Accelerating heuristic convergence on the "Evolution of Mona Lisa" problem by including image-centric mutation operators

Theodor-Alexandru **Vlad**
Faculty of Computer Science
"Alexandru Ioan Cuza" University of Iasi
Iasi, Romania
vladtheodor0@gmail.com

Eugen Nicolae **Croitoru**
Faculty of Computer Science
"Alexandru Ioan Cuza" University of Iasi
Iasi, Romania
eugennc@uaic.ro

***Abstract*—The "Evolution of Mona Lisa" problem aims to approximate a target image by overlapping many semi-transparent polygons. The problem has been tackled in the past using multiple Nature-Inspired heuristics, and our main contribution is adding image-centric mutation operators (scaling, rotating and translating polygons).**

We compare Genetic Algorithms, Hill-Climbing and Simulated Annealing. A candidate solution has variable length (of, at most, 300 decagons) and, due to the variable opacity of polygons, order matters – resulting, in practice, in a pseudo-Messy GA. We use the same representation and mutation operators for the trajectory methods which, due to the focus on wall-clock time, outperform our GA implementation.

We find that these methods retain good image approximation at good run times: 98.9-99.2% (mean on 30 images), with a time limit of 30 minutes, on images 500-pixels tall.

***Index Terms*—Simulated Annealing, Hillclimbing, Genetic Algorithm, Image approximation**

I. Introduction

Approximating an image using polygons of varying colors and degrees of transparency is a problem first formulated in 2008 by Roger Alsing under the name *EvoLisa* [1]; the topic has received the attention of programmers and researchers alike in trying to conceive an evolutionary algorithm that can determine a set of polygons that closely resemble a target image (Leonardo da Vinci's *Mona Lisa* is commonly used as benchmark).

This problem is different from generating new images similar to an input sample of images (such as generating images with GANs [7]); the aim is to generate a model of a single target image, with approaches having large differences in terms of training time, model complexity, risk of overfit/premature convergence.

Obtaining a *deeper* model representing an image can be of use; changes to that model are more likely to yield *meaningful* alternatives to the original image, while altering the original pixels, with their implied uniform statistical distribution, is far less likely to. Such alternatives can be sought due to technical or artistic reasons, such as colour image "tracing", compression, or aesthetic alteration.

A. Previous approaches

Alsing's [1] approach was based on a simple Hillclimber which started with an empty candidate solution to which polygons were added over time. The existing polygons were mutated as to create offspring, which would replace the parent in the case of a better fitness value. A few implementations had appeared since Alsing's formulation of the problem. One such implementation [2] used a parallel Genetic Algorithm (GA), where the number of polygons in a chromosome was fixed, not variable.

A formal definition of the problem was stated by Gia Thuan Lam et. al. [3]. It is in this publication that some of the problems of earlier implementations were identified, such as the slow convergence speed and the general focus on exploration when searching for better solutions. As a result, a Simulated Annealing approach was proposed, which yielded better results than previous methods.

By including more information from the target image, a Hillclimbing-based algorithm combined with Canny Edge Detection and Delaunay Triangulation was shown by Julia Garbaruk et. al. [4] to obtain very good results.

B. Proposed approach

We choose to maintain the inspiration from the Central Dogma of Molecular Biology, by not including image-specific information into the chromosome; instead, we only allow the algorithm contact with the target image during evaluation (phenotype-level interaction with the environment, but no genotype-level interaction with the environment). Instead of image-specific/instance-specific information, we include problem-specific information, by adding mutation operators suited to working on polygons – the traditional geometric operations of scaling, translation and rotation.

We take these mutation operators and use them to generate a neighbourhood for three trajectory heuristics: Best-Improvement Hill-Climbing (HCBI), First-Improvement Hill-Climbing (HCFI), and Simulated Annealing (SA). While the size of the neighbourhood increases with the addition of

978-1-6654-6545-8/22/$31.00 ©2022 IEEE
DOI 10.1109/SYNASC57785.2022.00030

multiple ways of generating a neighbour, we still expect decreased convergence time relative to population-based methods (and good results, due to problem-adapted neighbourhood generation and good exploration).

C. Paper structure

Section II, Method, describes our method in detail, containing information about the representation, operators and algorithms used.

Section III, Experiment, describes our experimental set-up, data set, and problem-size-dependent algorithm parameters.

Section IV, Results, contains both results and their interpretation. Sub-section IV.A contains the overall results and their immediate discussion.

Section V, Conclusions, contains a top-level comparison and discussion of our results.

II. Method

We started algorithm design with a Genetic Algorithm (GA) inspiration, then further propagated the representation and mutation operators to trajectory methods: Best-Improvement Hill-Climbing (HCBI), First-Improvement Hill-Climbing (HCFI), and Simulated Annealing (SA). The main goal was to balance good image approximation with good run times; going through multiple design and test cycles, we found that GA convergence was slower, in terms of wall-clock time, than the basic trajectory methods we initially explored, and those results further guided our algorithm design, and made us investigate trajectory methods more thoroughly.

A. Representation

A chromosome, or candidate solution, contains a list of polygons. The length of that list is variable, although through preliminary tests, that number has been limited to between 50 and 300 polygons. Each polygon can have between 3 and 10 vertices in bi-dimensional space. A polygon also has a single RGBA colour (Red, Green, Blue, Alpha-opacity).

Because the length of a polygon is variable and because, during image construction, an opaque polygon which is drawn later can obscure polygons drawn previously, we are dealing with an ordered list of polygons, and not a set. Also, in practice, we are using a Messy Genetic Algorithm [8], as later chromosomes can modify the expression of previous ones: a dominant-recessive mechanism based both on list index but also on geometric position. In practice, we frequently observe opaque polygons – it is likely easier for an algorithm to improve fitness by adjusting a single parameter, the opacity of a polygon.

B. Evaluation and fitness

A candidate solution's fitness is calculated based on the distance between the image generated by rendering it (its phenotype) and the target image. We render the image – the polygons – against a black background, and the final image has no transparency. In the final images, each pixel has 3 channels, R, G, or B, and a channel intensity is an integer between 0 and 255.

We compute the channel-wise squared distance and sum it to a variable, then normalise it:

$$g = \frac{\sum_{i=0}^{n-1} d_i^2}{255^2 \cdot n}$$

where by d_i we denote the difference on a single color channel, and n is the total number of colour channels in an image ($3 \cdot Width \cdot Height$). This sum is then normalized, thus yielding a number between 0 and 1 which measures the distance between the individual and the target image.

We obtain a fitness function f by inverting g in regard to multiplication (and squaring the result, to increase selection pressure): by maximising f, the difference between images, g, is minimised.

We further obtain a human-readable measure of image approximation, f', by inverting g in regard to addition; when this number reaches 1, the two images are identical.

$$f = g^{-2}, f' = 1 - g$$

C. Generic operators

The operators shared between the GA and the trajectory methods are all mutation operators.

As a starting point, we have used the mutation operators defined by Roger Alsing [1]:

- deleting a random polygon from a candidate solution (probability: 0.04 – we attempt to simplify and trim our chromosome, as much as feasible)
- adding a randomly-generated polygon to a candidate solution (probability: 0.1)
- deleting a vertex (probability 0.01)
- adding a vertex (probability 0.01)
- slight adjustments of a polygon's individual vertex coordinates:
  - a small movement, of a few pixels (probability: 0.01)
  - a medium movement, of tens of pixels (probability: 1E-4)
  - a large movement, perhaps across the whole image (probability: 1E-6)
- adjustments of one RGBA channel value (probability: 0.01)

In addition to this, four new mutation operators are being introduced:

- rotating a polygon about its centroid with a given angle (probability: 0.01)
- scaling a polygon with a factor $\in [0.5, 1.5]$ (probability: 0.01)
- translating a polygon (different from existing translation-related mutation since here all the points are offset with the same amount in the same direction) (probability: 0.01)
- swapping two arbitrary polygons in a chromosome's representation (probability: 0.01)

The first three mutations are the basic bi-dimensional transformations; in introducing them, we reasoned that we should

have operators adapted to the objects we are evolving – bi-dimensional polygons should be changed by traditional bi-dimensional geometric operations.

The fourth mutation was added because, as explained earlier, we have a dominant-recessive mechanism. By swapping polygon indices, the algorithm can swap gene dominance. We presume this is of particular importance for approximating small details, although we do not explicitly favour small-area polygons (i.e. the swap indices are generated with uniform probability).

Each mutation operator has its own mutation probability, and every mutation has a chance of being applied to an individual during evolution.

D. Algorithms and algorithm-specific operators

1) Genetic Algorithm: For the Genetic Algorithm, we focused the large search space while maintaining good time performance. We encouraged constant exploration, and we delayed exploitation until the later generations – as exploiting on such a large space early on is likely to be wasted computation. Due to the size of the search space – and the frequency of mutations needed to explore it –, implementing a simple evaluation-caching mechanism wouldn't have reduced the number of evaluations needed.

We encourage constant exploration by using Hypermutation [9] [11] [10]. We tune it as such: if the best candidate solution doesn't change fitness for 50 generations, we multiply the probability of each mutation operator by 10, for a single generation. This ensures the GA constantly explores the search space, and does not have long epochs of static convergence.

We delay exploitation by starting with a cross-over probability of only 1%, and linearly increasing it to 20% during 10000 generations. The process is similar to that of cooling the temperature parameter for a Simulated Annealer, in the sense that the probability is multiplied with a constant number `CX_TEMP` each generation until it reaches the desired value:

$$CX\_TEMP = \left(\frac{0.2}{0.01}\right)^{\frac{1}{10000}} \approx 1.0002996$$

The crossover operator has been adapted in order to reflect the variable length of a chromosome. Instead of single/multiple cut-point crossover, we implemented the *Cut-and-Splice* [8] operator, which works by cutting two parents at random points in their list of polygons, then cross-concatenating the four newly-obtained slices into two offspring solutions. The four candidates (two parents, two offspring) are then sorted by fitness, and only the first two are preserved.

The population size is 50.

2) Hill-Climbing: We use two Hill-Climbing strategies, First Improvement and Best Improvement. Due to the random nature of the mutation operators we use to generate neighbours, we cannot practically exhaustively explore the neighbourhood of a candidate solution. Thus, these strategies must differ from their textbook examples.

For First Improvement, we generate a new neighbour of the current solution until we find a better one; as the stop condition is wall-clock time, this strategy does stop. In practice, due to the very large neighbourhood size, this strategy does not get stuck in local optima.

For Best Improvement, we generate 25 neighbours, and we choose the best one.

3) Simulated Annealing: For Simulated Annealing, we perform an exploration step to choose the starting temperature. In our preliminary tests, we have observed that a general-use initial temperature for every image does not exist, since it is image-dependent.

The way we determine a good starting value is as follows: for a given target image we generate 100 random solutions, calculate their average fitness, then perform binary search on the resulting number in order to see how much further we need to divide the average fitness; in the isolated case of Mona Lisa, we found the best results are obtained when dividing the average by 10, and storing that value as the initial temperature. The purpose of this is to have the algorithm start as close as possible to an "elbow" moment in the relationship between temperature and evaluation, such that the minimum amount of time is spent on cooling down and exploring solutions that are going to be overwritten anyway due to a high-temperature, yet we don't cut off valuable initial exploration.

We also chose a temperature multiplier of 0.997 (also chosen empirically, but kept constant), which is applied at every "step". Each step consists of generating at most 100 neighbours, and replacing the candidate solution in a First-Improvement fashion.

III. Experiment

A. Data set

We chose 30 images, to ensure a sufficiently diverse and large sample to test our algorithms.

Each image was resized such that its height became approximately 500 pixels, while its aspect ratio remained constant – thus, the width was variable. In our data set, the width has a median of 490 and an IQR of 166.75, since we sought to choose a balanced set of portrait and landscape images.

From our investigation of the state-of-the-art, this is somewhat larger than the image sizes most frequently used, thus, requiring more time to evaluate, since we do compare pixel-to-pixel.

We included pictures, paintings, and computer-generated objects (a fractal tree).

B. Parameters

Since we ran both population and trajectory methods, we sought to compare the methods fairly by having the elapsed wall-clock time as a stop condition. We chose to allow each algorithm to run for 30 minutes.

In practice, the GA ran for $\approx 50000 - 55000$ generations, with a population size of 50. The trajectory methods also performed a similar number of evaluations.

IV. Results

Although we ran multiple smaller-scale experiments to evaluate the impact of various parameters (most work was done for the new mutation operators), they are largely subsumed in the final comparison; as such, these are the results we will show here.

The similarity (or approximation) measure we use here is the f' measure described in Section II.B. If it is 0, the images are entirely dissimilar (e.g. a white image approximating a black image), if it is 1, the images are pixel-perfect identical. It is not a linear measure, but a squared measure – the mean squared per-channel Euclidean distance. When we present these results as percentages, we try to have them easier to visually compare by multiplying with 100; we do not mean to imply linearity. The resulting images are accessible at https://profs.info.uaic.ro/~eugennc/research/res/.

A. Full results

Name	GA	HCBI	HCFI	SA
20th Century	98.171%	98.731%	98.876%	98.913%
Toyota AE86	97.392%	98.411%	98.323%	98.411%
Bowl of fruit	99.329%	99.575%	99.545%	99.560%
Caragiale	96.780%	97.350%	97.620%	97.575%
Cat	99.068%	99.280%	99.299%	99.286%
Croissant	98.896%	99.192%	99.305%	99.311%
Eagle	98.915%	99.215%	99.246%	99.207%
Eminescu	98.647%	99.098%	99.133%	99.232%
Fractal tree	94.992%	96.551%	96.639%	96.737%
Ginevra	99.002%	99.411%	99.423%	99.380%
Girl	99.427%	99.625%	99.621%	99.655%
House	97.877%	98.177%	98.200%	98.200%
Iron Man	99.592%	99.666%	99.687%	99.697%
Lady	99.700%	99.760%	99.767%	99.793%
Lincoln	99.144%	99.431%	99.508%	99.563%
Logo FII	99.803%	99.874%	99.884%	99.880%
Marilyn	98.455%	98.925%	98.969%	99.004%
Mona	99.184%	99.277%	99.315%	99.328%
Musashi	98.985%	99.189%	99.184%	99.208%
Mushroom	97.688%	97.915%	97.864%	98.006%
Landscape	99.106%	99.293%	99.295%	99.338%
Landscape 2	99.244%	99.440%	99.443%	99.448%
Rubik's Cube	96.165%	98.296%	98.310%	98.229%
The Scream	98.936%	99.180%	99.218%	99.201%
Space	98.938%	99.170%	99.185%	99.216%
Starry Night	98.581%	98.813%	98.809%	98.802%
Tower of Pisa	98.418%	98.731%	98.752%	98.753%
Van Gogh	98.382%	98.764%	98.748%	98.852%
Walter White	99.640%	99.725%	99.720%	99.748%
Zebra	96.328%	96.988%	96.997%	97.200%
Average	98.49%	98.90%	98.93%	98.96%
Stdev	1.15%	0.82%	0.80%	0.78%
Median	98.92%	99.18%	99.20%	99.21%
IQR	0.95%	0.7%	0.69%	0.67%

TABLE I: Full results: image approximation (similarity)

As shown both in Table I and Fig. 1, the GA is slightly outperformed by the trajectory methods. SA seems to be minutely better than the other methods, as it most often obtains the best similarity score, but the difference is both extremely small and not statistically significant.

Based on these results, we believe that adding the traditional 2-dimensional geometric transformations as mutation operators was a success, yielding good results in reasonable time.

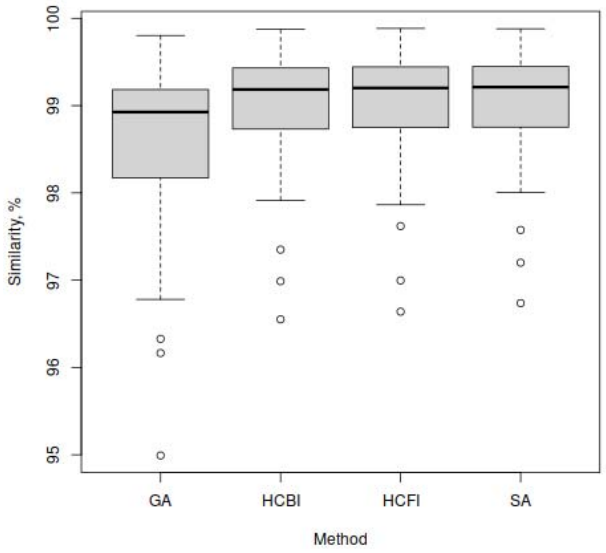


Fig. 1: Results, boxplot: image approximation (similarity)

Increasing the neighbourhood in problem-appropriate ways has lowered its difficulty – or, alternatively, increased the problem-solving ability of the algorithms using these operators. In short, by using problem-specific information, we have better fit our algorithms to the problem [12] [13]. Thus, the problem can be solved by our customised trajectory methods slightly faster than our customised GA.

B. Particular results

This subsection covers interesting results – either very good, very bad, or just well-known.

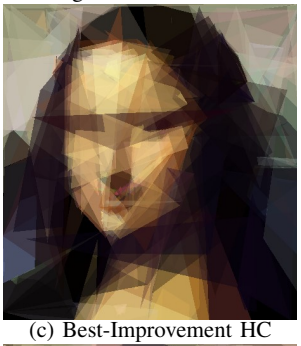
1) Mona Lisa: Figure 2 shows a comparison of the solutions yielded by the four algorithms in reference to the portrait of Mona Lisa. Although the difference between GA (99.184%) and SA (99.328%) seems small, SA better approximates the details in the painting; the image area those pixels occupy is relatively small, but the relevance to human perception is large. SA better reproduces human-relevant details, such as eyes, mouth, nose, and facial shading.

From visually inspecting this and other results, we conclude that the human perception difference is greater than the pixel-difference would suggest, although the ranking isn't inverted. However, we can use our (admittedly, subjective) perception to distinguish among the statistically-grouped trajectory methods, and tentatively claim that Simulated Annealing does provide better visual approximations.

2) Best results: The best results were, unsurprisingly, obtained on images easier to approximate through polygons. In Fig. 3, we see a painting on a flat background (which is very easy to approximate); the "Logo FII" image (not included here due to space reasons) is a flat logo, made up entirely of polygons, thus very easy to approximate.

(a) Reference image

(b) Genetic Algorithm



(c) Best-Improvement HC

(d) First-Improvement HC

(e) Simulated Annealing

Fig. 2: Result comparison: Mona Lisa

3) Worst results: In Fig. 4 we see that a highly-intricate image is hard to approximate by overlapping polygons – indeed, such an image would be better approximated by overlapping line segments.

Fig. 5 shows this yet again: a zebra's larger stripes can be approximated with a small number of polygons, in a time-constrained run, but when those details become minute, this approach yields poor results. Or, rather, the algorithm chooses to save up its polygon budget by covering the fine detail with only a few polygons averaging out that detail.

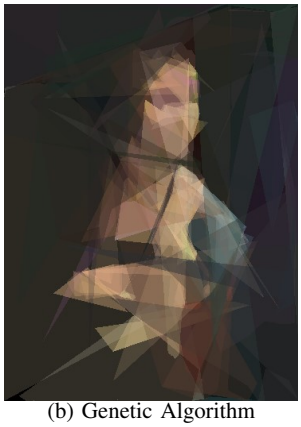
Both images show that larger branches/stripes are approximated well, smaller details are not.

C. Convergence speed

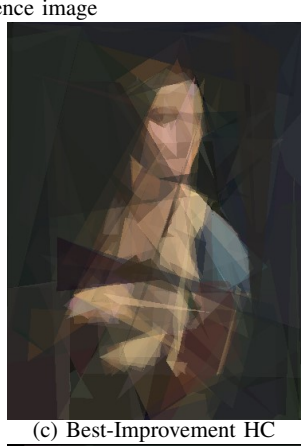
When analysing convergence, we find it useful to report both our similarity score (f') and the internal fitness value (f); they are defined in Subsection II.B .

For instance, in the case of Mona Lisa, the f and f' values are shown in Tab II.

(a) Reference image

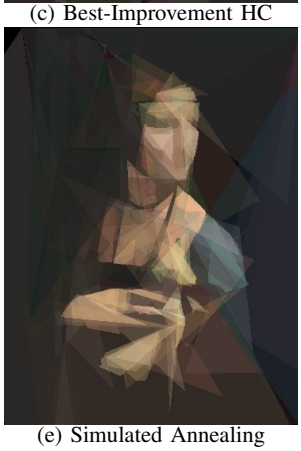


(b) Genetic Algorithm



(c) Best-Improvement HC

(d) First-Improvement HC

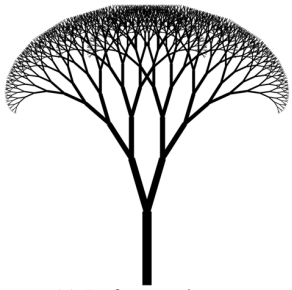


(e) Simulated Annealing

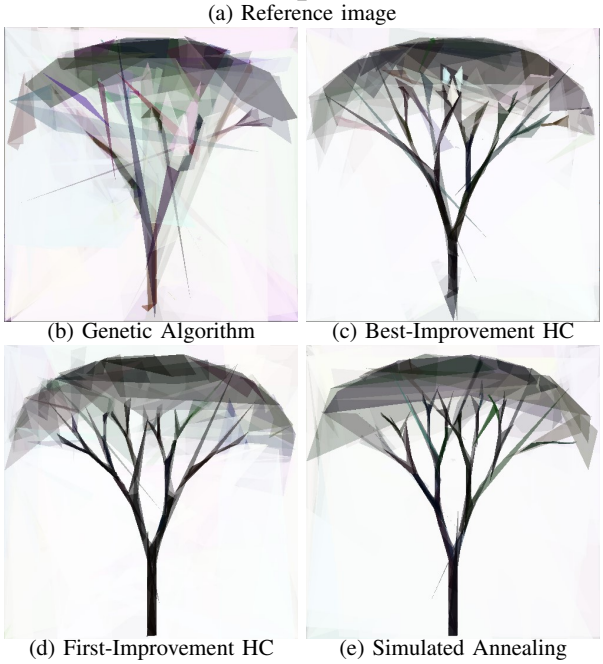
Fig. 3: Result comparison: Lady with an ermine

When examining at the initial phase of the SA algorithm (Fig. 6), we see that we "waste" relatively little time on a too-high temperature: within $\approx 30s$ we reach an "elbow". This was our aim when we set the initial temperature as described in Sub-sub-section II.D.3 .

The evolution of the average fitness over 30 minutes of runtime for the full sample of 30 images can be observed in Fig. 7. On average, SA performs marginally better, followed closely by HCFI, HCBI, and last by the GA.



(a) Reference image

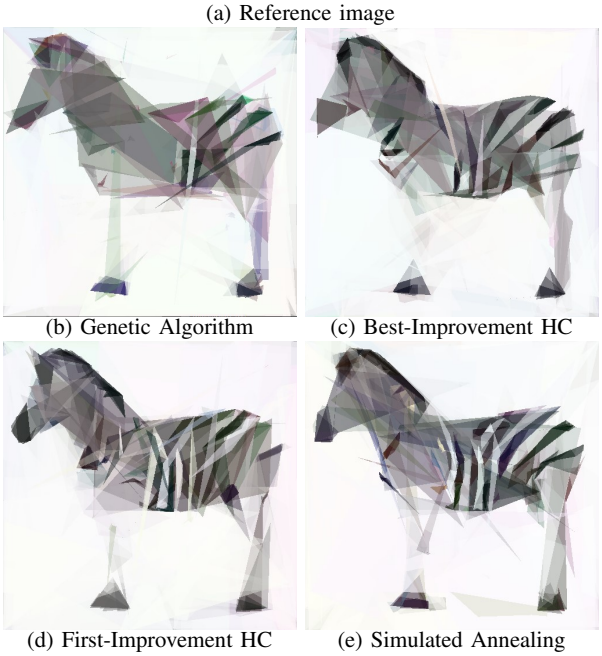


(b) Genetic Algorithm (c) Best-Improvement HC

(d) First-Improvement HC (e) Simulated Annealing

Fig. 4: Result comparison: Fractal tree

(a) Reference image



(b) Genetic Algorithm (c) Best-Improvement HC

(d) First-Improvement HC (e) Simulated Annealing

Fig. 5: Result comparison: Zebra

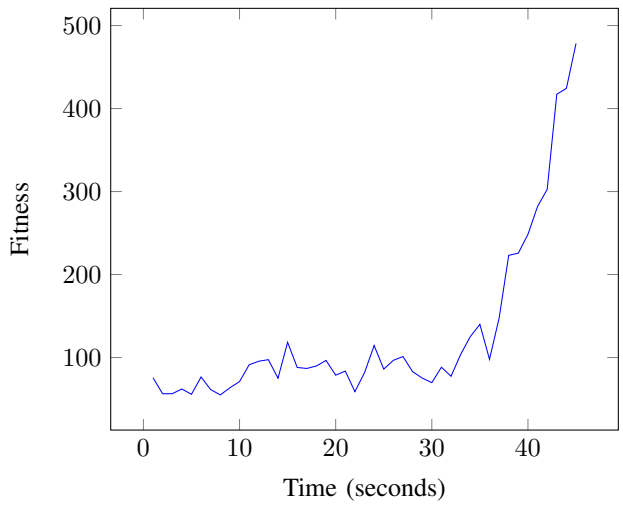


Fig. 6: Simulated Annealing - first 45 seconds (Mona Lisa)

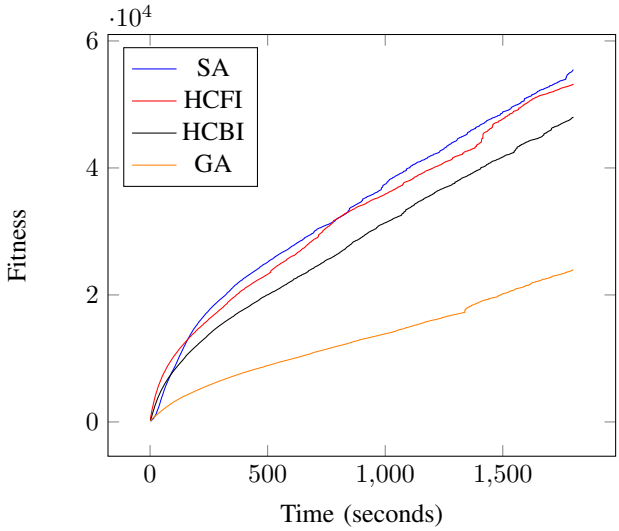


Fig. 7: Average fitness evolution by method

Name	GA	HCBI	HCFI	SA
f	15018	19130	21311	22144
f'	99.184%	99.277%	99.315%	99.328%

TABLE II: Mona Lisa - the relationship between f, fitness, and f', similarity

The average is not true for every instance; for example, in the case of image titled *Eagle* (Table I), the ordering of the results is – from best to worst – HCBI > HCFI > SA > GA.

While monitoring the average fitness over time, we came across another interesting result: the time it takes for each method to reach a certain similarity percentage differs between methods. This is presented in Table III and displays the average (across all images) time in seconds to yield a solution with 96%, 97% or 98% similarity.

Method	96% sim.	97% sim.	98% sim.
GA	12s	27s	75s
HCBI	5s	8s	20s
HCFI	2s	4s	12s
SA	15s	24s	38s

TABLE III: Average times to reach certain similarity thresholds

Here we observe an advantage of the Hillclimbing algorithms over the Simulated Annealer: speed of convergence in the early stages. HCFI in particular is the fastest to reach any of the considered similarity percentages. As seen in Figure 7, at about 150-200 seconds the SA approach overtakes HCFI, but HCFI remains the best contender as far as the first 3 minutes of runtime are concerned. As such, one possible idea worth researching is generating a good solution using HCFI, and then improving it with SA – or, due to the similarity of the methods, having a non-monotonous temperature, using a greedy descent to approach good solutions, then switching to a non-zero temperature to enable attraction basin escapes.

V. Conclusion

In this paper, we analysed the impact image-centric, 2D geometry inspired mutation operators can have on approximating images by overlapping coloured and semi-transparent polygons.

Overall, we found that adding these problem-specific operators improved both results and algorithm speed.

When comparing to Julia Garbaruk et. al. [4], which we hold as the current state of the art, we find that we can approximate larger images, faster, and still retain good results. We explain this by the fact that we maintain the inspiration from the Central Dogma of Molecular Biology: by only allowing the algorithms to interact with the target image during evaluation, we access the pixel buffer less, and thus perform fewer costly operations (and less bottleneck or cache-miss -prone I/O). Instance-specific information is relatively costly to access, whereas problem-specific information can be introduced during the algorithm design phase.

The Genetic Algorithm performed worse than the trajectory methods; since we have added problem-specific mutation operators, which seem to allow the efficient exploration of a huge neighbourhood, it would seem the statistical estimation a GA routinely performs is not as needed. Of course, we have only performed time-limited comparisons, as this was one of our goals. It could be that, allowed more computational resources, the GA could arrive at better approximations than the trajectory methods.

Among the trajectory methods, the results were not statistically separable (we performed a Welch t-test with Holm p-value adjustment, $\alpha < 0.05$, which suggested no significant difference). However, by visually inspecting the images, Simulated Annealing *seemed* to have arrived, in most cases, at better visual approximations, approximating more details which to us seemed relevant (such as human facial features, like eyes, mouth, nose, small facial shading).

Acknowledgement

We would like to thank Ruxandra Vâlcu for her feedback and editing help.

References

[1] Roger Alsing, *Genetic Programming Evolution of Mona Lisa* (2008) https://rogerjohansson.blog/2008/12/07/genetic-programming-evolution-of-mona-lisa/

[2] Dr. Karoly Zsolnai-Feher, *Repainting the Mona Lisa with a Genetic Algorithm* (2015) https://users.cg.tuwien.ac.at/zsolnai/gfx/mona_lisa_parallel_genetic_algorithm/

[3] Gia Thuan Lam, Kristiyan Balabanov, Doina Logofatu and Costin Badica, *Novel Nature-Inspired Selection Strategies for Digital Image Evolution of Artwork* (2018), International Conference on Computational Collective Intelligence, pp 499–508 https://www.researchgate.net/publication/328403711_Novel_Nature-Inspired_Selection_Strategies_for_Digital_Image_Evolution_of_Artwork

[4] Julia Garbaruk, Doina Logofatu, Costin Badica and Florin Leon, *Digital Image Evolution of Artwork Without Human Evaluation Using the Example of the Evolving Mona Lisa Problem*, Vietnam Journal of Computer Science Vol. 9, No. 2 (2022) 203–215 https://www.worldscientific.com/doi/epdf/10.1142/S2196888822500075

[5] Kun Li, Ming Li and Hao Chen, *The Empirical Analysis of Exploration-Exploitation Tradeoff for Uncertain Environment*, 2014 Seventh International Symposium on Computational Intelligence and Design https://sci-hub.se/10.1109/ISCID.2014.83

[6] S. Kirkpatrick, C. D. Gelatt, Jr. and M. P. Vecchi, *Optimization by Simulated Annealing*, Science (1983) Volume 220 - Number 4598 https://sci-hub.se/10.1126/science.220.4598.671

[7] Goodfellow, Ian and Pouget-Abadie, Jean and Mirza, Mehdi and Xu, Bing and Warde-Farley, David and Ozair, Sherjil and Courville, Aaron and Bengio, Yoshua, *Generative adversarial nets*, 2014, Advances in neural information processing systems

[8] Goldberg, D. E.; Korb, B.; Deb, K. *Messy Genetic Algorithms : Motivation Analysis, and First Results*. 1989, Complex Systems. 5 (3): 493–530

[9] Cobb, H. G., *An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environment*, 1990, NRL Memorandum Report 6760

[10] Morrison, R. W. and De Jong, K. A., *Triggered hypermutation revisited*, 2000, Evolutionary Computation, Proceedings of the 2000 Congress on., IEEE, Vol. 2, pp. 1025-1032

[11] Stolzenburg, W., *Hypermutation: Evolutionary fast track?*, 1990, Science News, No. 25, Vol. 137, pp. 391

[12] Wolpert, David H. and William G. Macready, *No free lunch theorems for search*, 1995, SFI-TR-95-02-010, Santa Fe Institute

[13] Wolpert, David H. and William G. Macready, *No free lunch theorems for optimization*, 1997, Evolutionary Computation, IEEE Transactions on, No. 1, Vol. 1, pp. 67-82