampling to obtain numerical results. They are often used in physical and mathematical problems and are most valuable when using other approaches is difficult or impossible. Monte Carlo methods are mainly used in three problem classes: optimization, numerical integration, and generating draws from a probability distribution. Between them, G4 and MCNP are known for their versatility and predictive capability. In particular, the G4 toolkit is a versatile tool widely utilized in high-energy physics, medical physics, and radiation protection (for an overview of the functionality and models highlighting enhancements in performance, event biasing, and various modeling capabilities across diverse application domains, including high-energy physics, astrophysics, medical physics, and radiation protection, [16] and [17]). One can use the central limit theorem to compute uncertainties in Monte Carlo simulations. The central limit theorem states that the sum of a large number of independent random variables will tend toward a normal distribution, regardless of the distribution of the original variables. This means that the uncertainty in the simulation can be estimated by calculating the standard deviation of the results. This section presents the method used to simulate particle transport and investigate radiation shielding properties. The approach involves using the G4 toolkit and a comparison with MCNP code as a validation point.

### 2.1 G4 and MCNP libraries

G4, version 11.1.1, and MCNP, version 6.1, were used in this project. Specifically, the "Shielding" physics list was employed for G4 to optimize radiation shielding properties, while the ENDF/B-VII.0 library was used for MCNP. The "Shielding" physics list of G4 incorporates the G4ParticleHP module, which utilizes the G4NDL nuclear data libraries to simulate neutron interactions. The G4NDL libraries are a collection of nuclear data libraries used for simulating neutron transport below 20 MeV within the Geant4 particle physics simulation toolkit. These libraries are based on nuclear data written initially in the ENDF-6 (Evaluated Nuclear Data File) format, a standard format for representing nuclear data in various applications, including reactor physics, radiation shielding, and medical physics. JEFF-3.3 is the source of neutron cross-sections and final states, replacing the previous reliance on ENDF/B-VII.1, which was predominant until release 10.6, (more on G4 libraries in [18]). JEFF-3.3 nuclear data library is the nuclear data library that provides the best matching between the MCNP computational results and the experimental data. G4's physical processes and transport algorithms accurately simulated these interactions. The resulting energy deposition in different layers facilitated an

**Fig. 2** Geometry details used for the Monte Carlo simulations



efficient search for optimal material configurations using GA optimization. The “Shielding” physics list was a predefined reference for simulating radiation shielding.

## 2.2 Simulation setup

To ensure precise modeling and analysis of neutron interactions while maintaining simplicity and clarity, a monochromatic parallel neutron beam with an energy of 14 MeV was chosen as the input spectrum source. These neutrons were generated as primary particles, resulting in a single energy peak. The parallel beam configuration ensured consistent neutron momentum vectors and simplified the simulation by eliminating directional variations. The 14 MeV neutron is the energy of the neutrons resulting from the deuterium–tritium fusion reaction, and this energy is the reference energy for the fusion reaction and is used in the ITER and DEMO research thermonuclear facilities. This choice ensured precise modeling and analysis of neutron interactions while maintaining simplicity and clarity (of course, any input spectrum can be used). The sensitive detector was made of silicon (silicon detectors are commonly used in radiation detection due to their excellent energy resolution and high detection efficiency). The G4 code generated outputs as.txt files, including the TID, TID Uncertainty, and the percentage component contribution to the TID (optionally) in a specific.txt file. Other individual files were generated for each particle type passing through the detector, containing information on kinetic energy, energy loss, and particle count. The uncertainty in TID was calculated based on the standard deviation of energy deposits during the simulation.

## 2.3 Detector and source details

The detector geometry played a crucial role in the G4 simulation. It specified the dimensions and materials of the layers in the multilayered structure. The configuration of the layers was read from a configuration.txt file and modified after each simulation based on the GA optimization. About the computational times, in the case of 500,000 impinging neutrons for each simulation run, the measured time for each step of the for loop where the Monte Carlo.exe is called is more than 99% of the total computational time of the step (in the less than 1% there is the rest of the Python script, which includes the reading and modifying section of the input.txt file for the.exe). The world volume, representing the overall dimensions of the simulation environment, was defined as a box-shaped volume (with dimensions of 30 cm × 30 cm × 60 cm) filled with air (Fig. 2).

The layer configuration was read from a.txt file and updated after each simulation based on the GA optimization. To keep the number of simulations performed within an acceptable range, the material configuration list consisted of 11 commercially available materials, each with 5 possible thicknesses (Table 1). This particular material selection and thicknesses are, of course, just an example of what could be an initial material list for a possible multilayered structure with a total thickness to be used on a tabletop experiment. Each material’s minimum and maximum thicknesses and the step size for thickness variation were specified. For materials until Al of Table 1, testing them with a thickness lower than 2.0 cm was useless since the number of impinging neutrons would be too high to appreciate. For materials with a high atomic number, from Al on Table 1, their density prevented using more than 1.0 cm for a practical tabletop experiment.

The layers were positioned sequentially along the z-axis of the world volume, with different materials, thicknesses, and simulation colors assigned based on their order. This arrangement facilitated the formation of multilayered structures. The sensitive detector, made of silicon, was a box-shaped volume with a thickness of 1 cm inside the volume box-shaped. It was positioned 5 cm away from the last layer along the z-axis of the world volume. The neutron source emitted monochromatic neutrons with an energy of 14 MeV. It had a square shape with dimensions of 2 cm × 2 cm. The z-coordinate of the source was fixed at − 30 cm, ensuring a consistent source position throughout the simulations. The primary particles emitted from the source had their momentum toward the positive z-axis. They were emitted parallel to the z-axis, ensuring a uniform direction of particle propagation.

**Table 1** Material configuration list used for the multilayer shielding analysis

Material	Min thickness (cm)	Max thickness (cm)	Step (cm)	Density [g/cm <sup>3</sup> ]
B4C	2.0	10.0	2.0	2.52
LiF	2.0	10.0	2.0	2.64
Paraffin	2.0	10.0	2.0	0.93
Polyethylene	2.0	10.0	2.0	0.94
Borotron	2.0	10.0	2.0	1.00
Al	2.0	10.0	2.0	2.70
Ta	0.2	1.0	0.2	16.65
W	0.2	1.0	0.2	19.3
Pb	0.2	1.0	0.2	11.35
Cu	0.2	1.0	0.2	8.96
Fe	0.2	1.0	0.2	7.87

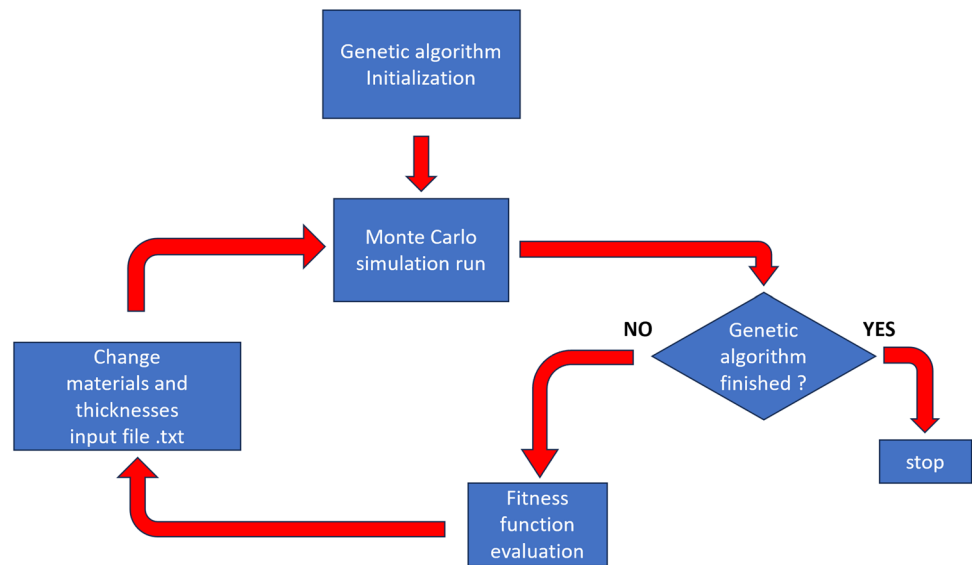
### 3 Brute force and dictionary attack approach

Using computer security terminology, the brute-force approach can examine and evaluate all possible combinations of materials and thicknesses to find optimal solutions through an exhaustive search. While this approach may be suitable for more straightforward problems, it becomes impractical and inefficient when dealing with complex problems, such as determining the optimal configuration of multilayer structures. The huge number of possibilities makes it infeasible to search through all combinations, even for relatively small structures, due to the NP-hard mathematical nature of the problem (there is no known algorithm that can find the best solution in polynomial time, and it is unlikely that such an algorithm exists). The dictionary attack, instead, uses a pre-compiled list of commonly used materials (here, in particular, the material configuration list. For the original information security concepts [19]). Their combination provides a practical approach to the complexity of the problem. To illustrate the overwhelming complexity, consider a 5-layer structure with a total thickness of 10 cm. By utilizing the Cartesian product to combine materials and thicknesses for each layer, the total number of combinations can be calculated as the product of the number of materials ( $n$ ) and the number of thicknesses ( $m$ ), raised to the power of the number of layers ( $L$ ), resulting in  $(n \times m)^L$ . Hence, with a dictionary of 11 materials and 5 thicknesses each, the number of possible combinations for 5 layers would be 503,284,375 (as stated before, this example focuses on a tabletop experiment to illustrate the method's validity. The number, type, and step of the thicknesses can be easily adapted to many different situations). However, it is essential to note that not all of these combinations are valid, as "duplicates" may frequently exist. Here, duplicates refer to combinations of materials and thicknesses that are essentially the same, and that can be merged. For instance, a combination of  $x$  or  $y$  cm of material A, immediately followed by  $y$  or  $x$  cm of material A, is equivalent to one combination of  $(x + y)$  cm of material A. These are considered "merged solutions". By identifying and merging these duplicates, it is possible to significantly reduce the number of combinations that need to be evaluated. Nonetheless, even after eliminating duplicates, the number of valid combinations in this case still reduces to 2,908,296. Despite a small dictionary and this significant reduction, the computational complexity of examining such vast possibilities remains prohibitively high and impractical. For instance, assuming a simulation time of 1 min per simulation, it would take over 5.5 years to complete the simulations of the valid combinations using the brute-force approach. Therefore, alternative methods, such as the GA optimization technique, are essential for efficiently exploring the solution space and finding near-optimal solutions within a reasonable timeframe (a recent review of GA can be found in [20]).

### 4 Optimization constraints

The proposed algorithm incorporates various constraints related to the number of layers and the total thickness of the multilayered structure. These constraints are essential for exploring complex structures to enhance shielding performance while ensuring physical feasibility. The constraint on the maximum number of layers is particularly stringent to prevent excessive complexity in shield implementation. In the initial stages of the algorithm, the constraints regarding the maximum total thickness and the number of layers are not strictly enforced. Instead, these constraints gain importance gradually as the algorithm progresses through successive generations. The algorithm adjusts the penalty weights based on the evolving population and search process by incorporating dynamic penalties. This dynamic nature allows the algorithm to balance exploration and exploitation, facilitating improved convergence toward optimal or near-optimal solutions. In the proposed algorithm, these penalty factors start at an initial value and increase linearly over the generations until they reach a final value. The current generation number and the maximum number of generations determine the rate of increase. The dynamic penalty factors are used to calculate penalties for the number of layers and the total thickness. These penalties are subtracted from the fitness score, so solutions with more layers or greater total thickness will have lower fitness scores. This approach allows the genetic algorithm to explore a wide range of solutions in the early generations, while gradually

**Fig. 3** General flowchart of the genetic algorithm proposed with the embedded Monte Carlo simulation proposed



favoring solutions with fewer layers and smaller total thickness in the later generations. This strategy guides the search for feasible and optimal solutions.

### 5 Genetic algorithm

The general flowchart of the proposed algorithm working together with the Monte Carlo simulation is summarized in Fig. 3.

GAs belong to the class of optimization algorithms that draw inspiration from natural selection. Python was used to implement the specific GA. GAs are categorized as metaheuristic algorithms alongside well-known approaches such as Simulated Annealing, Ant Colony Optimization, and Particle Swarm Optimization [20]. These algorithms utilize a population of individuals, represented as chromosomes, to generate multiple potential solutions for a given problem. In this work, the genes are the layers, thicknesses, and material types. These three components together form the chromosomes. Each chromosome represents a potential solution to the problem, and the genetic algorithm evolves these chromosomes over generations. GAs strive to balance exploration and exploitation during the problem-solving process: Exploration refers to generating and evaluating new potential solutions by combining and modifying existing ones. Exploitation refers to selecting and preserving the best solutions and focusing the search around them. A good balance between exploration and exploitation is essential for finding “optimal” or near-optimal solutions efficiently (with optimal, we are referring to the solution that best satisfies the objective function of the problem at hand. In this case, this means minimizing the TID with the minimum number of layers and the total thickness). Too much exploration may lead to a slow convergence or getting lost in irrelevant regions of the solution space. Too much exploitation may lead to a premature convergence or getting trapped in local optima.

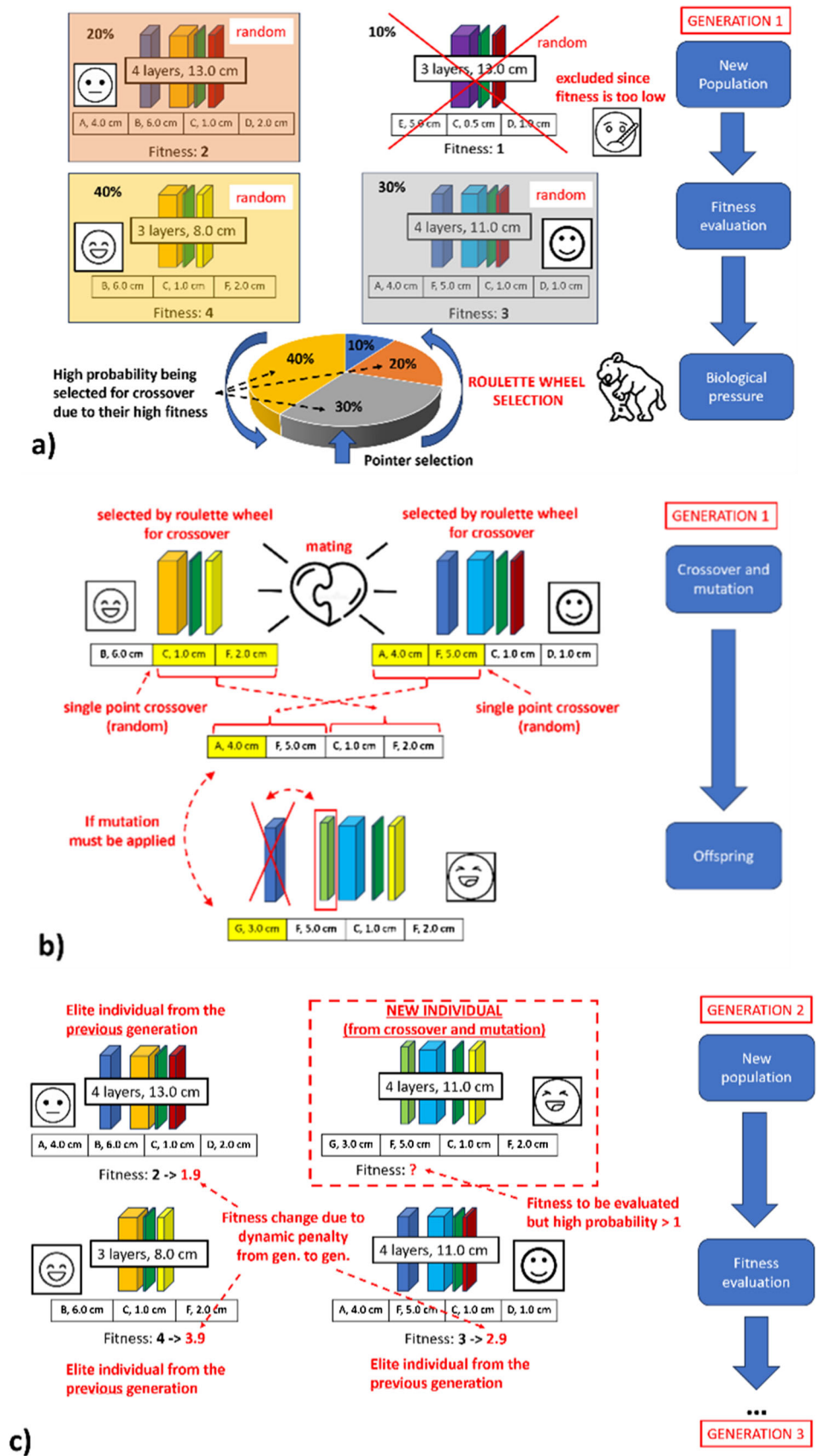
Initially proposed by John Holland in 1975 and collaborators ([21, 22]), GAs draw inspiration from natural selection and genetics, precisely the concept of “survival of the fittest.” They find applications in various domains, such as soft computing, machine learning, and operations research, particularly for search and optimization problems. In a GA, the population undergoes iterative evolution over generations through three fundamental operations: selection, crossover, and mutation (Fig. 4a–c). The survival of the fittest concept is embodied by the fitness value, representing the figure of merit for the GA. The fitness of each individual in the population is evaluated considering the TID, the number of layers, and the total thickness of the material list in the *fitness* variable evaluation (the minus sign in the various addenda is indicative of a search for a maximum):

$$\text{fitness} = -\text{TID} \times \text{tid}_w - \text{layer}_p \times \text{layer}_w - \text{thickn}_p \times \text{thickn}_w + \dots$$

where  $\text{layer}_p$  and  $\text{thickn}_p$  are the dynamic penalties and  $\text{tid}_w$ ,  $\text{layer}_w$ , and  $\text{thickn}_w$  are the weights of the constraints. Of course, it is possible to add other constraints, such as materials cost, density, etc., depending on the context.

Selection in GAs involves choosing individuals from the current population to serve as parents for the next generation. Through crossover and mutation techniques, new individuals are created and added to the population. Each iteration in a GA is referred to as a generation. In each generation, individuals are evaluated based on their fitness to solve the problem, and a new population of candidate solutions is formed using biologically-inspired operators like mutation, crossover, and selection. In GAs, chromosomes are typically represented as strings, with each gene or character assigned a value. The order of genes on a chromosome, known as loci, plays a crucial role. GAs explore the solution space by manipulating the chromosomes and strive to converge toward optimal solutions. The main script of our algorithm begins by initializing a population of material lists, where each material list represents

**Fig. 4** The main steps of the GA here, implemented: **a** generation1: starting of a new population with randomly selected individuals, fitness evaluation, and roulette wheel selection for crossover, **b** generation1: single-point crossover and mutation (if necessary), and **c** generation2: elite individuals with updated dynamic penalty fitness value, new individual addition and creation of new population



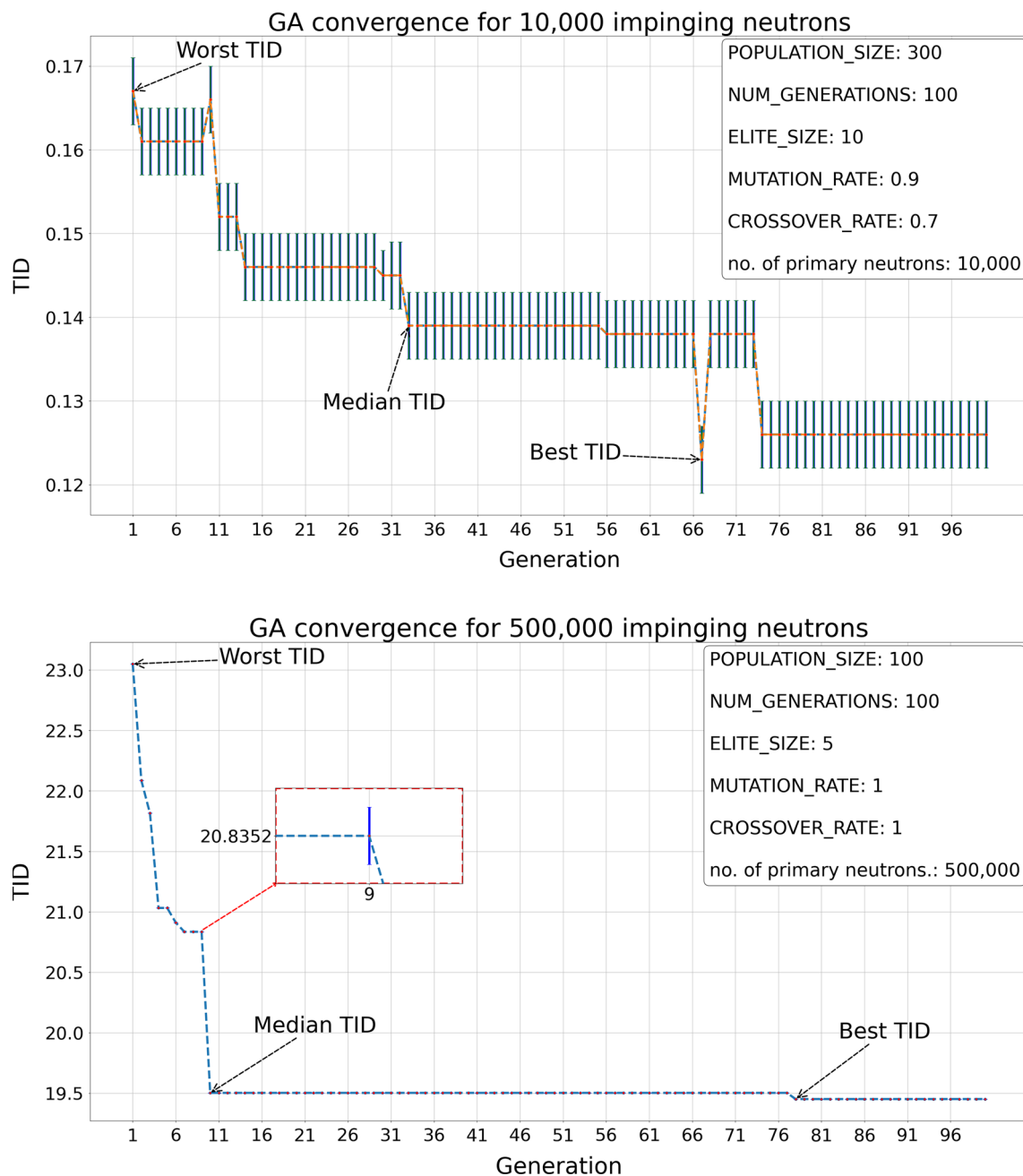
a potential solution and consists of random layers and thicknesses. Elite individuals are selected based on their *fitness* values, with the ELITE\_SIZE parameter determining the number of top performers to be preserved.

The choice of ELITE\_SIZE can significantly impact the performance of the GA. As for other parameters, finding the correct value is often a matter of trial and error. Generally, if ELITE\_SIZE is too large, the GA may become too greedy and converge prematurely to a local optimum. If ELITE\_SIZE is too tiny, the GA may lose good solutions and take longer to converge. ELITE\_SIZE is often set to 10% or 20% of the population because this range typically balances exploration and exploitation. It allows the GA to preserve a reasonable number of good solutions without overly biasing the search for these solutions. The population is updated in each generation by creating offspring using single-point crossover and mutation operations. Single-point crossover involves selecting a random point or position in the genetic material (in this case, the material list) of two parent individuals. The genetic information beyond that point is swapped between the parents, resulting in two new offspring individuals. On the other hand, mutation randomly selects a layer in the material list and modifies its material or thickness using the mutation function. The fitness score calculation incorporates a dynamic penalty factor ([23]) that initially promotes exploring a broader range of material lists. A penalty function is employed in the search space to enforce thickness and material properties constraints. This function calculates penalty values based on the degree of a constraint violation, penalizing exceeding the target number of initial and final layers and the maximum allowable thickness. By increasing penalties for more significant violations, the algorithm avoids non-compliant individuals and converges toward optimal solutions that meet the required standards. This approach enhances reliability and ensures adherence to constraints. The selection method used here, is the roulette wheel selection or proportionate fitness selection, a commonly used method for selecting individuals from a population for reproduction. The roulette wheel selection works by assigning a portion of the wheel's circumference to each individual in the population based on their relative fitness. The size of each portion is proportional to the individual's fitness value compared to the total fitness of the population. During the selection process, a spinning wheel metaphor is used, where a pointer randomly stops at various positions on the wheel. The probability of the pointer stopping at a particular position is directly proportional to the fitness proportion of the individual associated with that position. The performance and efficiency of a GA depend on critical parameters such as population size, number of generations, elite size, mutation rate, and crossover rate. Larger populations and more generations generally allow for better solutions but also increase computational complexity. The chosen values for population size, number of generations, and elite size strike a balance between the level of optimization and computational cost. For example, a high value for population size promotes diversity and thorough exploration. The elite size parameter retains high-quality solutions while allowing for diversity within the population. A sufficient number of generations is selected to ensure convergence within a reasonable timeframe. The mutation and crossover rates influence the algorithm's exploration and exploitation capabilities.

### 5.1 GA parameters fine-tuning

The parameters fine-tuning for a new run is an essential step that can be applied from the results and insights of the previous runs (for the choice of parameters, [24]). The goal is to optimize the performance of the GA with a better-converging solution, avoiding unnecessary long computation times on the one hand and poor local minimum solutions on the other. To this end, insights can be gained from the previous attempts to explore new parameter values (if needed, it is also possible to implement a meta-optimization of the best parameters easily [25]). The following is an example of an analysis of multiple input results:

- **NUMBER OF IMPINGING PARTICLES:** With 10,000 impinging particles, it is possible to obtain a TID uncertainty  $\sigma_{\text{TID}}$  of approximately  $5 \times 10^{-7}$  Gy/neutron and a computing time of around 50/60 s, depending on the particular configuration with an Intel i7 CPU @ 2.60GHz with 16 GB RAM. When using 500,000 particles, the  $\sigma_{\text{TID}}$  reduces to nearly  $10^{-12}$  Gy/neutron, but with a much longer computing time (on the same computer) of approximately 1000/1400 s, always depending on the configuration of the layers to be simulated. All the results in this work were conducted with 500,000 particles.
- **POPULATION\_SIZE:** A larger population can help explore a larger search space with a drawback in long computation time. Setting this value between 100 and 300 is advisable based on trial and error. All the results in this work were conducted with a POPULATION\_SIZE = 100, 150, 200, and 300 individuals.
- **NUM\_GENERATIONS:** Using 100 for this value often gives a useless convergence tail. From trial and error, reducing it to around 70 or 80 is possible. All the results in this work were conducted with a NUM\_GENERATIONS = 100.
- **ELITE\_SIZE:** To preserve good solutions from generation to generation, the ELITE\_SIZE parameter can be set to around 10–20% of the population size. All the results in this work were conducted with an ELITE\_SIZE = 5, 10, 15, and 20%.
- **MUTATION\_RATE:** A moderate mutation rate can help introduce diversity into the population. A MUTATION\_RATE parameter around 0.05–0.1 is advisable based on a trial and error method. All the results in this work were conducted with a MUTATION\_RATE = 0.7, 0.8, 0.9, and 1.0.
- **CROSSOVER\_RATE:** The CROSSOVER\_RATE parameter determines the likelihood of genetic material exchange between individuals. Based on a trial and error method, a CROSSOVER\_RATE parameter around 0.8 can be set. All the results in this work were conducted with a CROSSOVER\_RATE = 0.7, 0.8, 0.9, and 1.0.



**Fig. 5** Single node results for the GA, with 10,000 (upper) and 500,000 (bottom) impinging neutrons with different GA parameters and convergence behaviors. On the y-axis, the TID is expressed in MeV/g. The standard deviation is represented by the blue bar. The best (lowest) TID is reached at generation 67 (upper) and at generation 78 (bottom). The inset on the right graph (bottom) shows that the TID uncertainty for 500,000 impinging neutrons is negligible

## 6 Single node results

Running the algorithm on a single node of the custom parallel computing architecture (Sect. 9), with a simulation having a number of primary particles equal to 10,000 or 500,000 impinging neutrons yields the results shown in Fig. 5 (the error bars represent the TID uncertainties corresponding to 1 standard deviation) where the convergence to a (sub) optimal solution is evident:

As shown in Fig. 5, for 10,000 and 500,000 impinging neutrons, the algorithm initially demonstrates a rapid decrease in the TID values, indicating a significant improvement in solution quality during the early iterations. This initial phase often involves the algorithm exploring a wide range of candidate solutions in search of the global optimum. The algorithm converges toward a particular search space region as the iterations proceed. As the plots show, the convergence rate gradually slows down ([24] for more on convergence analysis). The best TID result is reached in generation no. 67 for the 10,000 impinging neutrons (TID: 0.123 MeV/g,



fitness:  $-0.2003$ ), but later, it was abandoned by the algorithm due to its high number of layers and total thickness (hence giving a poor fitness value) until generation 100 (TID:  $0.126$  MeV/g, fitness:  $-0.2415$ ). Instead, for 500,000 impinging neutrons, the best TID results with fitness value were reached at generation 78 (TID:  $19.4524$  MeV/g, fitness:  $-13.76$ ), and then it was maintained until generation 100 (TID:  $19.4524$  MeV/g, fitness:  $-13.8$ ). Local minima represent a challenge for optimization algorithms, as they may get trapped in suboptimal solutions. Hence, it is crucial to carefully analyze the characteristics of the fitness landscape to devise strategies that overcome these local minima and guide the algorithm toward the global optimum.

The GA prefers higher fitness to lower TID, but the fitness is not only determined by the TID but also by the number of layers and the total thickness of the shield. These constraints are enforced by dynamic penalties that increase over the generations. Therefore, the GA tries to find a trade-off between minimizing the TID and satisfying the constraints. As humans, we would solve this trade-off by considering the specific application and context of the shield. For example, if we have a limited space and budget, we would prefer a shield with fewer layers and smaller thickness, even if it has a slightly higher TID.

On the other hand, if we have more flexibility and resources, we would choose a shield with lower TID, even if it has more layers and a larger thickness. The lowest TID point at 67 generations, for 10,000 impinging neutrons, is a local minimum, not a global minimum. This is because the GA abandoned this solution in the later generations due to its high number of layers (5) and total thickness (9.8 cm), which violated the constraints and resulted in a low fitness score. The GA converged to a different solution with fewer layers (3) and smaller thickness (6.4 cm), which had a higher fitness score, even though it had a slightly higher TID. This solution is closer to the global minimum, balancing the TID and the constraints better than the solution at 67 generations. The lowest TID point at 67 generations, for 10,000 impinging neutrons, was reached at 67% of the total computation time (nearly 3 days of computational time), and the lowest TID point at 78 generations, for 500,000 impinging neutrons, was reached at the 78% of the total computation time (nearly 15 days of computational time).

## 7 Total Ionizing Dose Database Vault

The Total Ionizing Dose Database Vault (TDV) concept (something similar but more general to the “Hall Of Fame” in [26]) is an important new aspect of the current method. It involves using a local Database (here, SQLite was used, but other DBs with concurrent access features are possible) that is shared and synchronized with other cluster nodes. We chose SQLite because it is a lightweight database system that does not require a separate server process, which makes it easy to use and manage. However, implementing the TDV directly into Python using the Pandas library is a viable alternative. However, one thing to consider is that while Pandas is excellent for data analysis, it might not be as efficient as SQLite for data storage and retrieval, especially for large datasets. SQLite databases are stored on disk, which allows them to handle larger datasets than Pandas, which stores data in memory.

In conclusion, SQLite and Pandas have their strengths and weaknesses, and choosing between them depends on the project’s specific requirements. For this study, we found SQLite to be a suitable choice. The purpose of the TDV is to preserve the individuals from the population based on their performance within the given constraints, even if they were lost during the evolution process due to selection, crossover, or mutation. By employing standard SQL queries, the TDV can be sorted at any time, allowing the retrieval of the individual that had the particular TID value (or range of values) ever recorded. In addition to preserving individuals, the TDV concept is a valuable knowledge treasure. It can be consulted anytime, even during the code execution process. Before conducting a simulation, the Python code checks the TDV to determine if a specific simulation has already been performed on any of the computers in the cluster in the past runs. This feature avoids (quite perfectly) redundant simulations, saving a lot of computational resources and time. By leveraging the TDV’s consultable nature, researchers can efficiently utilize existing knowledge and optimize their simulation processes within the cluster. An example of the use of the TDV is provided in Table 2, showcasing the ten overall best and worst results obtained from running the GA for over 43,000 simulations conducted on five computers in a custom parallel architecture (Sect. 9). Each simulation run consisted of 500,000 impinging neutrons. In the last column of Table 2, the shielding effectiveness (SE) is introduced, defined as the ratio between TID and  $TID_0$ , where TID and  $TID_0$ , respectively, represent the TID mitigated by the multilayer shield and the unmitigated TID (i.e., only air with the same thickness as the multilayer shield. For a photon analogy, [27], par. 4.7.4).

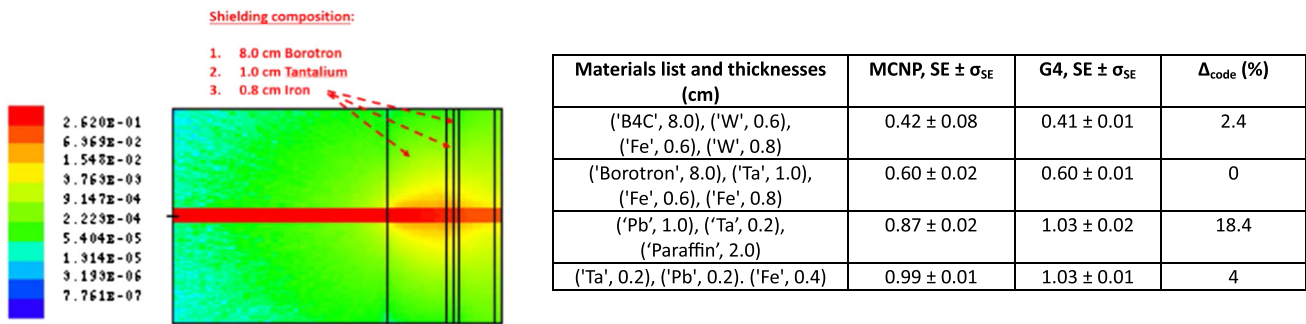
In Table 2, the best multilayer shield is represented by  $ID = 1$ ; it can reduce, to nearly 38%, the TID on the silicon slab if compared to the same thickness layer of air (9.8 cm). This is achieved by using five layers with four different materials, with  $B_4C$  as the first layer facing the incoming flux of neutrons and Tungsten as the last layer to stop gammas. From Table 2, it is possible to get the following general insights into the choice and arrangement of materials for the best neutron shielding:

- Boron carbide ( $B_4C$ ) consistently appears in the most optimal configurations between the first layers.  $B_4C$  is known for its high thermal neutron capture cross-section, effectively attenuating neutron energy and making it a popular choice for neutron shielding.
- Tungsten (W) and Tantalum (Ta) are frequently selected materials between the last layers. These high-density metals effectively shield against gamma radiation and contribute to neutron absorption properties, enhancing overall shielding performance.
- Copper (Cu) and Iron (Fe) are also included in some configurations, mainly in the middle layers. These materials, with moderate densities, offer additional gamma radiation attenuation while contributing to neutron absorption to a lesser extent than  $B_4C$ .

**Table 2** TDV's first and last results using 500,000 impinging neutrons running the GA for more than 43,000 simulations on (up to) five computers in a custom parallel computing architecture

ID	Materials list with their thicknesses [cm]	No. of layers	Total thickness [cm]	TID ± σ TID [MeV/g]	TID <sub>0</sub> ± σ TID <sub>0</sub> [MeV/g]	SE ± σ SE
1	('G4_B4C', 6.0), ('G4_Fe', 1.0), ('G4_W', 1.0), ('G4_Ta', 0.8), ('G4_W', 1.0)	5	9.8	17.35 ± 6E-07	45.7078 ± 7E-07	0.3795 ± 0.0015
2	('G4_B4C', 6.0), ('G4_Ta', 0.8), ('G4_W', 1.0), ('G4_Ta', 0.8), ('G4_W', 1.0)	5	9.6	17.6864 ± 6E-07	46.1305 ± 7E-07	0.3832 ± 0.0015
3	('G4_B4C', 6.0), ('G4_W', 1.0), ('G4_Fe', 0.8), ('G4_Ta', 0.8), ('G4_W', 1.0)	5	9.6	18.0851 ± 6E-07	46.1305 ± 7E-07	0.3918 ± 0.0013
4	('G4_Ta', 1.0), ('G4_W', 1.0), ('G4_B4C', 6.0), ('G4_Ta', 1.0), ('G4_W', 0.6)	5	9.6	18.2299 ± 6E-07	46.1305 ± 7E-07	0.3950 ± 0.0013
5	('G4_B4C', 6.0), ('G4_Fe', 1.0), ('G4_W', 1.0), ('G4_Cu', 0.8), ('G4_W', 1.0)	5	9.8	18.3141 ± 6E-07	45.7078 ± 7E-07	0.4008 ± 0.0013
...	...	...	...	...	...	...
43775	('G4_Pb', 0.4), ('G4_PARAFFIN', 2.0)	2	2.4	48.2037 ± 6E-07	45.7147 ± 7E-07	1.055 ± 0.013
43777	('G4_Pb', 0.4), ('G4_POLYETHYLENE', 2.0)	2	2.4	48.2282 ± 6E-07	45.7147 ± 7E-07	1.055 ± 0.013
43778	('G4_Pb', 0.2), ('G4_PARAFFIN', 2.0)	2	2.2	48.3121 ± 6E-07	46.1412 ± 7E-07	1.047 ± 0.013
43779	[('G4_POLYETHYLENE', 2.0)]	1	2.0	48.3359 ± 6E-07	45.8293 ± 7E-07	1.055 ± 0.013

The TID values are expressed in MeV/g here, and the table is sorted by TID (ascending). SE < 1 means an effective shield, SE ~ 1 is equivalent to having no shield, and SE > 1 is worse than having no shield. σ TID, σ TID<sub>0</sub>, and σ SE are the standard deviation of TID, TID<sub>0</sub>, and their ratio SE (the propagation of uncertainty for the ratio SE = TID/TID<sub>0</sub> was used)



**Fig. 6** left: MCNP flux map for a particular multilayer shielding ('Borotron', 8.0), ('Ta', 1.0), and ('Fe', 0.8), right: comparison table between MCNP and G4 (with 100,000 impinging neutrons)

Overall, the observed trends emphasize the importance of effectively employing a combination of materials with diverse properties to shield against neutron and gamma radiation (e.g., [12]). The real selection and arrangement of materials should also consider other factors such as available space, cost considerations, and specific application requirements. It is crucial to note that the suitability of a shielding configuration can vary depending on the radiation sources, energies involved, and the context of the application. Moreover, the SE analysis suggests that sometimes “nothing is better than something” since SE ~ 1 or higher means a factor of the TID received comparable to air, and it would be useless (if not self-defeating) a bad multilayer shield. One thing to note is that smaller uncertainties are obtained by increasing the number of impinging neutrons in the simulation (for more information on Monte Carlo code uncertainties, refer to [28]). These results highlight the trade-off between shield performance (TID) and shield design (number of layers and total thickness). Different shield configurations might be optimal depending on the application’s specific requirements. For example, if space and weight are not a concern, a configuration with more layers and a larger total thickness might be preferred for its superior radiation protection. However, if space and weight are limited, a simpler and smaller configuration might be more appropriate, even if it offers less radiation protection.

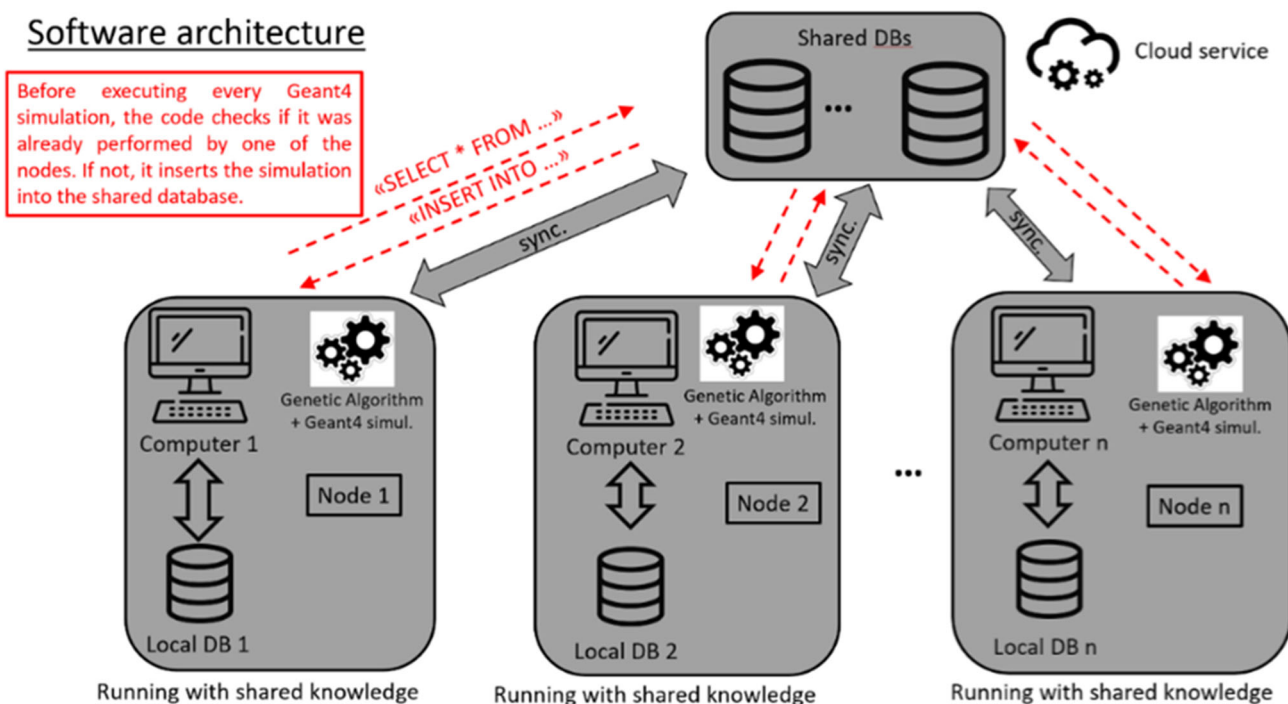
### 8 G4/MCNP code comparison

To establish a validation point using a comparison with another Monte Carlo code, specific simulations were conducted with G4 v. 11.1.1 and MCNP v. 6.1, equipped with ENDF/B-VII.0 libraries (for details on some comparison issues, reference [29]). These simulations were carried out for low and high values of SE. The G4 simulations were conducted this time with 100,000 impinging neutrons; instead, the MCNP simulations (Fig. 6, left) were conducted until the percentage error was satisfying (lower than 1%). The setup of the MCNP simulations was identical to the G4 one.

The comparison table (Fig. 6, right) shows that the Monte Carlo code comparison between G4 and MCNP was excellent for low SE values and not for SE around 1. The reason probably relies on the fact that when much secondary radiation is produced, the differences between the codes (libraries, models) are more likely to emerge. Since the focus is on low SE values, it can be stated that the percentage difference (Δ<sub>code</sub>) between G4 and MCNP is comparable to the percentage uncertainties (σ<sub>SE</sub>) on SE (Table 2). Hence, the choice of which Monte Carlo code to use in the GA for a fast evaluation of the SE is irrelevant for low SE values.

### 9 Custom parallel computing architecture

It was already mentioned (Sect. 5.1) the importance of conducting multiple runs with different parameter values to reduce the risk of being stuck in local minima. Moreover, to enhance the computation speed, a custom distributed system with synchronization capabilities was used, employing cloud service and local databases (for distributed systems architecture, refer to [30] and [31]). A custom parallel computing architecture (Fig. 7) can be leveraged in radiation shielding simulations, and rather than relying on existing libraries or computer architectures, the current approach aims to make these calculations accessible to all in the simplest possible way. This custom approach utilizes multiple computers with processors and local databases connected through a cloud service. The Python script responsible for the simulations runs simultaneously on each computer, together with its own TDV database, within the distributed system. Before running a new simulation, the script checks if the same simulation was already done by one of the nodes of the cluster, and, in this case, it recovers the TID data without performing the same simulation again. This setup enhances computation speed (avoiding already done simulations) and mitigates the risk of being stuck in local minima by conducting multiple runs with different parameter search values and persisting the relevant data on the synchronized TDV (shared knowledge). A cloud service is here, employed as a central hub to allow synchronization and data sharing. The results generated from each computer’s local TDV database are shared and consolidated through this cloud service. This custom parallel computing architecture optimizes



**Fig. 7** Software architecture schema with node, cloud service, and synchronized local TDV databases (shared knowledge)

the efficiency and effectiveness of radiation shielding simulations while maintaining simplicity and accessibility for all users. A potential flaw of the proposed approach is missing the global minimum if all the nodes converge to a local minimum. This is a possibility if the probability of this happening is low. Moreover, some strategies can be used to mitigate this risk, such as restarting the algorithm on each node multiple times (with different populations, generations, and elite size values), increasing the diversity (the algorithm can maintain a diverse population by using different selection, crossover, and mutation operators). This can prevent premature convergence and increase the chances of finding the global minimum.

The advantages of this architecture are:

- **Resilience:** If one node is turned off or crashes, all the other nodes continue working without interruption.
- **Scalability:** If a new node is added to the network, all the existing nodes continue working without issue, and the computation speed is increased due to the addition of the new node that with the database synchronization, will increase the number of simulations already available for all the nodes.
- **Autonomy:** If the cloud service experiences downtime, all individual nodes continue operating in a standalone configuration.
- **Flexibility:** Each node can have distinct GA settings, such as population size, generation number, and elite size.
- **Efficiency:** The searched TID value is first checked if it is already present in the local TDV database (synchronized with all the other nodes) without performing the time-consuming simulation. Before synchronization happens, there is a small probability that the same simulation will be conducted again. Analyzing a single TDV database to identify the best configuration immediately without waiting for the algorithm to complete on all nodes is also possible.

## 10 General conclusion

This work presented a GA-based approach for optimizing radiation shielding materials and their configurations. The algorithm effectively minimizes the TID on a silicon slab, while considering the number of layers and the total thickness of materials as constraints. Dynamic penalties and roulette wheel selection enhance the algorithm's performance, and the GA-based approach proves to be efficient and robust in solving complex optimization problems, reducing the number of simulations needed. The parameter tuning is crucial to ensure optimal performance. Parameters such as population size, crossover, mutation rates, and selection probabilities require careful consideration here, found through a trial and error process (but a meta-optimization of the parameters is possible). A comparison between G4 v. 11.1.1 and MCNP v. 6.1 was conducted with good agreement for low SE values. A custom distributed parallel computing architecture here, offers immense computation speed benefits in large-scale, complex simulations. A "TID Database Vault" entity was introduced to store the TID results obtained during optimization (consultable at any moment). This repository of material configurations enables researchers to leverage custom parallel computing for faster simulations,

avoiding repeating already done simulations. After more than 43,000 simulations, for impinging 14 MeV parallel neutrons, the best multilayer shield with the constraints of a maximum 10.0 cm thickness and a max of 5 layers was found to be: (1, B<sub>4</sub>C, 6.0 cm), (2, Fe, 1.0 cm), (3, W, 1.0 cm), (4, Ta, 0.8 cm), and (5, W, 1.0 cm). This configuration can reduce, to nearly 38%, the TID on the silicon slab if compared to the same thickness layer of air (9.8 cm). In summary, combining the GA-based approach, the TDV concept, and custom parallel computing architecture enables accelerated and comprehensive optimization of radiation shielding materials. This approach has broad applications in space exploration, medical devices, nuclear facilities, and other areas where efficient and effective radiation shielding is vital for safety and performance.

**Funding** Open access funding provided by Ente per le Nuove Tecnologie, l'Energia e l'Ambiente within the CRUI-CARE Agreement.

**Code availability** All the codes used in the manuscript are available and were written in Python for the GA part, in C++ for the Geant4 simulation, and in Fortran for the MCNP simulations.

**Data Availability Statement** This manuscript has associated data in a data repository. [Authors' comment: The article's data are available as an SQLite database from the corresponding author upon reasonable request.]

## Declarations

**Conflict of interest** The authors declare no competing interests.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. H. Daneshvar, K.G. Milan, A. Sadr, S.H. Sedighy, S. Malekie, A. Mosayebi, Multi-layer radiation shield for satellite electronic components protection. *Nat. Sci. Rep.* **11**, 20657 (2021). <https://doi.org/10.1038/s41598-021-99739-2>
2. S.-C. Kim, H. Byun, Development of ultra-thin radiation-shielding paper through nanofiber modeling of morpho butterfly wing structure. *Nat. Sci. Rep.* **12**, 22532 (2022). <https://doi.org/10.1038/s41598-022-27174-y>
3. M. Kleedtke, S. Hua, S. Pozzi, Genetic algorithm optimization of tin–copper graded shielding for improved plutonium safeguards measurements. *Nucl. Instrum. Methods Phys. Res. Sect. A Accel. Spectrom. Detect. Assoc. Equip.* **988**, 164877 (2021)
4. G. Mazzitelli et al., The DTT device: safety, fuelling and auxiliary system. *Fusion Eng. Des.* **122**, 375–381 (2017). <https://doi.org/10.1016/j.fusengdes.2017.05.131>
5. X.-J. Zhang, K.-Z. Chen, X.-A. Feng, Material selection using an improved genetic algorithm for material design of components made of a multiphase material. *Mater. Des.* **29**, 972–981 (2008). <https://doi.org/10.1016/j.matdes.2007.03.026>
6. ITER. <https://www.iter.org/>
7. DTT. <https://www.dtt-project.it/>
8. R.T. Santoro, Radiation shielding for fusion reactors. *J. Nucl. Sci. Technol.* **37**(sup1), 11–18 (2000). <https://doi.org/10.1080/00223131.2000.10874838>
9. Y. Oka, *Radiation shielding for fission reactors* (Atomic Energy Society of Japan, Japan, 2000)
10. X. Fu, Z. Ji, W. Lin, Y. Yu, T. Wu, The advancement of neutron shielding materials for the storage of spent nuclear fuel. *Sci. Technol. Nucl. Install.* **2021**, 5541047 (2021). <https://doi.org/10.1155/2021/5541047>
11. Y.Q. Chen, B.H. Yan, The technology of shielding design for nuclear reactor: a review. *Prog. Nucl. Energy* **161**, 104741 (2023). <https://doi.org/10.1016/j.pnucene.2023.104741>
12. B. Liu et al., Multi-objective optimization design of radiation shadow shield for space nuclear power with genetic algorithm. *Front. Energy Res.* **10**, 800930 (2022). <https://doi.org/10.3389/fenrg.2022.800930>
13. P. Panikkath, P.K. Sarkar, S. Krishnaswamy, A technique of solving an ill-posed inverse problem of neutron spectrum unfolding using a genetic algorithm search within Monte Carlo iterations. *Eur. Phys. J. Plus* **136**(4), 450 (2021). <https://doi.org/10.1140/epjp/s13360-021-01437-5>
14. Geant4. <https://geant4.web.cern.ch/>
15. MCNP. <https://mcnp.lanl.gov/>
16. J. Allison et al., Geant4 developments and applications. *IEEE Trans. Nucl. Sci.* (2006). <https://doi.org/10.1109/TNS.2006.869826>
17. S. Agostinelli et al., Geant4—a simulation toolkit. *Nucl. Instrum. Methods Phys. Res. Sect. A Accel. Spectrom. Detect. Assoc. Equip.* **506**(3), 250–303 (2003). [https://doi.org/10.1016/S0168-9002\(03\)01368-8](https://doi.org/10.1016/S0168-9002(03)01368-8)
18. Credit/Citations for Data Files distributed with Geant4. [https://geant4.web.cern.ch/download/data\\_files\\_citations](https://geant4.web.cern.ch/download/data_files_citations). Accessed 3 Jul 2023
19. I. Alkhwaja et al., Password cracking with brute force algorithm and dictionary attack using parallel programming. *Appl. Sci.* **13**(10), 5979 (2023). <https://doi.org/10.3390/app13105979>
20. S. Katoch, S.S. Chauhan, V. Kumar, A review on genetic algorithm: past, present, and future. *Multimed. Tools Appl.* **80**(5), 8091–8126 (2021). <https://doi.org/10.1007/s11042-020-10139-6>
21. K.A. De Jong, *An analysis of the behavior of a class of genetic adaptive systems* (University of Michigan, Michigan, 1975)
22. J.H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence* (MIT Press, Cambridge, 1992)

23. E.P. Dadios, J. Ashraf, Genetic algorithm with adaptive and dynamic penalty functions for the selection of cleaner production measures: a constrained optimization problem. *Clean Technol. Environ. Policy* **8**, 85–95 (2006)
24. X.-S. Yang, *Chapter 5–Genetic Algorithms*. In ed. by X.-S. B. T.-N.-I. O. A. Yang 77–87, (Elsevier, Oxford, 2014)
25. E. Skakov, V. Malysh, Parameter meta-optimization of metaheuristics of solving specific NP-hard facility location problem. *J. Phys. Conf. Ser.* **973**, 12063 (2018). <https://doi.org/10.1088/1742-6596/973/1/012063>
26. E. Wirransky, *Hands-On Genetic Algorithms with Python*. (Birmingham, 2020).
27. M. Cappelli, *Instrumentation and Control Systems for Nuclear Power Plants*. (Elsevier Science, 2023).
28. F. Renner, J. Wulff, R.-P. Kapsch, K. Zink, Uncertainties in Monte Carlo-based absorbed dose calculations for an experimental benchmark. *Phys. Med. Biol.* **60**, 7637–7653 (2015). <https://doi.org/10.1088/0031-9155/60/19/7637>
29. E. Mendoza, D. Cano-Ott, A. Ibarra, F. Mota, I. Podadera, Y. Qiu, S.P. Simakov, Nuclear data libraries for IFMIF-DONES neutronic calculations. *Nucl. Fusion* **62**, 106026 (2022). <https://doi.org/10.1088/1741-4326/ac814f>
30. A.S. Tanenbaum, M. van Steen, *Distributed systems: principles and paradigms*, 2nd edn. (Pearson Prentice Hall, New Jersey, 2007)
31. G. Coulouris, J. Dollimore, T. Kindberg, G. Blair, *Distributed systems: concepts and design*, 5th edn. (Addison-Wesley Publishing Company, New York, 2011)