

# Constructing Domino Portraits

Robert Bosch

Department of Mathematics, Oberlin College, Oberlin, OH USA, 44074  
bobb@cs.oberlin.edu

**Abstract.** We use integer programming to construct approximations of target images using complete sets of double nine dominoes.

## 1 Introduction

In 1993 the artist Ken Knowlton created a portrait of *Scientific American's* “Mathematical Games” columnist Martin Gardner out of six complete sets of double nine dominoes [1]. Figure 1 displays portraits of Marilyn Monroe and John Lennon that were made by R. Bosch out of complete sets of double nine dominoes (nine sets per portrait). In this brief note, we describe a new, integer-programming-based method for constructing such portraits.

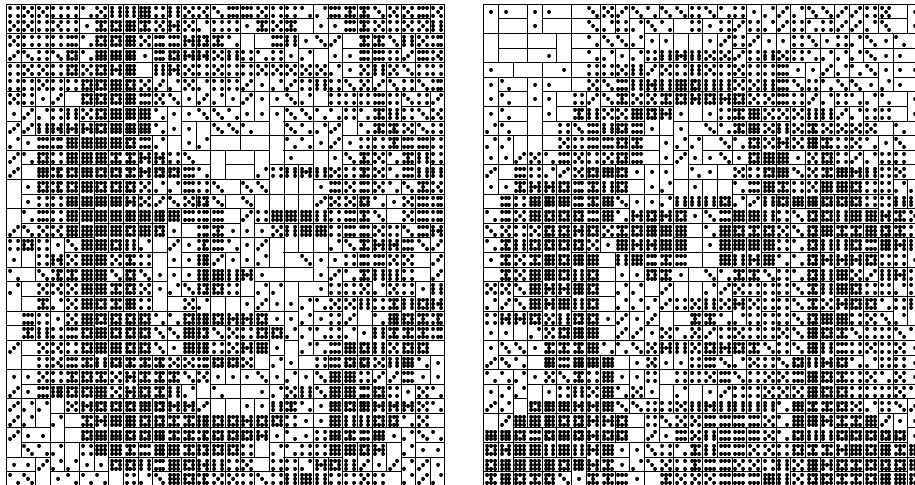


Fig. 1. Marilyn (9 sets) and John (9 sets)

## 2 Dominoes

Figure 2 displays a complete set of double nine dominoes. There are 55 dominoes in a complete set—10 doubles and  $\binom{10}{2} = 45$  non-doubles. Each double has two

orientations:  $v$  (vertical) and  $h$  (horizontal). Each non-double has four orientations:  $v_1$  (vertical with the lower-numbered square on top),  $v_2$  (vertical with the lower-numbered square on the bottom),  $h_1$  (horizontal with the lower-numbered square on the left), and  $h_2$  (horizontal with the lower-numbered square on the right).

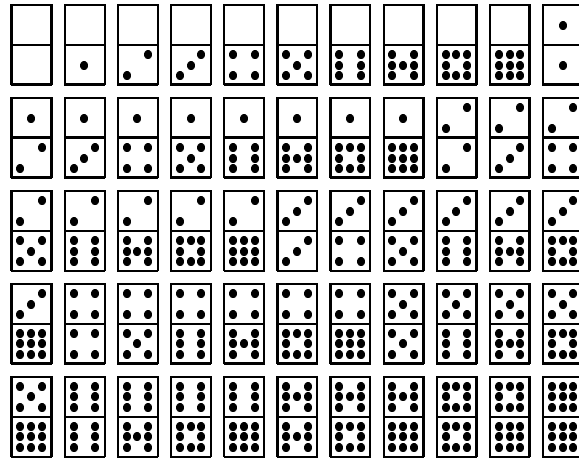


Fig. 2. A complete set of double nine dominoes

Since each domino is made of two squares,  $s^2$  complete sets can be used to create pictures on an  $11s \times 10s$  canvas of squares.

### 3 The Target Image

We first convert the target image into PGM (portable graymap) format. Each pixel of the image is given a grayscale value between 0 (completely black) and 255 (completely white). We then divide the image into  $11s$  rows and  $10s$  columns of  $k \times k$  squares of pixels, one square for each square of our canvas. Finally, for each square, we compute the mean grayscale value and then convert the mean to an integer between 0 (completely white) and 9 (completely black). We then let  $g_{i,j}$  denote the resulting grayscale value of the image's row- $i$ -column- $j$  square.

## 4 An Integer Programming Formulation

### 4.1 Decision Variables

We let  $x_{m,n,o,i,j}$  equal 1 if domino  $(m,n)$  is placed in orientation  $o$  with its top left square in the row- $i$ -column- $j$  square of the canvas and 0 if not. (We keep  $m \leq n$  throughout.) It is easy to show that if the canvas has  $11s$  rows and  $10s$  columns, then the number of decision variables is  $22,000s^2 - 2,100s$ .

## 4.2 Objective Function

We let  $c_{m,n,o,i,j}$  equal the cost of placing domino  $(m,n)$  in orientation  $o$  with its top left square in the row- $i$ -column- $j$  square of the canvas. We measure costs using a 2-norm. For example, if  $o = v_1$ , then  $c_{m,n,o,i,j} = (m - g_{i,j})^2 + (n - g_{i+1,j})^2$ , the sum of penalties we charge ourselves for placing the “ $m$ ” in square  $(i, j)$  and the “ $n$ ” in square  $(i + 1, j)$ . Clearly, our goal is to minimize

$$\sum_{m,n,o,i,j} c_{m,n,o,i,j} x_{m,n,o,i,j}.$$

## 4.3 Constraints

We need two types of constraints. The “type-one” constraint

$$\sum_{o,i,j} x_{m,n,o,i,j} = s^2$$

stipulates that domino  $(m, n)$  is to be used  $s^2$  times. (Recall that we are using  $s^2$  sets of dominoes.) We need 55 type-one constraints—one for each distinguishable domino. The “type-two” constraint

$$\begin{aligned} & \sum_m x_{m,m,v,i,j} + \sum_m x_{m,m,v,i-1,j} \\ & + \sum_m x_{m,m,h,i,j} + \sum_m x_{m,m,h,i,j-1} \\ & + \sum_{m<n} x_{m,n,v_1,i,j} + \sum_{m<n} x_{m,n,v_2,i,j} \\ & + \sum_{m<n} x_{m,n,v_1,i-1,j} + \sum_{m<n} x_{m,n,v_2,i-1,j} \\ & + \sum_{m<n} x_{m,n,h_1,i,j} + \sum_{m<n} x_{m,n,h_2,i,j} \\ & + \sum_{m<n} x_{m,n,h_1,i,j-1} + \sum_{m<n} x_{m,n,h_2,i,j-1} = 1 \end{aligned}$$

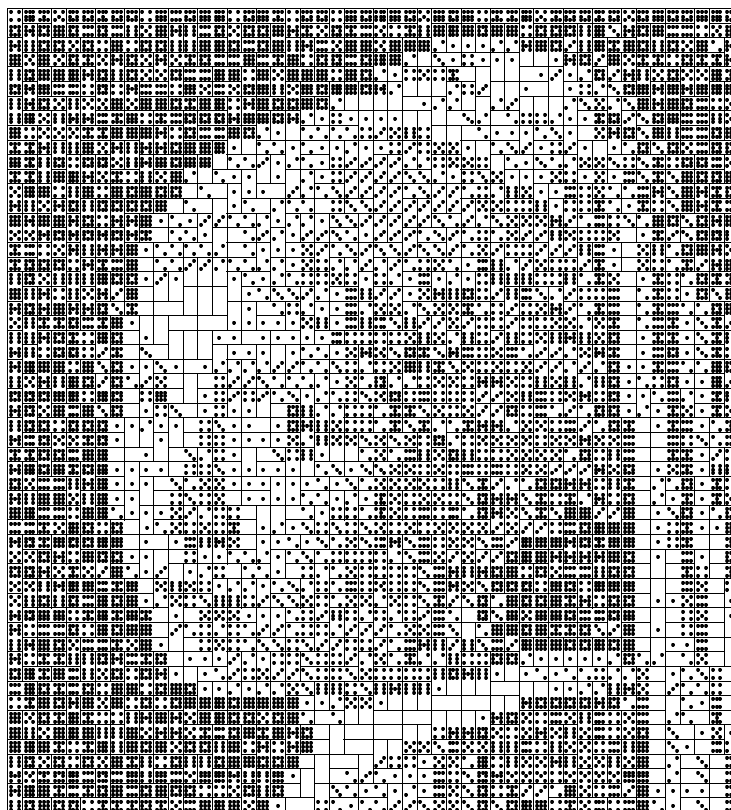
states that the row- $i$ -column- $j$  square of the canvas must be covered by exactly one domino. We need  $110s^2$  type-two constraints—one for each square of the canvas.

## 4.4 Results

The resulting integer programs are quite large. Each “ $s = 3$ ” (9-set) integer program has 191,700 decision variables and 1,405 constraints. Each “ $s = 7$ ” (49-set) integer program has 1,063,300 decision variables and 5,445 constraints.

Fortunately, these integer programs are usually very easy to solve. The IPs used to create the 9-set portraits of Marilyn Monroe and John Lennon (see

figure 1) required 763 seconds and 567 seconds, respectively, and the IP used to create the 25-set approximation of Vermeer's "Girl with a Pearl Earring" (see figure 3) required 3,916 seconds. The IP used to create a 49-set approximation (not shown) of the same Vermeer painting required 15,816 seconds. In each case, the solution to the linear programming relaxation was integer valued. (All computations were performed with CPLEX (version 6.6) on an 800 Mz Pentium III PC.)



**Fig. 3.** Vermeer's Girl with a Pearl Earring (25 sets)

We have created over 50 domino portraits, and in all but three cases the solution to the linear programming relaxation was integer valued. And in each of those three cases, a very high quality feasible solution (one with an objective value within a fraction of a percent of the linear programming bound) was found very quickly.

## References

1. E. BERLEKAMP and T. RODGERS, *The Mathematician and Pied Puzzler: A Collection in Tribute to Martin Gardner*, AK Peters, 1999.