# Paintings-from-Polygons: Simulated Annealing

Redouane Dahmani[1][0000−0001−8771−3086], Sven Boogmans[2][0000−0002−9422−3023], Arne Meijs[3][0000−0003−1747−730X], and Daan van den Berg[4][0000−0001−5060−3342]

[1] Institute for Interdisciplinary Studies (IIS), University of Amsterdam
[2] Master Information Studies, University of Amsterdam
[3] Minor Programmeren, University of Amsterdam
[4] Informatics Institute, University of Amsterdam
{redouane.dahmani,sven.boogmans,arne.meijs}@student.uva.nl
D.vandenBerg@uva.nl ✉

**Abstract.** Paintings-from-polygons is an optimization problem in which the objective is to approximate a target bitmap by optimally arranging a fixed number of semi-opaque colored polygons. In a recent report, two optimization algorithms showed strong performance on the task, but the third, simulated annealing, failed miserably. The authors conjectured its poor performance to be due to the specific cooling schedule used.
In this study, we use the same problem instances but parameterize simulated annealing with eight new cooling schedules known from literature: Geman & Geman with different $c$-parameter settings, geometric, linear, cosine, linear reheat, sigmoid, and staircase. We find that the previous conjecture was right: the performance of simulated annealing on this problem critically relies on its cooling schedule, and can be quite successful once parameterized properly. Moreover, we find a consistent 'performance hierarchy' in which *most* of the cooling schedules' performance appears related to their average temperatures *only*.

**Keywords:** Paintings-from-Polygons · Computational creativity · Evolutionary art · Paintings · Polygons · Simulated annealing · Cooling schedules · NP-hard · Optimization · Heuristic algorithm

## 1 Introduction

As a discipline, computational creativity has flourished in recent history. Subjects such as casual creation, photo quality classification, music genre detection, evolutionary architecture and visual imagery have gone through substantial development in the past decade [11][6][14][19][20]. Many of these are in some way related to optimization algorithms such as hillclimbing, plant propagation, simulated annealing or genetic algorithms, which have all proven successful in both theoretical and practical settings [18][25][29][12].

Being located on the intersection of computational creativity and algorithmic optimization, the paintings-from-polygons (PFP) problem's objective is to

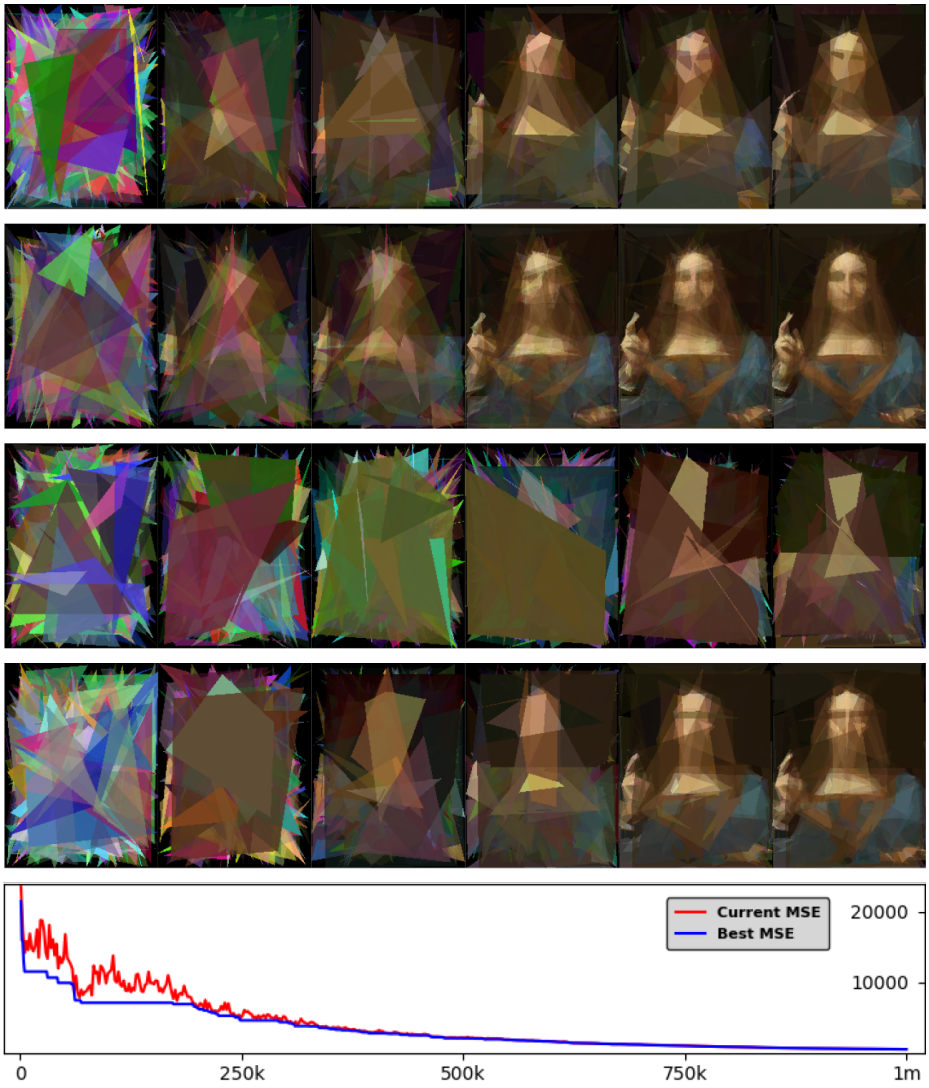


Fig. 1: The choice of cooling schedule greatly influences the performance of simulated annealing on paintings-from-polygons. Top to bottom: linear reheat, Geman & Geman (c=1), sigmoidal, and geometric cooling schedules; the fifth box shows the MSE for the algorithmic approximation directly above it. All constellations contain 1000 vertices in 250 polygons, and completed a run of $10^6$ iterations (eq.: evaluations). A video clip of the process is publicly available[1].

approximate a given target bitmap by arranging a fixed number of semi-opaque colored polygons. On initialization, the polygon constellation's properties are assigned random values, but the numbers of polygon and vertices are fixed (to

250 and 1000 in this study) throughout an experiment [22]. From there, heuristic algorithms can change a polygon's color, move one of its vertices, transfer a vertex from one polygon to the next, or mutate the *drawing index* of the polygons, a parameter that determines the rendering sequence when generating an output bitmap. By iteratively applying these mutations to the polygon constellation and rendering afterwards, a target bitmap, usually a painting, is approximated ever closer (Fig. 1). The objective value for each rendered bitmap is the difference to the target bitmap, expressed as the mean squared error (MSE) over the color channels:

$$MSE = \sum_{i=1}^{x \cdot y \cdot 3} \frac{(Rendered_i - Target_i)^2}{x \cdot y} \tag{1}$$

Here, $x$ and $y$ are the dimensions of the target bitmap in pixels, $Rendered_i$ is the value of a red, green or blue (RGB) channel in the bitmap rendered from the polygon constellation, and $Target_i$ is the value of the corresponding RGB-channel for the target bitmap. Since the maximum difference between two RGB channels is 255, and there are three channels in one pixel, the maximum MSE is $255^2 \cdot 3 = 195075$. The minimum MSE of zero is reached iff the rendered polygon constellation and the target bitmap are identical, a situation that is highly unlikely for any realistic number of vertices. A simplified form of PFP was proven to be NP-hard in 2020 [8]. Although this technically speaking guarantees nothing about the *full* PFP-problem, it has no known subexponential exact methods, and thereby provides a suitable testing ground for heuristic algorithms. It also gives rise to some interesting questions about the complexity[5] of visual imagery, and produces aesthetically pleasing results.

There are $256^{3^{(180 \cdot 240)}} \approx 7.93 \cdot 10^{312,107}$ different existable bitmaps of 180 by 240 pixels, the default size in the experiments. A maximum of $180 \cdot 240 \cdot 4 = 172,800$ vertices is therefore sufficient to exactly reproduce any given target bitmap of aformentioned dimensions[22]. However, Paauw and Van den Berg also give a lower bound of 39,328 vertices, the lowest number that can give rise to more different constellations than the number of possible bitmaps of 180 by 240 pixels. These numbers are a long way beyond computational efficiency, and assuming the state space is not *completely* convex, the problem could well be untractable, and heuristic algorithms such as simulated annealing, could provide a feasible way forward.

In their seminal study, Paauw and Van den Berg used a stochastic hill-climber, simulated annealing, and plant propagation on paintings-from-poygons [22]. These three algorithms all retained their best found objective values (or 'fitness') throughout the runs of $10^6$ function evaluations (Fig. 1, lower part). The hillclimber and plant propagation achieved relatively good results, but the same could not be said for simulated annealing. In this study however, we'll take care of that.

[5] The term is intentionally loosely applied here; it could mean 'Kolmogorov complexity', or 'RLE-complexity', amongst others.

## 2 Simulated Annealing

Introduced by Kirkpatrick, Gelatt & Vecchi in 1983 and independently by Černý in 1985, the simulated annealing algorithm draws inspiration directly from statistical mechanics, the central discipline of condensed matter physics [18][10][15]. In the physical annaealing process, a desirable low energy state of a metal is reached by optimizing the spatial configuration of the atoms. To reach such a configuration, dislocated atoms need to move to vacant sites which can be facilitated by cooling the liquid metal *slowly*. Higher temperatures increase the atoms' movement, but also increases the number of expected vacant sites due to the increase of mixing entropy. Simulated annealing emulates this process for combinatorial optimization problems, even up to the exact Boltzmann factor (Eq.2) used for 'movement' through a combinatorial state space. As such, it is one of the few heuristic algorithms in the field that is more than just a metaphor; it truly pries at the barrier between physics and information science. Good solutions to combinatorial problems such as timetabling, the wiring of electronic systems, the job shop scheduling problem and the traveling salesman problem have been found earlier using simulated annealing [10][18][5][30][23].

For many heuristic algorithms, parameterization is "essential for good algorithm performance" [26], and simulated annealing is no exception. It has proven quite sensitive to parameter changes in an early study by Christopher Skiścim and Bruce Golden, who also illustrated the difficulty of cooling down slowly on the one hand, but not *too* slowly on the other [24]. A great number of cooling schedules have been tested throughout history, varying from linear temperature decrease to schedules that also allow reheating [9]. The logarithmic schedule by Geman & Geman however, is the only one that has theoretically been proven to reach a global optimum [13][21]. But the proof critically relies on the number of iterations going to infinity, so aptly elaborated on by Kenneth Boese and Andrew Kahng: a 'where-you-are' schedule like Geman & Geman's, which needs infinite computation time, is practically useless [9]. This is the critical mistake Paauw and Van den Berg made on paintings-from-polygons, and even though they *did* retain the 'best-so-far' constellations throughout the runs, their intermediate and final results from simulated annealing were of abominable quality compared to their other two algorithms[22]. According to the authors, its failure is "easily explained from its high temperature", a claim that immediately warrants a follow-up investigation. In this study, we will try nine different cooling schedules, see to what extend the authors were right, and whether any improvement is possible.

For now, we specifically do not address the other two algorithms from the earlier study, and with good reason: PPA has shown to be largely parameter insensitive and unbiased ([17][16][31], ongoing work), and the stochastic hillclimber's behaviour is so closely akin to simulated annealing with Geman&Geman on $c = 1$, that a separate quantitative parametrization analysis seems somewhat superfluous. A future study in *qualitative* parameters however (i.e. the algorithmic components), is still an open avenue waiting to be explored.

## 3 Simulated Annealing on Paintings-from-Polygons

A simulated annealing run on the paintings-from-polygons problem starts with randomly distributing $v$ vertices over $p$ polygons ($v = 1000, p = 250$ in this study). Each polygon is first assigned three vertices, after which the remaining vertices are randomly distributed. Then, all the vertices are assigned random coordinate values within the dimensions of the target bitmap. Every polygon is randomly assigned color and opacity (eq.: 'alpha') values within the bounds of its four RGBA-channels. Each iteration, a randomly selected mutation is performed on the polygon constellation. There are four different mutations types, each of which has equal probability of occurring:

1. The **change color** mutation randomly chooses one polygon and one of its RGBA channels, and assigns it a new random value $0 \leq q \leq 255$.
2. The **move vertex** mutation randomly chooses one vertex from one polygon and assigns it new random values $0 \leq x < xMax$ and $0 \leq y < yMax$.
3. The **transfer vertex** mutation randomly chooses two polygons, $p1$ and $p2$, in which $p1$ is not a triangle. A randomly selected vertex is then deleted from $p1$, and a new vertex is placed somewhere on the line between two neighboring vertices in $p2$.
4. The **change drawing index** mutation randomly chooses a polygon and assigns its drawing index a new random value $0 \leq q < p$. If the new index is lower than the current index, all indexes of the in-between polygons will be increased by one. If the new index is higher than the current index, all indexes of the in-between polygons above the new index will be decreased by one.

After each iteration, the new constellation is rendered and the new MSE is calculated. If the new constellation has a lower MSE (equiv.: a smaller pixel-by-pixel difference with the target bitmap), the mutation is accepted and this new constellation will be the starting point for the next iteration. Mutations that lead to a constellation with a higher MSE are not immediately rejected though. Analogous to the annealing metaphor, they might still be accepted, depending on the magnitude of the difference and the temperature. The acceptance probability: $P(accept)$ is given by the Boltzmann factor

$$P(accept) = e^{(\frac{-\Delta MSE}{T_i})} \tag{2}$$

in which $\Delta MSE$ is the difference in MSE between the new and the old constellation, and $T_i$ is the temperature at iteration $i$ retrieved from the cooling schedule. Notice the correlation between the MSE difference and the acceptance probability: a greater increase in MSE is still acceptable as long as the temperature $T_i$ is still high. Correct parameterization of the cooling schedule is therefore of critical importance to both annealing and simulated annealing. For PFP however, it appears as average the temperature *alone* predicts an upper bound on the performance, irrespective of the actual trajectory a schedule might traverse.

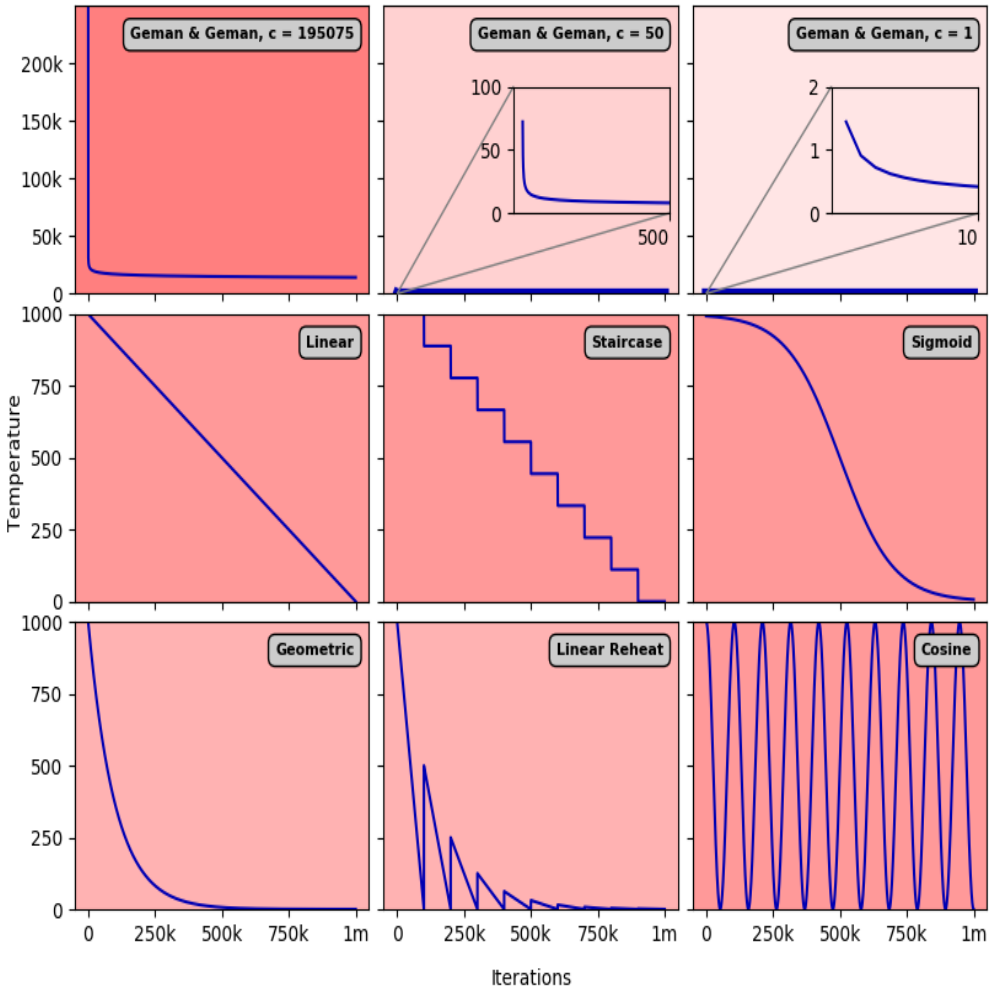


Fig. 2: The course of the temperature for different cooling schedules. The redder the figure, the higher its average temperature. Note that the vertical scale for the top row is much larger, due to the extremities in the Geman & Geman schedule.

## 4 Cooling Schedules

The first three cooling schedules used in this study are of the **Geman & Geman** type, in which the temperature at iteration $i$ is given by

$$T_i = \frac{c}{log(i+1)}. \tag{3}$$

In the seminal study, its parameter was set to $c = 195075$, as it is the maximal existable error barrier that separates two local minima in this problem [22]. In

this study, we replicate the experiment with its original settings, but also add two new parameter settings for this schedule: $c = 50$ and $c = 1$. The number of iterations $i = 10^6$, lowering the temperature from $T_1 \approx 648025$ to $T_{10^6} \approx 32512$ for $c = 195075$. For $c = 50$, the temperature descends from $T_1 \approx 166$ to $T_{10^6} \approx 8$, and for $c = 1$, it ranges from $T_1 \approx 3$ to $T_{10^6} \approx 0.17$ (Fig. 2).

In the **linear** cooling schedule, the temperature $T_i$ is defined as:

$$T_i = T_1 \cdot (1 - \frac{i}{i_{max}}) \tag{4}$$

where in this study $i_{max} = 10^6$ and $T_1 = 1000$, resulting in a linear decrease of temperature throughout the iterations (Fig. 2). The **staircase** cooling schedule also starts $T_1 = 1000$, but drops its temperature once every $10^5$ iterations, resulting in a temperature of 0 for the final $10^5$ iterations. The motivation for the staircase cooling schedule stems from a 2016-publication by Strobl and Barker, who observed that temperatures should be kept constant for some duration of time in order to reach thermal equilibrium [28].

In the **sigmoidal** cooling schedule, temperature follows a sigmoidal decrease given by

$$T_i = \frac{1000}{1 + e^{10^{-5} \cdot (i - 5 \cdot 10^5)}} \tag{5}$$

This cooling schedule decreases slowly in temperature during the run, with its steepest decline, its maximum absolute derivative, at $5 \cdot 10^5$ iterations. It comes from a website with cooling schedules once maintained by Brian T. Luke. It appears to have gone offline[6], but the schedule (and the author) can still be found in a set of public slides at Stanford University [3]. The **geometric** cooling schedule has a history of giving "satisfactory results" on a wide variety of optimization problems [27]. Its temperature $T_i$ is defined as:

$$T_{i+1} = \beta \cdot T_i \tag{6}$$

where in this study, $\beta$ is fixed at 0.99999, with $T_1 = 1000$.

In a study on best-so-far solutions by Boese and Kahng, optimal cooling schedules were shown to be non-monotonically decreasing [9]. As the simulated annealing algorithm used on PFP study also retains its best-so-far value, periodical reheat might provide an alluring alternative to more traditional schedules. The **linear reheat** cooling schedule linearly lowers its temperature to zero in epochs of $10^5$ iterations, but then reheats to half the previous epoch's temperature, as can be seen in figure 2. This results in nine reheating cycles, with their respective peaks being $T_i = \{1000, 500, ..., 3.90625\}$ at $i = \{0, 10^5, ..., 9 \cdot 10^5\}$. The **cosine** cooling schedule follows a simple cosine function, defined as

$$T_i = 500 \cdot cos(\frac{i}{16753}) + 500. \tag{7}$$

[6] BTL's cooling schedules can still be found through the waybackmachine [2].

The fraction $\frac{i}{16753}$ is used to ensure 9.5 cycles in $10^6$ iterations (notice that $16753 \approx \frac{10^6}{2\pi * 9.5}$) and the constant term +500 at the end of the equation ensures the oscillating function starts at $T_1 = 1000$ to and ends at $T_{10^6} = 0$.

## 5 Experiments and Results

The performance of the different cooling schedules is evaluated by performing five runs of simulated arnealing with each cooling schedule on each of the nine different 180x240 target bitmaps: "Salvator Mundi", "Lady with an Ermine" and "Mona Lisa" painted by Leonardo Da Vinci between 1490 and 1503, "The Starry Night" by Van Gogh in 1889, Portrait of composer Johann Sebastian Bach by Elias Gottlieb Haussmann in 1746, "The Kiss" by Gustav Klimt in 1908, "Composition with Red, Blue and Yellow" by Mondriaan in 1930, "The Persistence of Memory" by Dali in 1931, and "Convergence" by Pollock in 1952. Each run consisted of $10^6$ iterations[7], with a nonchanging total of 250 polygons and 1000 vertices (the experiment's source code and data are publicly accessible [4]). For every combination of cooling schedule and target bitmap, the normalized average performance is calculated as the mean final MSE of the five runs

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{8}$$

in which $x$ is the average MSE of the five runs after $10^6$ iterations, $x_{min}$ the lowest MSE of the five, and $x_{max}$ the highest. Notice that a low normalized MSE corresponds to a high performance.

After completing all 405 runs, a firm pattern in the performance of the cooling schedules emerges (Fig. 3). The cooling schedule of Geman & Geman with $c = 1$ performs best on all paintings, immediately followed by the geometric, linear reheat and Geman & Geman with $c = 50$. These top four best performing cooling schedules form a group with relatively similar performance when compared to the other cooling schedules. Fifth, sixth and seventh are the sigmoidal, linear and cosine cooling schedules, followed by the poor performance of the staircase. Very much in last place is the high $c$-valued Geman & Geman cooling schedule used in the study of Paauw and Van den Berg [22]. It is by far the worst possible choice out of these nine.

The performance hierarchy of the cooling schedules can be characterized more rigorously then by its final MSE order alone. It turns out that seven of nine cooling schedules' average log temperatures and final MSEs are bound by a linear equation[8] for each painting, with the error of fit between 0.11 and 0.04 on all paintings. Two schedules however, are different. These are the linear reheat

[7] "[Function evaluations are the gold standard for measuring the performance of heuristic algorithms]" [7], but in our case, the number iterations corresponds exactly to the number of evaluations, which is typical for simulated annealing.

[8] polynomial equations on nonlog data yield comparable results.

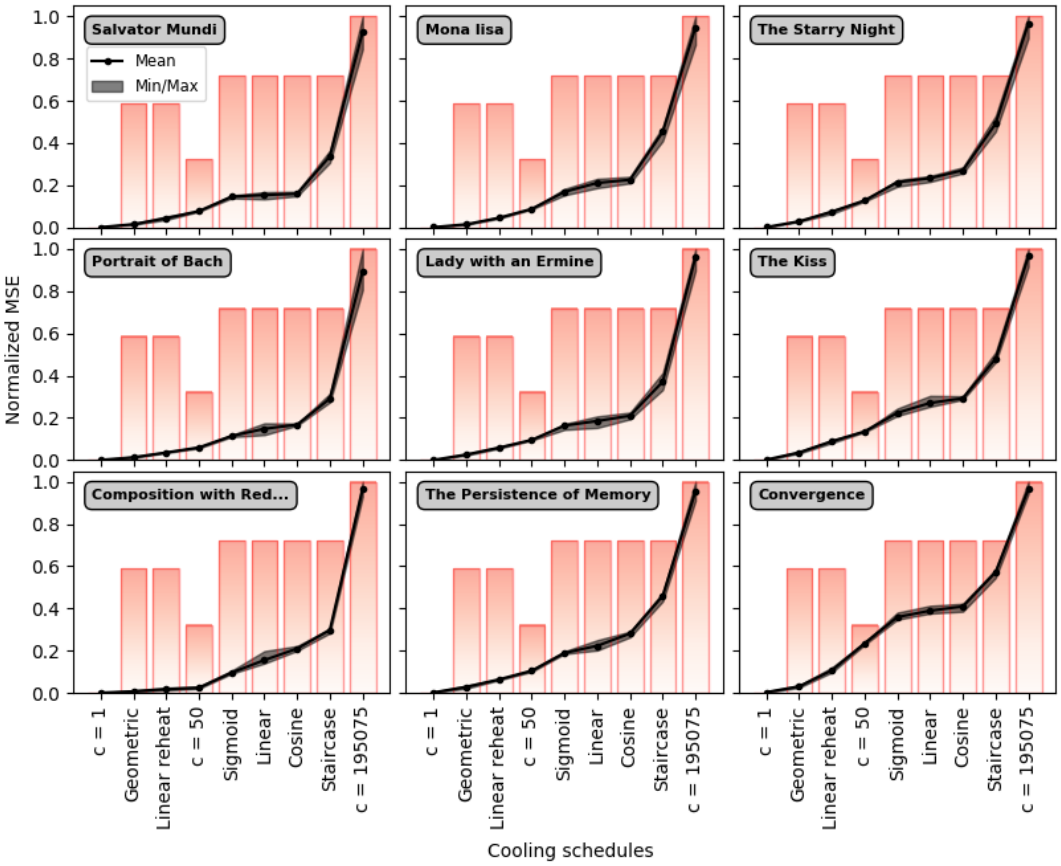


Fig. 3: Normalized mean end results for all nine cooling schedules reveal a consistent, painting invariant performance hierarchy for simulated annealing cooling schedules on the paintings-from-polygons problem. The red bars are the logarithmically plotted average temperatures of the corresponding schedules on the horizontal axes.

schedule, with an error of fit 6 to 11 times greater, and the geometric cooling schedule with an error of fit 6 to 44 times greater, depending on the specific painting. The remarkable thing is however, these errors are larger in *downward* direction.

In other words, when the final MSE and the temperature are compared, all schedules show roughly the same performance, independent of their actual shape. The only two exceptions are the linear reheat and geometric cooling schedules, which perform better than other schedules with the same average temperature. A future experiment, with all cooling schedules normalized to the same average temperature, should either confirm or disconfirm whether these differences are substantial.

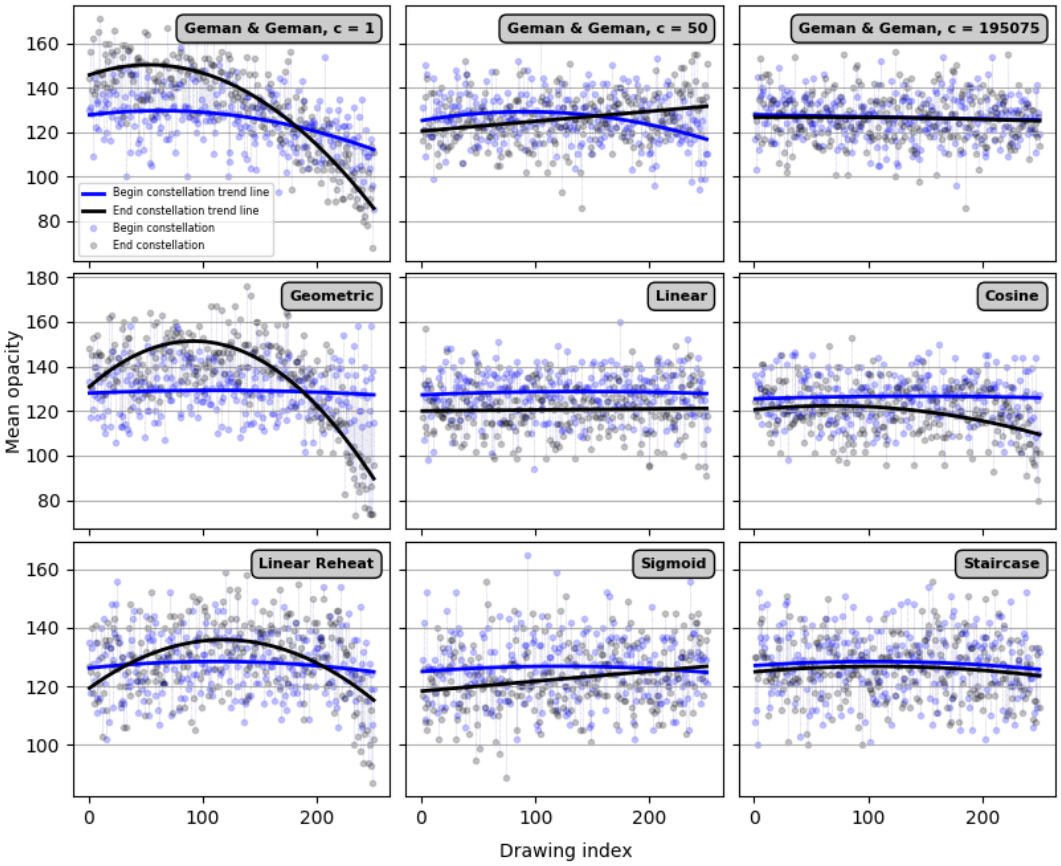


Fig. 4: Opacity per polygon drawing index averaged over 45 runs (nine paintings, five runs each) per schedule. Remarkably enough, the three best performing cooling schedules show a mean parabolic curvature increase, suggesting that good approximations have low-opacity polygons on top.

Another remarkable phenomenon can be witnessed in the structural development of the polygon constellations of this experiment. After random initialization, a typical constellation[9] holds about 111 triangles, 71 quadrangles, 40 pentagons, 18 hexagons and smaller numbers of 7-gons to to 12-gons. But a typical end constellation has up to 28% fewer quadrangles, penta- and hexagons, and very small numbers of larger polygons, up to as many as 35 vertices. Most striking however, is that the number of triangles *increases* by about 16%, to roughly 128. It is unknown what causes this increase, but it is thought that tri-

[9] Here, 'a typical constellation' means: values averaged over all constellations in the experiment.

angles 'facilitate' optimization through their relative mobility in vertex position. This however, still requires further confirmation.

A final interesting phenomenon is that the three best performing cooling schedules appear to exert an influence on the relation between a polygon's opacity and its drawing index. For the three best performing cooling schedules, a higher polygon drawing index corresponds to a lower opacity (Fig. 4). When fitting a parabola, its steepness increases 3 to 24 times for the best three cooling schedules, suggesting that succesful optimization runs produce more transparent polygons on top. Again, further analysis could (dis)confirm the consistency of this effect .

## 6 Conclusion

It appears that Paauw and Van den Berg were right in their assessment that the poor performance of their simulated annealing was due to the high temperatures used at the time. As shown in this study, the success of the algorithm depends largely on the average temperature of the schedule *only*, a few interesting exceptions aside. Furthermore, successful runs seem to increase the number triangles in a constellation, and decrease the opacity of its lastly drawn polygons. To what extent these properties are more typical to the problem or to simulated annealing itself still remains a subject for future endeavours.

## References


1. `https://www.youtube.com/watch?v=u91gRGY8ElQ` (Last accessed on May 23rd, 2020)
2. `http://web.archive.org/web/20190620071317/http://www.btluke.com/simanf1.html` (Last accessed on May 23rd, 2020)
3. `https://statweb.stanford.edu/~candes/teaching/stats318/Handouts/Annealing.pdf` (Last accessed on May 23rd, 2020)
4. `https://github.com/RedRedouane/Paintings-from-Polygons-Simulated-Annealing` (Last accessed on May 23rd, 2020)
5. Abramson, D., Krishnamoorthy, M., Dang, H., et al.: Simulated annealing cooling schedules for the school timetabling problem. Asia Pacific Journal of Operational Research **16**, 1–22 (1999)
6. Banal, S.L., Ciesielski, V.: A deep learning neural network for classifying good and bad photos. In: International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar). pp. 1–16. Springer (2020)
7. Bartz-Beielstein, T., Doerr, C., Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., Lopez-Ibanez, M., Malan, K.M., Moore, J.H., et al.: Benchmarking in optimization: Best practice and open issues. arXiv preprint arXiv:2007.03488 (2020)
8. van den Berg, D.: Simplified paintings-from-polygons is np-hard. Evo* 2020 p. 15 (2020)
9. Boese, K.D., Kahng, A.B.: Best-so-far vs. where-you-are: implications for optimal finite-time annealing. Systems & control letters **22**(1), 71–78 (1994)

10. Černỳ, V.: Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of optimization theory and applications **45**(1), 41–51 (1985)
11. Colton, S., McCormack, J., Berns, S., Petrovskaya, E., Cook, M.: Adapting and enhancing evolutionary art for casual creation. In: International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar). pp. 17–34. Springer (2020)
12. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Berlin, first edn. (2003). https://doi.org/10.1007/978-3-662-05094-1
13. Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on pattern analysis and machine intelligence (6), 721–741 (1984)
14. Heerde, F., Vatolkin, I., Rudolph, G.: Comparing fuzzy rule based approaches for music genre classification. In: International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar). pp. 35–48. Springer (2020)
15. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. Operations research **37**(6), 865–892 (1989)
16. de Jonge, M., van den Berg, D.: Parameter sensitivity patterns in the plant propagation algorithm. (submitted at the time of writing)
17. de Jonge, M., van den Berg, D.: Plant propagation parameterization: Offspring & population size. Evo* 2020 p. 19 (2020)
18. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.: Optimization by simulated annealing. Science **220(4598)**, 671–680 (1983). https://doi.org/10.1126/science.220.4598.671
19. Muehlbauer, M., Burry, J., Song, A.: An aesthetic-based fitness measure and a framework for guidance of evolutionary design in architecture. In: International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar). pp. 134–149. Springer (2020)
20. Neumann, A., Alexander, B., Neumann, F.: Evolutionary image transition and painting using random walks. Evolutionary Computation pp. 1–34 (2020)
21. Nourani, Y., Andresen, B.: A comparison of simulated annealing cooling strategies. Journal of Physics A: Mathematical and General **31**, 8373–8385 (1998). https://doi.org/10.1088/0305-4470/31/41/011
22. Paauw, M., Van den Berg, D.: Paintings, polygons and plant propagation. In: International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar). pp. 84–97. Springer (2019)
23. Rutenbar, R.A.: Simulated annealing algorithms: An overview. IEEE Circuits and Devices magazine **5**(1), 19–26 (1989)
24. Skiścim, C.C., Golden, B.L.: Optimization by simulated annealing: A preliminary computational study for the tsp. In: Proceedings of the 15th conference on Winter Simulation-Volume 2. pp. 523–535. IEEE Press (1983)
25. Sleegers, J., van den Berg, D.: Plant propagation & hard hamiltonian graphs. Evo* 2020 p. 10 (2020)
26. Smit, S.K., Eiben, A.E.: Comparing parameter tuning methods for evolutionary algorithms. In: 2009 IEEE congress on evolutionary computation. pp. 399–406. IEEE (2009)
27. Strenski, P.N., Kirkpatrick, S.: Analysis of finite length annealing schedules. Algorithmica **6**(1-6), 346–366 (1991)
28. Strobl, M.A., Barker, D.: On simulated annealing phase transitions in phylogeny reconstruction. Molecular phylogenetics and evolution **101**, 46–55 (2016)

29. Sulaiman, M., Salhi, A., Khan, A., Muhammad, S., Khan, W.: On the theoretical analysis of the plant propagation algorithms. Mathematical Problems in Engineering **2018** (2018)
30. Van Laarhoven, P.J., Aarts, E.H., Lenstra, J.K.: Job shop scheduling by simulated annealing. Operations research **40**(1), 113–125 (1992)
31. Vrielink, W., van den Berg, D.: Fireworks algorithm versus plant propagation algorithm. In: IJCCI. pp. 101–112 (2019)