symmetry

MDPI

Article

A Genetic Algorithm for Three-Dimensional Discrete Tomography

Elena Toscano and Cesare Valenti *

Dipartimento di Matematica e Informatica, Università degli Studi di Palermo, Via Archirafi 34, 90123 Palermo, Italy; elena.toscano@unipa.it
* Correspondence: cesare.valenti@unipa.it

Abstract: Discrete tomography is a specific case of computerized tomography that deals with the reconstruction of objects made of a few density values on a discrete lattice of points (integer valued coordinates). In the general case of computerized tomography, several hundreds of projections are required to obtain a single high-resolution slice of the object; in the case of discrete tomography, projections of an object made by just one homogeneous material are sums along very few angles of the pixel values, which can be thought to be 0's or 1's without loss of generality. Genetic algorithms are global optimization techniques with an underlying random approach and, therefore, their convergence to a solution is provided in a probabilistic sense. We present here a genetic algorithm able to straightforwardly reconstruct binary objects in the three-dimensional space. To the best of our knowledge, our methodology is the first to require no model of the shape (e.g., periodicity, convexity or symmetry) to reconstruct. Experiments were carried out to test our new approach in terms of computational time and correctness of the solutions. Over the years, discrete tomography has been studied for many interesting applications to computer vision, non-destructive reverse engineering and industrial quality control, electron microscopy, X-rays crystallography, biplane angiography, data coding and compression.

Keywords: discrete tomography; three-dimensional reconstruction; genetic algorithm

Citation: Toscano, E.; Valenti, C. A Genetic Algorithm for Three-Dimensional Discrete Tomography. *Symmetry* **2024**, *16*, 923. https://doi.org/10.3390/sym16070923

Academic Editor: Wiesław Leonski

Received: 12 June 2024
Revised: 9 July 2024
Accepted: 12 July 2024
Published: 19 July 2024

1. Introduction

The goal of computerized tomography [1] is the three-dimensional reconstruction of generic objects slice by slice. For a physical medium, a variable proportional to the density is summed along a sequence of rays to form a projection. The application of choice is medical diagnostics, and several hundreds of projections are needed because the body consists of numerous organs with different absorption rates. In the particular case of objects with a few density values, it is possible to reduce the number of projections, thus defining the so-called discrete tomography [2].

Classical algorithms of computerized tomography have been unable to provide satisfying results in the case of discrete tomography [3]. Although in the literature there are many works from a theoretical point of view, this latter reconstruction task is still particularly hard to face. Moreover, all current studies involve reconstructing a single slice at a time, without actually considering the entire three-dimensional object. The computational complexity of three-dimensional discrete tomography with missing data has been considered in [4] again from a mere theoretical viewpoint.

It can be proved in polynomial time whether there exists any object compatible with just a pair of projections [5,6]; the complexity of this reconstruction process becomes NP-hard in the case of at least three projections along non-parallel directions [7,8]. In addition, the number of binary images compatible with a small set of projections is typically huge even if these images can be very different from each other [9].

Exact reconstruction requires additional information: not only a big enough number of projections, but also the geometric and topological properties of the object [10]. For example, in crystallography, we may need some knowledge about the types of atoms to analyze, the

probability to find holes inside the object and its topology (e.g., some configurations are energetically unstable) [11,12]. Particular two-dimensional reconstruction algorithms were designed for ad hoc classes of binary images (e.g., periodic images, which have repetitions of pixels along some directions [13]; hv-convex polyominoes, which are connected sets with 4-connected rows and columns [14,15], convex objects [16]; and symmetric convex polygons [17], which is different from respecting the symmetry of particular organs of the patient [18]). Networks flows and Boolean solvers have been proposed to reduce the number of potential solutions [19]. Recently, a method has been proposed to determine the suitable directions of projections in the case of two-dimensional convex models [20]. Even a relatively seemingly simple case such as two-dimensional orthogonal discrete tomography with a Hamiltonicity constraint was proven to be NP-complete [21].

Heuristic approaches [22–25] have been applied to the reconstruction problem whereas evolutionary methodologies have shown their feasibility to reconstruct specific kinds of images such as circular or elliptical objects [26,27], images from interferometric fringes [28,29] and compact binary images [30]. A combined tabu-search and algebraic reconstruction algorithm has been developed for discrete tomography, although it needs a large number of projections [31].

Two-dimensional discrete tomography can be extended to the three-dimensional case by assuming some similarity between pairs of adjacent slices. This corresponds to the introduction of a model of the object to reconstruct, which may be useful from a general point of view though unfeasible in the case of security and medical imaging. To the best of our knowledge, no real, meaningful approach has been proposed so far.

This paper is organized as follows. After the description of the main parts of our evolutionary algorithm (Section 2), we describe the used dataset and the results (Section 3). Conclusions will highlight the advantages of the adopted technique, providing inspiration for future developments (Section 4).

2. Materials and Methods

In short, in a genetic algorithm, candidate solutions, called individuals, evolve from one generation to another toward better and better solutions: individuals can mate to exchange information and can suffer alterations in the hope that their new genetic arrangement is better. Obviously, the solution is not known, but it is known what it will have to look like.

Indeed, a fitness function evaluates how much each individual approaches the unknown solution and lets only the best individuals survive in the next generation according to an evolutionary pressure: the best individual of the entire process is presented as the final solution. To cover as large an area of the search space as possible, various techniques are considered to prevent individuals from being overly similar to each other. A description of modern evolutionary strategies is given in [32].

It should be noted that we do not know the exact solution to the problem, but a different representation of its appearance—specifically, its projections. Therefore, our goal is finding a data volume that satisfies the same set of projections, assuming that this is the desired solution. In fact, there is no guarantee that the output of any evolutionary algorithm will actually be the desired solution, even if it satisfies the entire set of projections. As a rule of thumb, the greater the amount of input information (i.e., the number of projections), the greater the chance that the solution will be correct.

This paper presents a substantially revised and extended account of research previously reported in [30]. Compared to that work, we are now able to directly reconstruct three-dimensional objects from a reduced number of appropriate projections in a reasonable time (please see Section 3). We set aside the initial idea of assuming that there is a correlation between adjacent slices because this implicitly introduces a model of the volume to reconstruct. Instead, our new approach forces the reconstruction by oblique projections that intersect different areas of separate slices. Therefore, our approach avoids the introduction

of any model that could force the reconstruction of the object towards a user-defined, albeit incorrect, shape.

The algorithm keeps the population size constant during the evolution, and it stops when a suitable solution is found. This avoids a rapid performance degradation. In any case, a halt condition is established by the total number ng of allowed generations.

A minimalist representation of the proposed algorithm is shown in Figure 1 (details will be provided shortly). The most suitable cardinality nc of the population, number ng of generations, and probabilities px and pm for the genetic operators will be considered in Section 3. For the sake of clarity, from now on, we will use the term *random* to indicate a pseudorandom generation according to a uniform probability distribution. In particular, we used the Mersenne Twister [33] algorithm with seed initialized according to the current time before each reconstruction to exclude repetitions in the experiments.

```
generate the initial random population of nc individuals
repeat until the goal is achieved
  every cb generations
    randomly shuffle the entire population
  otherwise
    randomly shuffle each separate deme
  under probability px choose pairs of random individuals
    apply crossover and keep the best two individuals
  under probability pm choose single individuals
    mutate qt points
  make a copy of the best individual within the population
draw the best individual so far encountered
```

Figure 1. Sketch of the proposed genetic algorithm.

2.1. Individuals

The proposed methodology allows us to *directly* reconstruct cubes with binary values; for practical purposes, we will consider $n \times n \times n$ points with $n = 32, 64$ and 128. It must be noted that these sizes correspond to binary images with about 181×181, 512×512 and 1448×1448 pixels, respectively, in the case of the standard two-dimensional discrete tomography; this provides a rough sense to compare our methodology against those already known in the literature in terms of data complexity.

We define an individual as a binary matrix of size $n \times n \times n$ with element equal to 0 if the corresponding point is white (i.e., it belongs to the background) or equal to 1 if the point is black (i.e., it belongs to the foreground). Please note that these two colors are swapped with respect to the usual representation in order to improve the graphical rendering of the forthcoming figures (see Figure 2).

Let k be the total number of foreground points in the body to reconstruct. Each individual in the population is actually a binary vector of length n^3 instead of k three-dimensional coordinates. This choice may result in a potential waste of memory in the case of very few foreground points (i.e., sparse data) but it simplifies the genetic operators, which must be performed quickly. Figure 3 shows the vector representation of the object in Figure 2**e**.

We used no distinctive information regarding the final goal; therefore, the initial individuals are formed by nc completely random vectors, each containing k elements equal to 1 and the remaining $n^3 - k$ elements equal to 0. Our experimental results confirm that a few generations are usually sufficient to approximate the searched solution, even in the case of small values of nc.

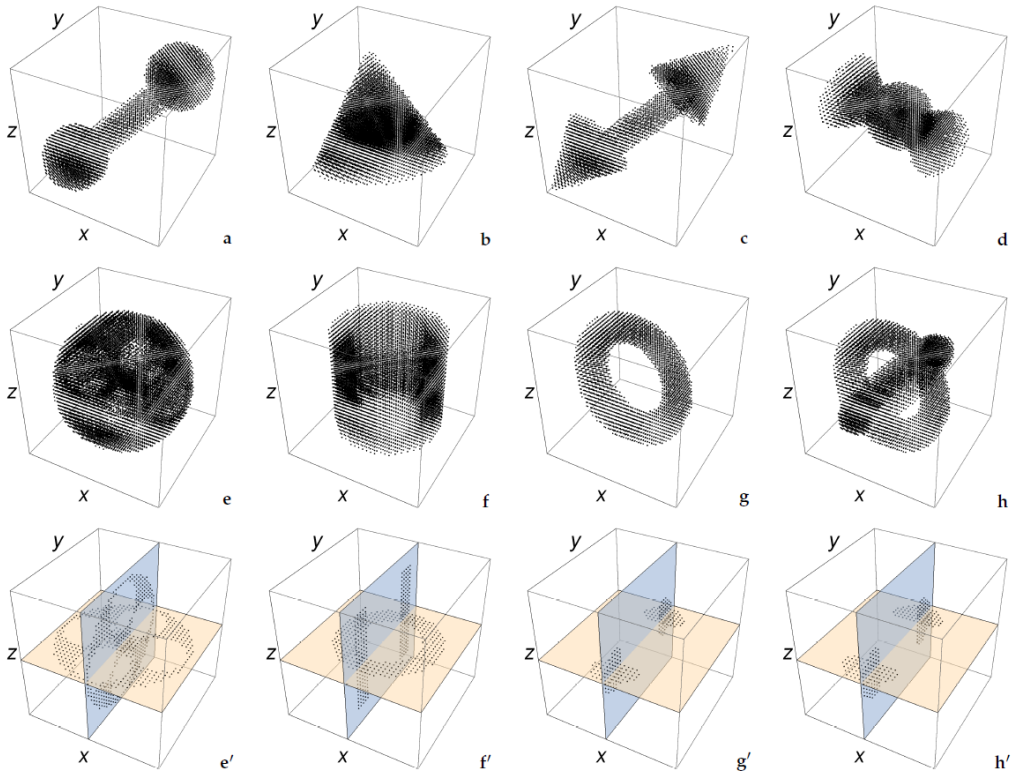

Figure 2. Some of the objects we considered for our experiments (**a**–**h**). For a better visualization, they are presented here by small dots. The lower objects (**e**–**h**) exhibit hollow spaces and therefore are shown also by means of a couple of their sections (**e′**–**h′**). For example, the sphere (**e**) contains six smaller empty spheres (**e′**).

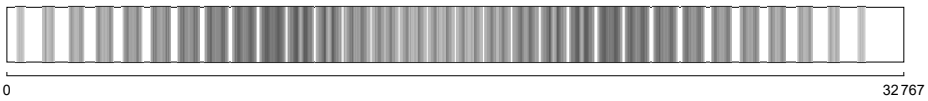

Figure 3. Binary individual vector of the object in Figure 2**e**.

2.2. Projections

Traditionally, the tomographic reconstruction has been approached by using projections that cut the body, literally slicing and then reconstructing it. We preferred to reconstruct the whole body at once from twelve projections intended as complete "radiographs"—that is, from a few planar images representing the number of black points along the direction perpendicular to each radiograph. When a large number of projections is available, the complexity increases because the image has to satisfy more constraints, and it is also true that this reduces the presence of eventual artifacts. In this paper, constraints are retained by projections in the three-dimensional space along non-coplanar directions: the underlying idea is that these projections should not revolve around any axis.

Besides the natural direct access through coordinates to each point in the individual, we created a structure (essentially a lookup table of memory locations of about 1.5, 12, 98 MiB with $n = 32, 64, 128$, respectively) that provides quick access to a whole projection plane by grouping all points that belong to it, arranged so as to reduce memory faults. We chose twelve particular projections (Figure 4) because points along the same direction vector are adjacent to each other and because they make it easy to create the lookup data structure. An animation that we include in the supplementary material shows that the projection planes $p_3 - p_{12}$ vary in shape and size inside the cube, depending on both the angle and depth of the considered projection. As a complete example, Figure 5 shows the projections of the object in Figure 2**e**.

Crossover and mutation (to be described in Sections 2.4 and 2.5) take most of the total processing time because each point modified in the individual's encoding is picked up by all twelve projections, which must be updated. In the case of crossover, the entire object modifies radically and, therefore, recalculating the projections is mandatory. On the other hand, mutation affects a few points and just some parts of the projections should be updated; nonetheless, it is more convenient to recalculate all whole projections. To recalculate the projections as fast as possible, we make extensive use of the ordered coordinates in the lookup table. According to this data structure, we actually deal with all twelve projections of a given object as a numeric vector. Figure 6 shows the projections in Figure 5 and should not be mistaken with the binary vector illustrated in Figure 3: the former is an example of input for our methodology whereas the latter is the corresponding output.

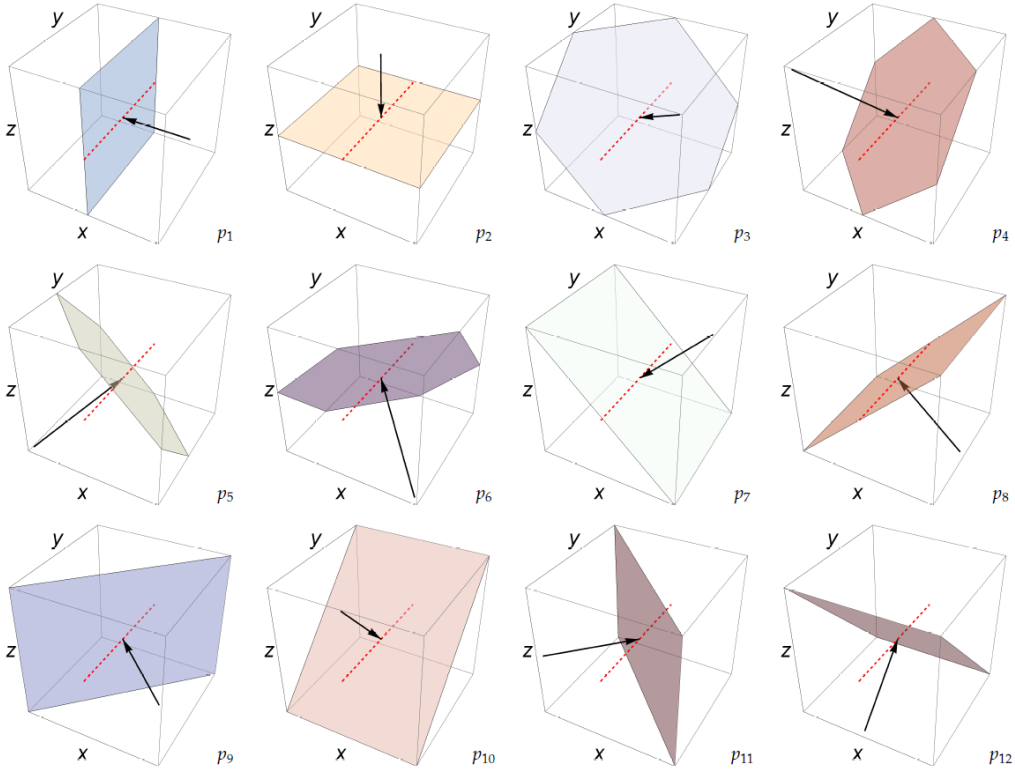

Figure 4. The twelve directions of the considered projections. The direction along the tunnel indicated by the red dashed line is missing to allow the body move along the tunnel of our mock-up equipment. All arrows point toward the center of the cube.

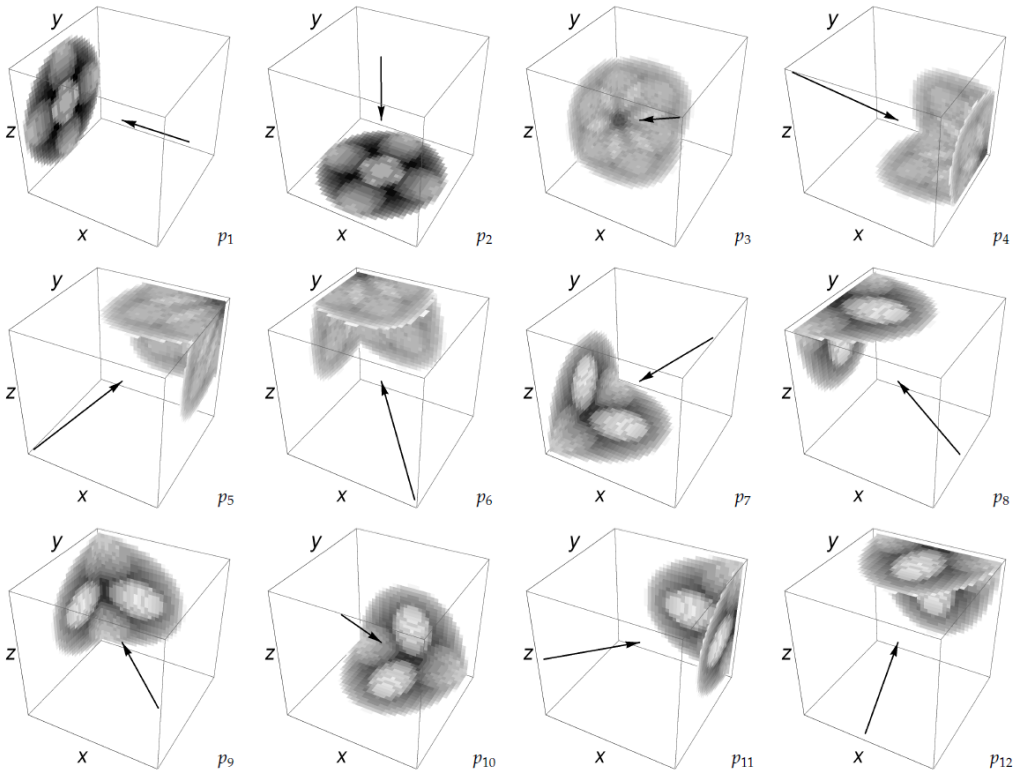

Figure 5. The projections used to reconstruct the object in Figure 2**e**. Pure black corresponds to the overall maximum count, equal to 28 foreground aligned points for this object (see Figure 6).

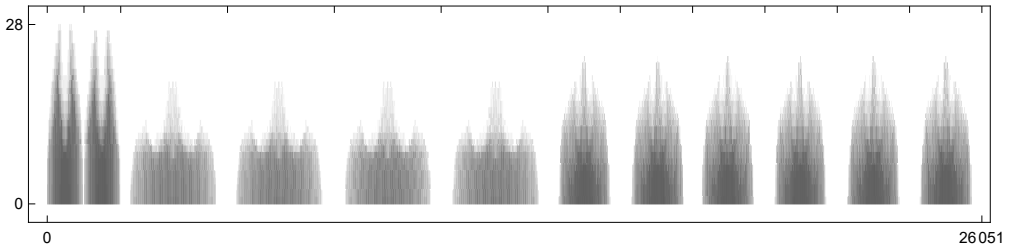

Figure 6. The projection vector of the object in Figure 2**e** stores all projections of Figure 5 and is provided as input to the proposed methodology. Twelve distinct zones are quite evident.

2.3. Fitness Function

Given an individual from the population, its fitness is merely the L_1 distance (i.e., the sum of the absolute value differences) between the individual's projection vector and that provided as input. Better individuals correspond to lower fitness; a zero value indicates that the individual's projections are identical to those of the object, although there is no guarantee that the two cubes are actually identical.

Despite its simplicity and fast computation, this fitness function provides enough variability to obtain correct results for all objects we considered. The population evolves until a stable solution is found or a maximum number of generations is reached.

2.4. Crossover

We considered several selection methods (e.g., tournament, ranking and roulette) to choose individuals for crossover, but they yielded poor results. As a consequence, we excluded them in the final version of the algorithm and the pair of individuals to recombine is randomly selected: this does not adhere strictly to the evolutionary theory, though it allows for more gene variability. Moreover, crossover occurs according to the probability px, which is set by the user.

Given any point within the cube and one of the possible twelve projections directions, the crossover operator cuts both individuals (i.e., parents) along the plane passing through that point and orthogonal to that direction. Both the point and the direction are determined randomly. The parts of the cut individuals are swapped to obtain a couple of offspring; their new projections must be calculated completely. The individuals thus obtained generally possess an incorrect number of foreground points, but comparing their projections with the input ones allows us to add random points where they are missing and to eliminate them where they are in excess. This last step of the operator basically constitutes some form of mutation, although it is aimed at improving the fitness. The projections of the final individuals must be updated. Figures 7 and 8 illustrate the crossover application on a couple of individuals attempting to reconstruct the object in Figure 2**e**.

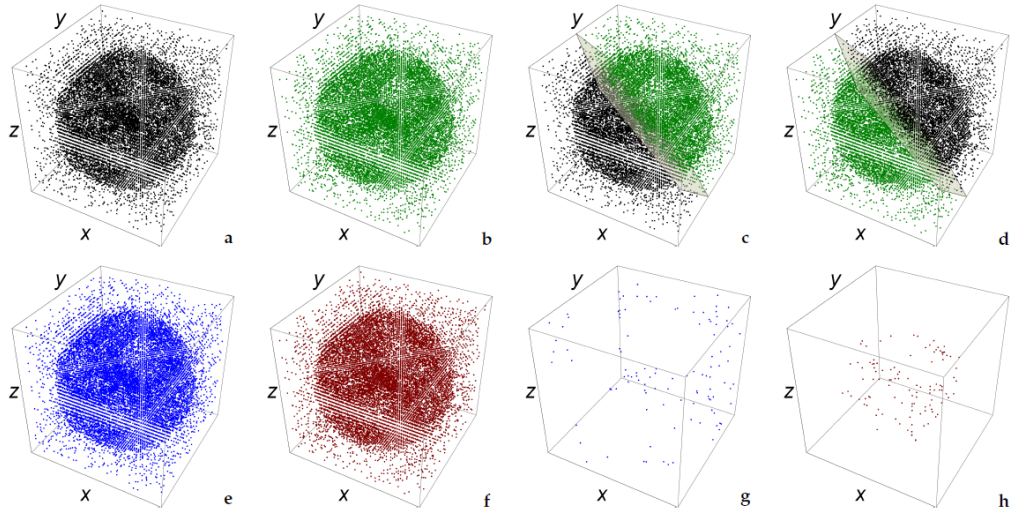

Figure 7. Given two parents (**a**,**b**), the offspring (**c**,**d**) are obtained by a random cut along one of the twelve directions, which is also randomly selected (p_5 in this example). It is usually needed to add or remove some points from the offspring (**e**,**f**) to keep their number. These points are highlighted in (**g**) (which shows the difference between (**c**,**e**) and (**h**) (which shows the difference between (**d**,**h**)). For interpretation of the colors, the reader is referred to the electronic version of this article.

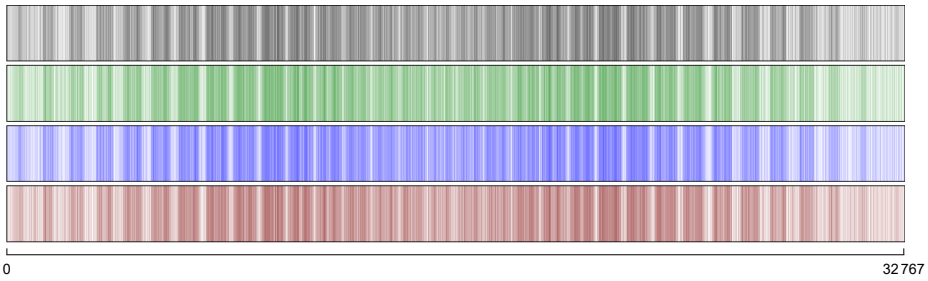

Figure 8. From top to bottom, binary individual vectors of the individuals in Figure 7**a**,**b**,**e**,**f**. For interpretation of the colors, the reader is referred to the electronic version of this article.

Parents and children are compared together based on their fitness values, and the two best ones remain in the future population to keep the population size constant throughout the whole process. This means that parents only are allowed to remain alive, as opposed to other techniques that replace parents with offspring in every case and regardless of their quality.

2.5. Mutation

This simple operator modifies the shape of an individual by changing the state of some of its points. The user specifies a priori the probability *pm* that an individual will undergo mutation, the amount *qt* of its points that will change from background to foreground, or

vice versa. This allows finer control on the behavior of the genetic algorithm in relation to the number of individuals: in the case of a very small population, it is advantageous to have few mutating individuals (a small value of pm), although the selected individuals must undergo a more pronounced mutation (a large value of qt). It is noteworthy that qt indicates the exact number of points, meaning that during mutation it is not possible for the same point to change its state more than once. Usually, pm is much smaller than px to guarantee a gradual convergence towards a stable solution within the search space.

As single isolated points (i.e., completely surrounded by points of opposite state) are generally attributed to noise, immediately after mutation we detect these isolated points and randomly choose them to change their state. Mutated points are chosen randomly and, therefore, the fitness of the new individual may be worse: an equal number of background and foreground points (actually, the minimum value between the quantity of these two sets of points) are exchanged to keep their total amount constant. This means that some isolated points may remain if the number of background and foreground is not equal. Regrettably, detecting isolated points is a particularly slow operation because it requires scanning each point within the cube through a sliding box with $3 \times 3 \times 3$ points. Figure 9 depicts an example of mutation.

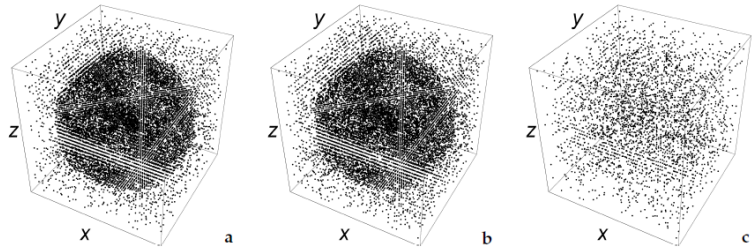

Figure 9. The object (**a**) in Figure 7**a** and one of its possible random mutations (**b**). Differences between (**a**) and (**b**) are highlighted in (**c**).

2.6. Crossbreeding

Crossbreeding is about improving the biological quality of a hybrid progeny obtained by sharing traits from parents of different lineages. Although the resulting offspring may sometimes be inferior to the parents, this problem is overcome by the genetic algorithm that quickly removes unpromising individuals. One possible pattern involves sub-populations (named demes) evolving independently during most of the time; at regular generations, all individuals are merged to compete altogether and mix their genotype with individuals from other demes [34].

When a deme remains isolated for a long time from the rest of the population, it may become a subspecies from the original population. In our case, each deme could converge independently to different solutions of the same problem, and this constitutes an implicit form of computational parallelism. From a practical point of view, we require that each deme evolves its gene pool independently of one another (thus covering different areas of the search space) and that in every cb generation the demes share information to converge toward a common best solution.

From Section 2.4, it is clear that a simple method of randomly choosing pairs of individuals to crossover is to shuffle randomly the population at the beginning of each generation. This is performed directly with a random permutation of nc integers, representing the indices of individuals in the entire population. So far, this corresponds to merging all the demes.

Let us consider each deme of equal size nc/nd, where nd is the number of demes. To separate the demes, it is sufficient to assemble a permutation of nc integers that has the indexes from 1 to nc/nd in the first nc/nd elements, the indexes from $1 + nc/nd$ to $2nc/nd$ in the elements from $1 + nc/nd$ to $2nc/nd$, the indexes from $1 + 2nc/nd$ to $3nc/nd$ in the elements from $1 + 2nc/nd$ to $3nc/nd$ and so on (Figure 10).

Basically, these nd demes remain alone for most of the time by way of this particular permutation; every cb generations, a totally random permutation of length nc merges them all together, thereby allowing individuals to migrate among them. From a computational perspective, the application of crossbreeding does not involve any real overhead with respect to the rest of the methodology.

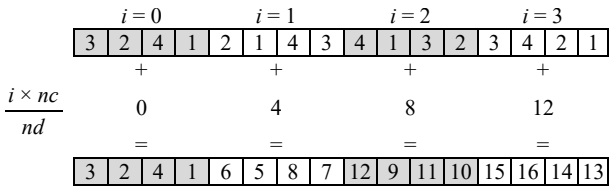

Figure 10. To randomly permute the nc individuals in nd demes, we apply the permutation made of nd offset random permutations with length nc/nd. Here, we consider $nc = 16$ and $nd = 4$.

2.7. Cloning

This is the ordinary elitist selection that assures that only the individual with the best fitness value will be present as two identical copies in the next population. In the case of demes, cloning is applied to each sub-population.

Usually, this reduces the variability of the hereditary characteristics in the case of very few individuals; nonetheless, we have experimentally verified that this strategy increases the accuracy of the final result when enough individuals are available.

3. Results

In order to verify the effectiveness of the proposed genetic algorithm, we produced twenty objects of different shapes with $32 \times 32 \times 32$, $64 \times 64 \times 64$ and $128 \times 128 \times 128$ points, sometimes composed by more than one part or with hollows inside them. Some instances have been shown in Figure 2.

We limited the preliminary experiments to $n = 32$; $nc = 64, 128, 256$; $px = 0.90, 0.95, 1.00$; $pm = 0.03, 0.05, 0.07$; $qt = 0.03, 0.05, 0.07$; $nd = 1, 2$; and $cb = 64, 128$ when $nd = 2$. This leads to 243 experiments per object. In addition, to minimize the effect of random fluctuations, each set of parameters was evaluated five times (i.e., 1215 experiments per object).

Despite the typical uncertainty of evolutionary approaches, all experiments returned correct results: not only were the final projections identical to the initial ones, but the reconstructed objects were actually identical to those satisfying the initial projections. Hence, the processing time alone can be used to identify the optimal values of parameters.

The application of crossover can be reduced as the number of individuals increases: with $nc = 64, 128, 256$, it is appropriate to use $px = 1.00, 0.95, 0.90$, respectively. On the other hand, we have already discussed how the computation time is linearly dependent on the number of individuals.

Regarding mutation, we have experimentally verified that the values $pm = 0.05$ and $qt = 0.07$ are the most convenient, independent of the number of individuals, thus confirming the importance of genetic diversity. In other words, it is advisable to mutate just a few individuals greatly.

Crossbreeding is irrelevant with $nc = 64, 128$ individuals because not enough gene diversity is available; instead, with $nc = 256$ individuals, it is useful to merge $nd = 2$ demes every $cb = 64$ generations.

We observed that the shape complexity of the object, which is difficult to formalize in rigorous terms, influences the computation time, whereas the mere number of points that compose the object does not modify so much the convergence rate towards the solution.

We almost always obtained the exact reconstruction of the object with $nc = 64$ individuals, making it useless to work with $nc = 128$ or even $nc = 256$ individuals; this entails a significant gain in processing time. However, it is suggested to increase the number of

individuals to compute the position of a large number of points or when the object consists of various components. We considered this possibility in the conclusion below.

Ultimately, the parameters that correspond to the best trade-off between convergence speed and correctness of the solution are, in general,

$$nc = 128,\ nd = 1,\ px = 0.95,\ pm = 0.05,\ qt = 0.07.$$

Figure 11 shows the *average* trend in reconstructing the objects in Figure 2 with this set of parameters. In the additional material, we provide animations for *single* reconstructions of the objects in Figure 2**e**–**h**.

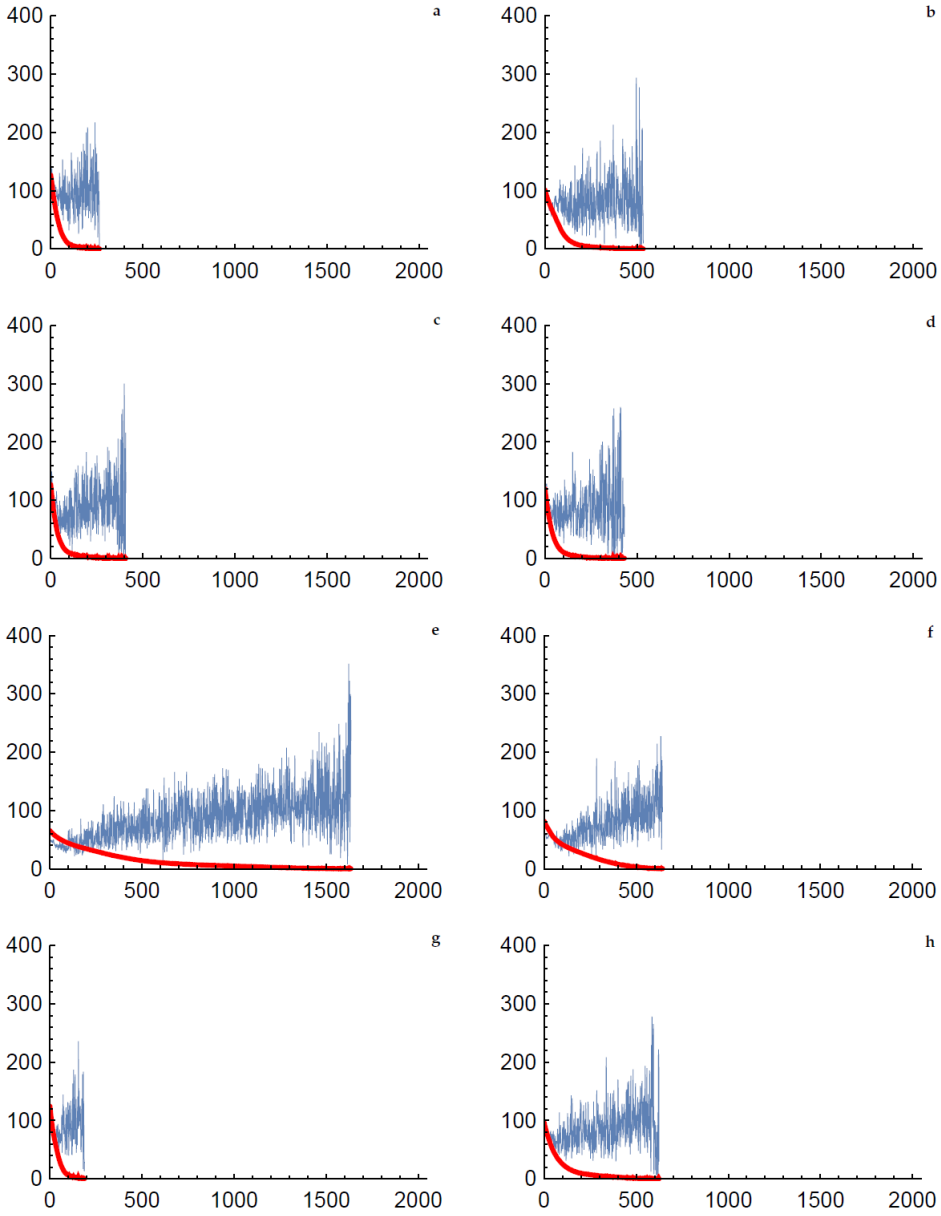

Figure 11. Average fitness (red) of the best individual and average standard deviation of the fitness (blue) among the five experiments per object in Figure 2, by using the preset parameters. The genetic diversity, highlighted by the increasing standard deviation, contributed to the convergence towards the correct solutions. As in the case (**e**), a greater number of generations indicates more complexity in the object to reconstruct. For interpretation of the colors, the reader is referred to the electronic version of this article.

Our algorithm was implemented in MatLab 2017b language for Windows 10 on an i7-2600 processor (produced in 2011) with 16 GiB of RAM. Although some experiments with $n = 32$ points and with $nc = 64$ individuals took just five seconds, the average time was about thirty seconds. A single experiment with $nc = 64$ individuals takes about four minutes when $n = 64$ and more than half an hour when $n = 128$. These times roughly double in the case $nc = 128$ and more than quadruple in the case $nc = 256$.

Altogether, the initial 24300 experiments (i.e., 1215 experiments for 20 objects) with just $n = 32$ required about three weeks. To make sure that the default parameters are indeed also the best in the case of $n = 64, 128$ and because the exhaustive completion of all experiments (i.e., with $n = 64, 128$) would take approximately four years, we performed further experiments with only the prefixed parameters plus 1% of random tests of the remaining experiments (corresponding to two weeks or so of calculations). Limited to the parameter values we considered, the search for the default ones is exhaustive in the $n = 32$ case and extended to the $n = 64, 128$ cases. All these evaluations confirmed the correctness of the default above parameters. Regarding the hardware we used, we expect that the latest generation i7 CPU with current RAM should provide just half the calculation time.

4. Discussion

Discrete tomography differs from standard computerized tomography in the small variety of density distribution of the object to analyze and in the very few projections to take. Although this research field is now more than thirty years old, it is still particularly complex to address, and the increasing number of recent publications highlights a great interest because of the variety of possible practical applications.

The methodology we presented here is able to compute in a reasonable amount of time the reconstruction of three-dimensional binary objects from twelve projections: about 30 s for a cube of $32 \times 32 \times 32$ points, which corresponds to a traditional slice of 181×181 pixels. Our evolutionary approach is simple and exhibits great flexibility because no model (e.g., periodicity, convexity or symmetry) is needed. To the best of our knowledge, this is the first discrete tomography algorithm able to deal with this type of reconstruction directly.

We were interested mainly in the development of reconstruction techniques, albeit in an efficient way on regular personal computers rather than in optimizations on particular hardware. In practice, we proved that it is possible to extend binary discrete tomography to the three-dimensional case with a sufficiently fast computing time.

It can be argued that twelve projections are more than what is usually proposed for the reconstruction in discrete tomography. We would like to emphasize that we use these projections for a direct reconstruction of three-dimensional objects of any shape—that is, without assuming any particular model. Indeed, many algorithms proposed in the literature use fewer projections to deal with planar reconstructions provided that it is given some information about the final appearance of the binary image.

The bottleneck of the proposed reconstruction methodology lies in accessing the three-dimensional coordinates of the points, basically as a function of the direction of the radiation considered. As described in Section 2.1, the shape of the plane orthogonal to this direction changes together with the position of the point itself. Due to this reason, we have precomputed all spatial coordinates, although this requires a large amount of memory in the case of big cubes.

Evolutionary algorithms are particularly suitable to exploit parallel implementations. Our code was written from scratch in MatLab language without even using its Global Optimization Toolbox. Except for the native parallelism offered by this vector programming language, no parallel instructions (multi-core CPU either GPU) have been applied explicitly. This should ease the translation of our methodology onto heterogeneous, high-performance machines by implementing in parallel all operations that can be subdivided into simpler steps to be performed independently of each other.

Author Contributions: Conceptualization, E.T. and C.V.; software, E.T. and C.V.; validation, E.T. and C.V.; writing—original draft preparation, E.T. and C.V.; writing—review and editing, E.T. and C.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Additional material can be found at https://tinyurl.com/toscanovalenti (accessed on 15 July 2024).

Acknowledgments: The authors express their gratitude for the helpful suggestions and appreciations by the anonymous reviewers. Elena Toscano is supported by the research fund of the University of Palermo: FFR 2024 Elena Toscano. Cesare Valenti is supported by the research fund of the University of Palermo: FFR 2024 Cesare Valenti. Both Elena Toscano and Cesare Valenti are members of the "Gruppo Nazionale Calcolo Scientifico—Istituto Nazionale di Alta Matematica (GNCS-INdAM)".

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kak, A.; Slaney, M. *Principles of Computerized Tomography Imaging. Society for Industrial Mathematics*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2001.
2. Herman, G.; Kuba, A. *Discrete Tomography: Foundations, Algorithms, and Applications*; Birkhäuser: Basel, Switzerland, 1999.
3. Gerard, Y. Reconstruction of Convex Sets from One or Two X-rays. *Fundam. Inform.* **2022**, *189*, 113–143. [CrossRef]
4. Kimura, K.; Kamehashi, T. Computational complexity of three-dimensional discrete tomography with missing data. *Jpn. J. Ind. Appl. Math.* **2021**, *38*, 823–858. [CrossRef]
5. Gale, D. A theorem on flows in networks. *Pac. J. Math.* **1957**, *7*, 1073–1082. [CrossRef]
6. Ryser, H. Combinatorial properties of matrices of zeros and ones. *Can. J. Math.* **1957**, *9*, 371–377. [CrossRef]
7. Gardner, R.; Gritzmann, P.; Prangenberg, D. On the computational complexity of reconstructing lattice sets from their X-rays. *Discret. Math.* **1999**, *202*, 45–71. [CrossRef]
8. Dulio, P.; Frosini, A.; Pagani, S.; Rinaldi, S. Ambiguous reconstructions of hv-convex polyominoes. *Discret. Math.* **2020**, *343*, 111998. [CrossRef]
9. Wang, B.; Zhang, F. On the precise number of (0,1)-matrices in U(R,S). *Discret. Math.* **1998**, *187*, 211–220. [CrossRef]
10. Matej, S.; Vardi, A.; Herman, G.; Vardi, E. *Discrete Tomography: Foundations, Algorithms, and Applications. Binary Tomography Using Gibbs Priors*; Birkhäuser: Basel, Switzerland, 1999; pp. 191–212.
11. Schwanders, P. Application of Discrete Tomography to Electron Microscopy of Crystals. In Proceedings of the Discrete Tomography Workshop, Szeged, Hungary, 5–27 August 1997.
12. Alpers, A.; Gritzmann, P.; Thorens, L. Stability and instability in discrete tomography Digital and Image Geometry. *Lect. Notes Comput. Sci.* **2001**, *2243*, 175–186.
13. Frosini, A.; Nivat, M.; Vuillon, L. An introductive analysis of periodical discrete sets from a tomographical point of view. *Theor. Comput. Sci.* **2005**, *347*, 370–392. [CrossRef]
14. Barcucci, E.; Del Lungo, A.; Nivat, M.; Pinzani, R. Medians of polyominoes: A property for the reconstruction. *Int. J. Imaging Syst. Technol.* **1998**, *9*, 69–77. [CrossRef]
15. Hantos, N.; Balázs, P. Reconstruction and Enumeration of hv-Convex Polyominoes with Given Horizontal Projection. In Proceedings of the Iberoamerican Congress on Pattern Recognition, Number 8258 in Lecture Notes in Computer Science, Havana, Cuba, 20–23 November 2013; pp. 100–107.
16. Tarsissi, L.; Kenmochi, Y.; Romon, P.; Coeurjolly, D.; Borel, J.P. Convexity preserving deformations of digital sets: Characterization of removable and insertable pixels. *Discret. Appl. Math.* **2023**, *341*, 270–289. [CrossRef]
17. Ceko, M.; Svalbe, I.; Petersen, I. A discrete projection analogue to Pick's theorem. *Graph. Model.* **2020**, *109*, 101066. [CrossRef]
18. George, A.; Chitteti, P.; Nair, S.; Karuppasami, R.; Joseph, M. Symmetry of computerised tomography of the brain in traumatic brain injury: A quality improvement audit. *BMC Neurol.* **2023**, *23*, 391. [CrossRef] [PubMed]
19. Di Marco, N.; Frosini, A. Properties of SAT Formulas Characterizing Convex Sets with Given Projections. *Lect. Notes Comput. Sci.* **2022**, *13493*, 153–166.
20. Pagani, S.; Tijdeman, R. Algorithms for linear time reconstruction by discrete tomography. *Discret. Appl. Math.* **2019**, *271*, 152–170. [CrossRef]
21. Bosboom, J.; Demaine, E.; Demaine, M.; Hesterberg, A.; Kimball, R.; Kopinsky, J. Path Puzzles: Discrete Tomography with a Path Constraint is Hard. *Graphs Comb.* **2020**, *36*, 251–267. [CrossRef]
22. Svalbe, I.; van der Spek, D. Reconstruction of tomographic images using analog projections and the digital Radon transform. *Linear Algebra Its Appl.* **2001**, *339*, 125–145. [CrossRef]
23. Anstee, R. The network flows approach for matrices with given row and column sums. *Discret. Math.* **1983**, *44*, 125–138. [CrossRef]
24. Balázs, P.; Balogh, E.; Kuba, A. Reconstruction of 8-connected but not 4-connected hv-convex discrete sets. *Discret. Appl. Math.* **2005**, *147*, 149–168. [CrossRef]

25. Batenburg, J. A network flow algorithm for reconstructing binary images from discrete X-rays. *J. Math. Imaging Vis.* **2007**, *27*, 175–191. [CrossRef]
26. Balázs, P.; Gara, M. An evolutionary approach for object-based image reconstruction using learnt priors. In Proceedings of the Scandinavian Conference on Image Analysis, Number 5575 in Lecture Notes in Computer Science, Oslo, Norway, 15–18 June 2009; pp. 520–529.
27. Venere, M.; Liao, H.; Clausse, A. A genetic algorithm for adaptive tomography of elliptical objects. *Signal Process. Lett.* **2000**, *7*, 176–178. [CrossRef]
28. Kihm, K.; Lyons, D. Optical tomography using a genetic algorithm. *Opt. Lett.* **1996**, *21*, 1327–1329. [CrossRef] [PubMed]
29. Kihm, K.; Lyons, D. Tomographic-image reconstruction using a hybrid genetic algorithm. *Opt. Lett.* **1997**, *22*, 847–849.
30. Valenti, C. A genetic algorithm for discrete tomography reconstruction. *Genet. Program. Evolvable Mach.* **2008**, *9*, 85–96. [CrossRef]
31. Frenkel, D.; Six, N.; De Beenhouwer, J.; Sijbers, J. Tabu-DART: A dynamic update strategy for efficient discrete algebraic reconstruction. *Vis. Comput.* **2023**, *39*, 4671–4683. [CrossRef]
32. Simon, D. *Evolutionary Optimization Algorithms*; Wiley: Hoboken, NJ, USA, 2013.
33. Matsumoto, M.; Nishimura, T. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.* **1998**, *8*, 3–30. [CrossRef]
34. Alba, E.; Troya, J. A survey of parallel distributed genetic algorithms. *Complexity* **1999**, *4*, 31–52. [CrossRef]