Figurative Tours and Braids

Robert Bosch
Dept. of Mathematics
Oberlin College
rbosch@oberlin.edu

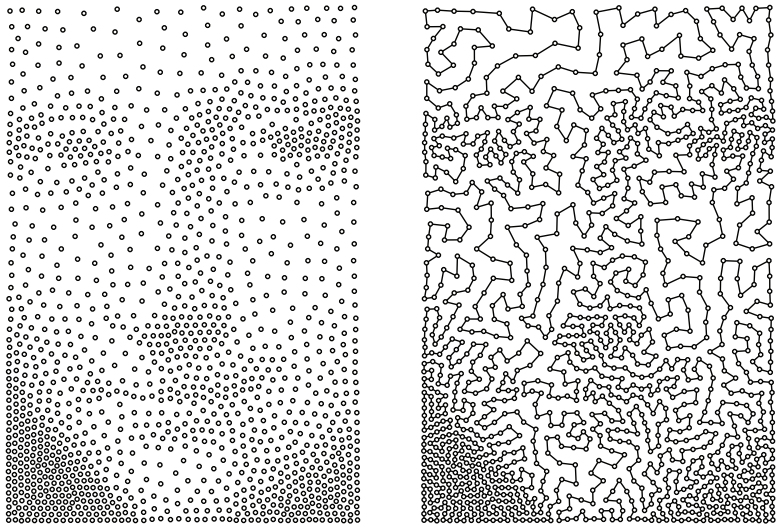
Tom Wexler
Dept. of Computer Science
Oberlin College
wexler@cs.oberlin.edu

Abstract

We start with a rectangular grid of points, and we connect pairs of points to form either a tour (a Hamiltonian cycle) or a braid (a collection of disjoint paths that start in the top row and end on the bottom). In each case, our goal is to design a graph that will closely resemble a grayscale target image when viewed from a distance. From up close, the graph will look like an abstract pattern. We formulate these design problems as integer programming problems.

Introduction

TSP Art [3,7] converts a grayscale target image into a collection of points, then treats the points as the cities of a Traveling Salesman Problem (TSP), and finally constructs a high quality tour that passes through them. If all goes well, the salesman's tour will also resemble the target image. For the example displayed in Figure 1, we used Concorde [4] to find an optimal tour. Even though this tour is the best one possible and does indeed resemble the target image, it doesn't achieve as good a likeness as the points do alone. The reason is simple: the ink required to connect the points introduces visual noise.

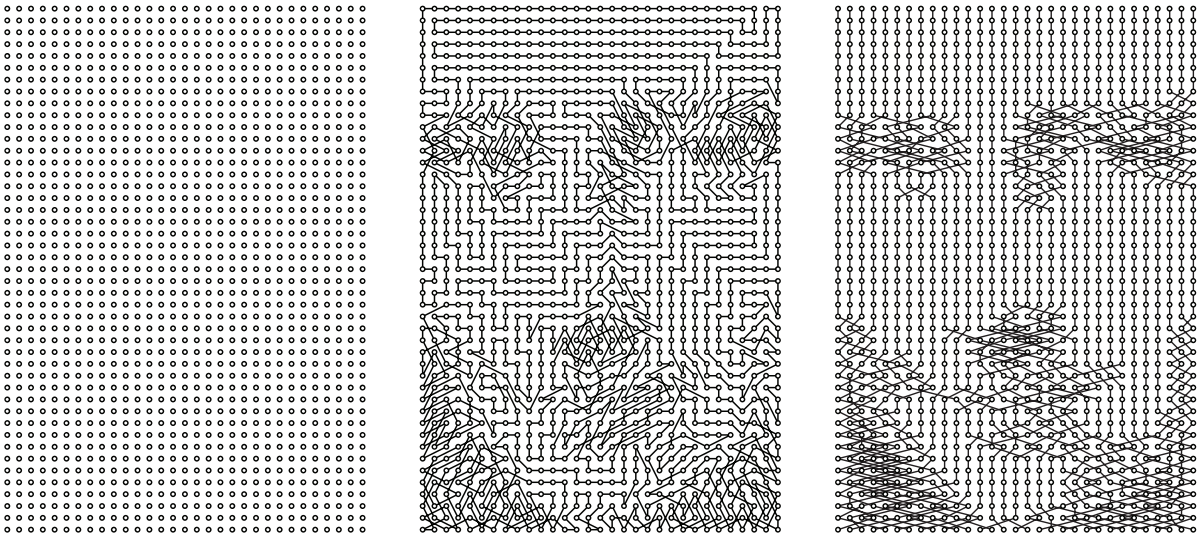


(a) (b)

Figure 1 : *(a) A 1395-city Mona Lisa TSP along with (b) an optimal tour.*

Here, we skip the usual method for positioning the points (stippling [8]), and instead arrange them in a rectangular grid, as in Figure 2(a). Because our point pattern is uniform, the points will look nothing like the target image. From a distance, they will resemble a gray rectangle. Consequently, if we are to have any success in producing a recognizable likeness of the target image, it will be entirely due to the choices we make when connecting the points (i.e., selecting the edges).

In the next few sections, we describe our approach to solving the *Figurative Tour Problem* (FTP). In the FTP, we are given a target image, a grid of points, and a set of possible edges. Our goal is to find the best figurative tour, a Hamiltonian cycle that most closely resembles the target image. Figure 2(b) displays a high quality (recognizable, but not necessarily optimal) figurative tour.



(a) (b) (c)

Figure 2 : *A grid of* 1395 *points (a) and a figurative tour (b) and figurative braid (c) that resemble Leonardo da Vinci's Mona Lisa.*

We then discuss a related problem, the *Figurative Braid Problem* (FBP). Given a target image and a grid of points, we must join the points into disjoint paths that start in the top row of the grid and end on the bottom. Again, our goal is to design a graph (here, a braid) that most closely resembles the target image. Figure 2(c) displays a figurative braid.

The Figurative Tour Problem

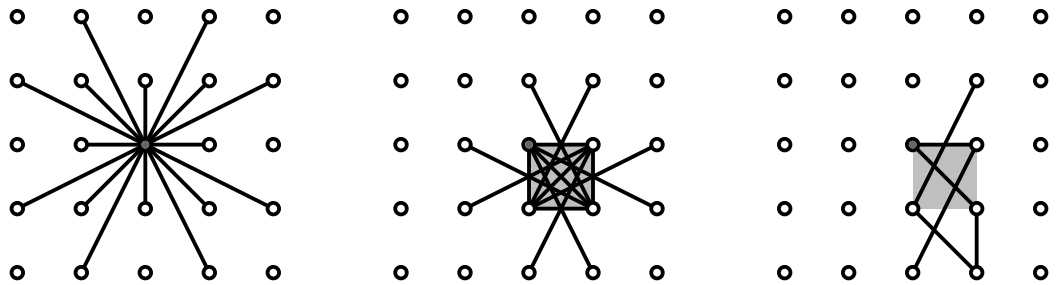
We start with a grayscale target image $\mathcal{I}$ and use vertical and horizontal lines to partition it into m rows and n columns of square blocks of pixels. If necessary, we crop the image. For each block of pixels we compute the average brightness $\beta_{i,j}$ of the pixels in the block, using a 0-to-1 black-to-white scale. We index rows with i and columns with j. We will build our tour upon a $(m+1)$-row by $(n+1)$-column grid of points, each positioned at the corner of one or more blocks. In Figure 2(a), $m = 44$ and $n = 30$.

We let E denote the set of possible edges (connections). For the figurative tour displayed in Figure 2(b), we limited ourselves to orthogonal King's moves, diagonal King's moves, and Knight's moves. That is, if each block is 1-by-1, we only allowed ourselves to use edges of length 1, $\sqrt{2}$, and $\sqrt{5}$.

We let $P_{i,j}$ denote the set of edges that are incident to point (i, j), and $B_{i,j}$ denote the set of edges that pass through or share a side with block (i, j), the block whose upper left corner is point (i, j). Figures 3(a) and 3(b) display these sets. For each block (i, j) and each edge $e \in B_{i,j}$, we need a measure of the amount of ink that the salesman tracks through block (i, j) if he traverses edge e in his tour. We denote this $\tau_{e,i,j}$ and call it the *trace of edge* e *on block* (i, j).

We found it convenient to have these $\tau_{e,i,j}$ values be integers. After considerable experimentation, we set $\tau_{e,i,j} = 12$ for orthogonal edges, $\tau_{e,i,j} = 35$ for diagonal edges, and $\tau_{e,i,j} = 28$ for Knight's move edges. These values are closely linked to the edge lengths. Each one is approximately 25 times the length of the

part of the edge that lies within the block. (We consider half of an orthogonal edge to lie in the block and the other half to lie in a neighboring block. In the diagonal case, the entire edge lies in the block. For a Knight's move edge, half of the edge lies in the block, and the other half lies in a neighboring block.)



(a) (b) (c)

Figure 3: *(a) The set of edges $P_{i,j}$ that are incident to the (shaded) point (i, j), (b) the set of edges $B_{i,j}$ that pass through the (shaded) block (i, j), and (c) a portion of a tour for which the trace on block (i, j) is $t_{i,j} = 103$.*

With the $\tau_{e,i,j}$ values, we can measure the total amount of ink that the salesman tracks through each block (i, j). We denote this $t_{i,j}$ and call it the *trace of the tour on block* (i, j). The smallest possible value for $t_{i,j}$ is 0. This happens when the salesman does not pass through the block. The next smallest possible value for $t_{i,j}$ is 12. This happens when the salesman travels along precisely one of the block's orthogonal edges and traverses none of its other edges. There are numerous other possible values for $t_{i,j}$. To ensure that a viewer can visually follow the tour, the largest $t_{i,j}$ we consider is 103. This happens when the salesman uses one of the block's orthogonal edges, one of its diagonal edges, and two of its Knight's move edges, as shown in Figure 3(c) ($12 + 35 + 2 \cdot 28 = 103$).

From $t_{i,j}$, we easily obtain $b_{i,j}$, the brightness of block (i, j) in the tour, measured on a 0-to-1 black-to-white scale. We simply set $b_{i,j} = 1 - 0.01\, t_{i,j}$. As the brightnesss of block (i, j) in the image $\mathcal{I}$ is $\beta_{i,j}$, the goal of the Figurative Tour Problem is to find a tour for which the $b_{i,j}$ values are as close to the $\beta_{i,j}$ values as possible. We therefore strive to minimize find the tour that minimizes the function

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \left(\beta_{i,j} - [1 - 0.01\, t_{i,j}]\right)^2 . \tag{1}$$

An Integer Programming Formulation of the FTP

For each $e \in E$, we let x_e be an indicator variable which is 1 if we use edge e in the tour, and 0 otherwise. To ensure that point (i, j) is both entered and exited by the tour, we include the "degree" constraint

$$\sum (x_e \,:\, e \in P_{i,j}) \,=\, 2. \tag{2}$$

To measure the trace the tour leaves on block (i, j), we include the equation

$$t_{i,j} = \sum (\tau_{e,i,j} x_e \,:\, e \in B_{i,j}). \tag{3}$$

A nonlinear integer programming formulation would minimize the objective function (1) subject to a copy of equation (2) for each point (i, j) and a copy of equation (3) for each block (i, j). Our approach

employs a standard integer programming (IP) modeling technique [9] to linearize (1). To do this, we introduce an indicator variable $y_{i,j,k}$ that is 1 if $t_{i,j} = k$ and 0 otherwise. This enables us to rewrite the quadratic function (1) as the linear function

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k}\left(\beta_{i,j} - [1 - 0.01\,k]\right)^2 y_{i,j,k}.$$

Note that the nonlinearity is now in the data, not the variables. To ensure that $y_{i,j,k}$ assumes the correct value of the trace, we impose two linear equality constraints:

$$\sum_{k} y_{i,j,k} = 1 \tag{4}$$

and

$$\sum_{k} k\, y_{i,j,k} = t_{i,j}. \tag{5}$$

Equation (4) states that $y_{i,j,k}$ must be assigned precisely one of the possible values for the trace (0, 12, 24, 28, 35, and so on up to the maximum value we allow, 103). All summations involving k are taken over just these values. Equation (5) forces the trace to be $t_{i,j}$.

Taken together, our IP formulation of the FTP is as follows:

minimize	$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k}\left(\beta_{i,j} - [1 - 0.01\,k]\right)^2 y_{i,j,k}$	
subject to	$\sum(x_t \,:\, e \in P_{i,j}) = 2$	for each point (i,j)
	$\sum_k y_{i,j,k} = 1$	for each block (i,j)
	$\sum_k k\, y_{i,j,k} = \sum(\tau_{e,i,j}\, x_e \,:\, e \in B_{i,j})$	for each block (i,j)
	$x_e \in \{0,1\}$	for each $e \in E$
	$y_{i,j,k} \in \{0,1\}$	for each k and block (i,j).

This IP formulation does not prevent subtours. To eliminate subtours, we follow a strategy similar to one used by TSP researchers [1,2,5]. We hope that we will not encounter subtours, but if we do, we will impose linear constraints that prohibit the ones we have found. We then re-solve the IP. We repeat this process until there are no more subtours.

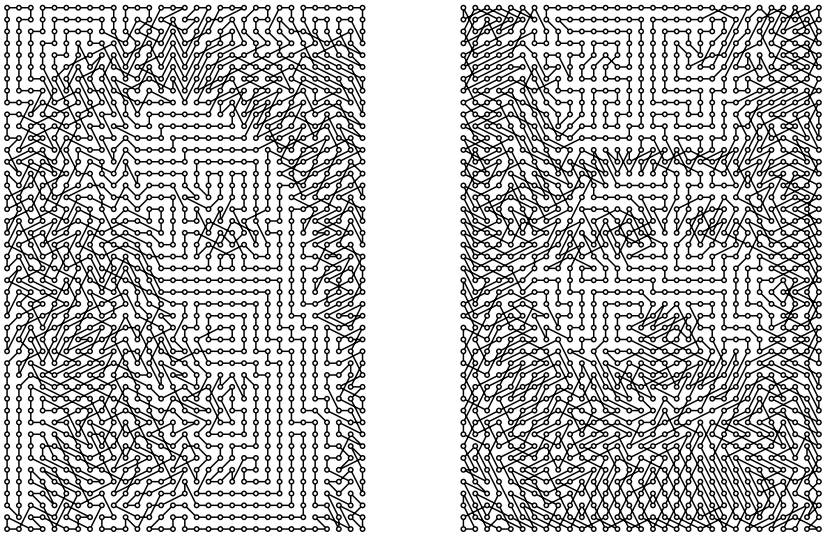
Discussion

It turns out that our IP model is very difficult to solve. We cannot solve it to optimality with the state-of-the-art solver Gurobi [6]. To obtain our figurative tours, we use an IP-based local search heuristic. Starting from *any* Hamiltonian cycle of the points, we allow ourselves to modify only those variables that correspond to points or blocks that fall within a small (e.g. 8×10 or 10×8) search window. We keep the other variables fixed. This gives us smaller IPs that Gurobi can solve to optimality in a relatively short amount of time (often less than one minute, but occasionally taking up to 20 minutes). After making a first pass—positioning the seach window in each possible location—we make a second pass, a third, and so on. We stop after seeing no improvement on a pass.

We then begin the second phase, the elimination of subtours. If our figurative tour has subtours, we position the search window so that it contains portions of more than one subtour, we include linear constraints that will eliminate these subtours (as in the TSP), and then use Gurobi to solve the resulting IP. We repeat until there are no more subtours.

For the figurative tour shown in Figure 2(b), we imposed 36 subtour elimination constraints. It is unlikely that the tour is optimal, but we do know that its objective function value is within 22.0 percent

of a lower bound on the optimal value. For the figurative tours shown in Figure 4(a) and Figure 4(b), we imposed 47 and 51 subtour elimination constraints respectively. Their objective function values are within 13.1 percent and 6.8 percent of their respective lower bounds. We obtained the lower bounds by instructing Gurobi to use branch and bound to solve the full IP models together with all of the subtour elimination constraints we had imposed to obtain our tours. In each case, we terminated Gurobi's branch-and-bound run after 5000 seconds.

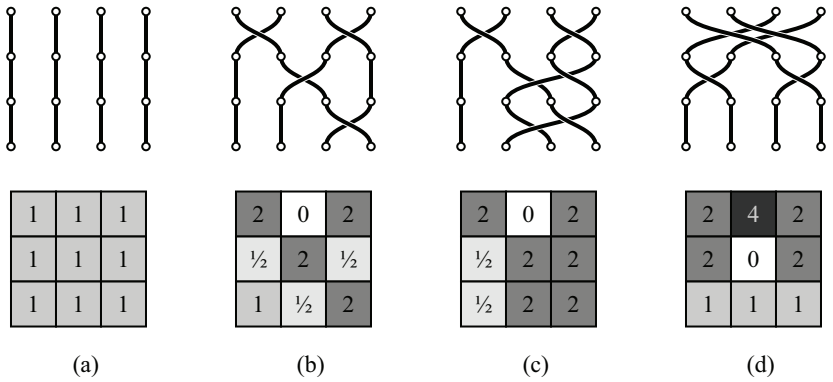


(a) (b)

Figure 4 : *Figurative tours of (a) Sandro Botticelli's Venus and (b) Graham Chapman's King Arthur.*

The Figurative Braid Problem

We now explore an alternate way of connecting the points in the $(m+1)$ by $(n+1)$ grid. Consider a *trivial braid*: $n+1$ vertical strands of thread. An example with $m = n = 3$ is shown in Figure 5(a). By twisting the strands through various blocks of the grid, we obtain *non-trivial braids*, as in shown in Figures 5(b-d).



(a) (b) (c) (d)

Figure 5 : *A trivial braid (a) and various non-trivial braids (b,c,d). The number of strands $s_{i,j}$ per block (i, j) is displayed below each braid.*

After twisting, some blocks have more crossing strands, while other blocks have fewer. Thus some

blocks appear darker while others appear lighter. The strand count $s_{i,j}$ for each block (i, j) is displayed below each braid. As in the FTP, we count vertical strands as half a crossing strand for the blocks on either side, but unlike the FTP, here we do not consider the length of a strand within a block.

Note that by twisting the strands, we have shifted the uniform darkness value (1) in Figure 5(a) to a range of values (0, $\frac{1}{2}$, 1 and 2) in Figure 5(b). By allowing strands to travel farther horizontally, we can create adjacent dark blocks, as in 5(c), or even darker values, as in 5(d). In this section, we explore the optimization problem of generating a braid that most closely resembles a target image.

As before, we start with a grayscale image $\mathcal{I}$ containing m rows (indexed by i) and n columns (indexed by j) of square blocks of pixels. We compute for each block a darkness value $\delta_{i,j}$ ranging from 0 (white) to 1 (black). Our braid is similarly built upon a $(m+1)$-row by $(n+1)$-column grid of points, each positioned at the corner of one or more blocks in $\mathcal{I}$. We draw $n+1$ strands, each of which intersects exactly one point in every row. No two strands may intersect the same grid point, and thus the strands induce a matching between adjacent rows of points. We also include a parameter Δ bounding the maximum number of columns a strand may pass through while crossing a single row. While a larger value for Δ allows for greater range of tone in the resulting image, smaller Δ values make for more visually "readable" braids, and faster computation as well. In Figure 2(c), $\Delta = 4$.

We define $s_{i,j}$ to be the number of strands crossing block (i, j). The darkness of cell (i, j) is approximately proportional to $s_{i,j}$. Therefore, since 2Δ is the maximum possible value of $s_{i,j}$, we define the darkness of our braid at pixel (i, j) to be $d_{i,j} = \frac{1}{2\Delta} s_{i,j}$. As the actual darkness value of pixel (i, j) in the image $\mathcal{I}$ is $\delta_{i,j}$, our goal is to find a braid for which the $d_{i,j}$'s are as close to the $\delta_{i,j}$'s as possible.

An IP Formulation of the FBP

Since our current problem formulation includes no constraints between rows, we focus on a single row of blocks i and drop i from our notation for clarity. Let $x_{j,j'}$ be an indicator variable which is 1 if point (i, j) is connected by a strand to point $(i + 1, j')$, and 0 otherwise. To ensure that exactly one strand enters and leaves each point, we include constraints $\sum_{j'=1}^{n} x_{j,j'} = 1$ and $\sum_{j=1}^{n} x_{j,j'} = 1$ for all j and j', respectively. The number of strands through block j is

$$s_j = \sum_{j' \le j} \sum_{j'' > j} (x_{j',j''} + x_{j'',j'}).$$

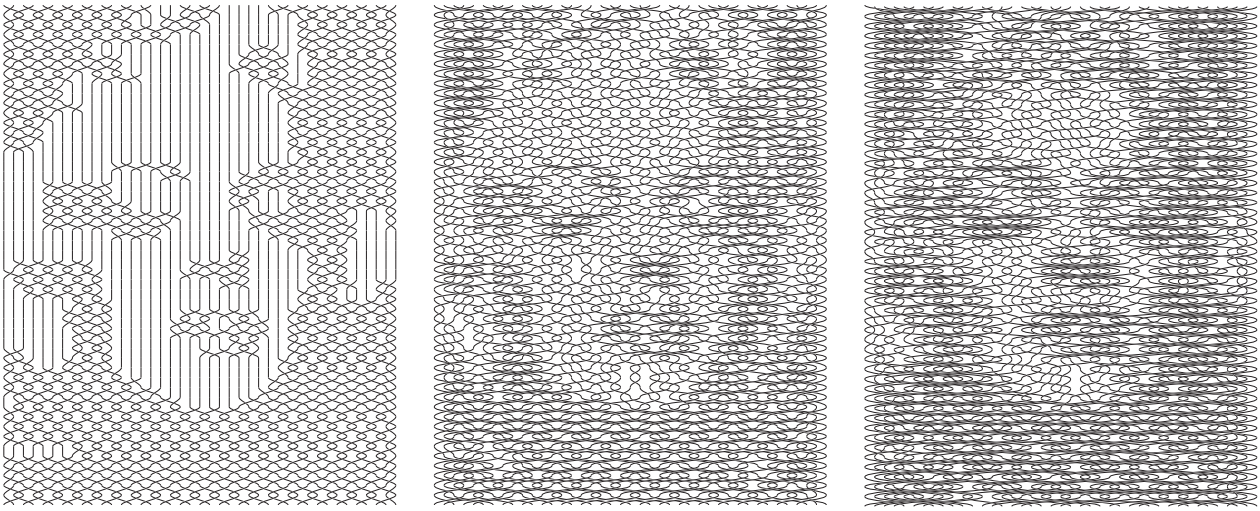
The darkness of our braid at cell j is $d_j = \frac{1}{2\Delta} s_j$. Since our goal is to simultaneously make each d_j value close to δ_j, we opt to minimize the sum of squared differences $\sum_j (\delta_j - d_j)^2$. As with the Figurative Tour Problem, this is not a linear objective function, but it can be linearized in the same way; we introduce an indicator variable $y_{j,k}$ that is 1 if $s_j = k$ and 0 otherwise. The error associated with block j being crossed by k strands is the constant $e_{j,k} = (\delta_j - \frac{k}{2\Delta})^2$. Thus we can rewrite our objective function as

$$\sum_{j,k} e_{j,k}\, y_{j,k}.$$

To ensure $y_{j,k}$ correctly reflects the number of strands through block j, we impose the equality constraints $\sum_k y_{j,k} = 1$ and $\sum_k k\, y_{j,k} = s_j$ for each j. Taken together, our IP formulation for a single row is:

$$\begin{array}{lll}
\text{minimize} & \sum_{j=1}^{n} \sum_{k=0}^{2\Delta} (\delta_j - \frac{k}{2\Delta})^2 y_{j,k} & \\
\\
\text{subject to} & \sum_{j'=1}^{n} x_{j,j'} = 1 & \text{for each } 1 \le j \le n+1 \\
& \sum_{j=1}^{n} x_{j,j'} = 1 & \text{for each } 1 \le j' \le n+1 \\
& \sum_{k=0}^{2\Delta} y_{j,k} = 1 & \text{for each } 1 \le j \le n+1 \\
& \sum_{k=0}^{2\Delta} k\, y_{j,k} = \sum_{j' \le j} \sum_{j'' > j} (x_{j',j''} + x_{j'',j'}) & \text{for each } 1 \le j \le n+1 \\
& x_{j,j'} \in \{0,1\} & \text{for each } 1 \le j \le n+1, 1 \le j' \le n+1 \\
& y_{j,k} \in \{0,1\} & \text{for each } 1 \le j \le n+1, 0 \le k \le 2\Delta.
\end{array}$$

Note that $x_{j,j'}$ variables above can be removed if they correspond to disallowed shifts (i.e. if $|j - j'| > \Delta$). Similarly, if we want to prohibit vertical segments in our strands for aesthetic purposes, we can set $x_{j,j} = 0$. The whole IP consists of one such optimization problem per row i. Using Gurobi [6] as the IP solver on a MacBook Pro laptop, the braids shown here were all produced in under a minute.



(a) (b) (c)

Figure 6: *Marilyn Monroe, 51-by-40, for* $\Delta = 2$ *(a),* $\Delta = 4$ *with no vertical segments (b), and* $\Delta = 6$ *with no vertical segments (c).*

Variations

Additional constraints can be added to forbid long chains of purely vertical segments, which introduces some small dependency between rows. In most of the images shown here we chose to disallow purely vertical segments altogether. The images in Figures 6 and 7 were generated in Postscript using Bezier curves for strands and grid points removed. A range of textures and tones can be achieved by adjusting line thickness and the control points of the Bezier curves.

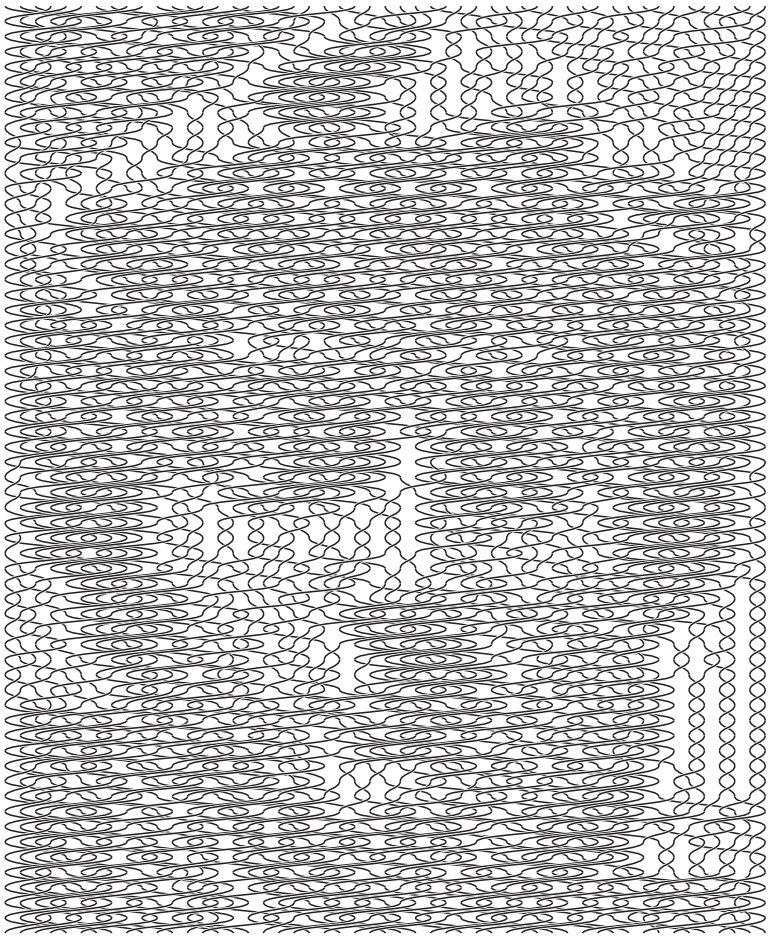


Figure 7 : *Michelangelo's David, 61-by-50, for* $\Delta = 5$ *with no vertical segments.*

References

[1] D.L. Applegate, R.E. Bixby, V. Chvátal, and W.J. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, Princeton, 2006.

[2] R. Bosch. Simple-closed-curve sculptures of knots and links. *Journal of Mathematics and the Arts*. 4(2):57-71, 2010.

[3] R. Bosch and A. Herman. Continuous line drawings via the traveling salesman problem. *Operations Research Letters*. 32(4):302-303, 2004.

[4] Concorde TSP Solver `www.math.uwaterloo.ca/tsp/concorde.html`. Accessed 15 March 2015.

[5] W.J. Cook, *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*, Princeton University Press, Princeton, 2012.

[6] Gurobi — The overall fastest and best solver ... `www.gurobi.com`. Accessed 15 March 2015.

[7] C.S. Kaplan and R. Bosch. TSP Art. In *Bridges Banff: mathematical connections in art, music, and science*, pages 139-146, 2009.

[8] A.Secord. Weighted Voronoi stippling. In *NPAR '02: Proceedings of the 2nd international symposium on non-photorealistic animation and rendering*, ACM Press, New York, pages 37-43, 2002.

[9] H.P. Williams, *Model Building in Mathematical Programming*, Wiley, West Sussex, 2013.