Painting by Numbers

Robert A. Bosch

September 28, 2000

A *paint-by-numbers* puzzle consists of an $m \times n$ grid of pixels (the *canvas*) together with $m + n$ *cluster-size sequences*, one for each row and column. The goal is to paint the canvas with a picture that satisfies the following constraints:

1. Each pixel must be black or white.
2. If a row or column has cluster-size sequence $s_1, s_2, \ldots, s_k$, then it must contain k clusters of black pixels—the first with s_1 black pixels, the second with s_2 black pixels, and so on.

It should be noted that "first" means "leftmost" for rows and "topmost" for columns, and that rows and columns need not begin or end with black pixels.

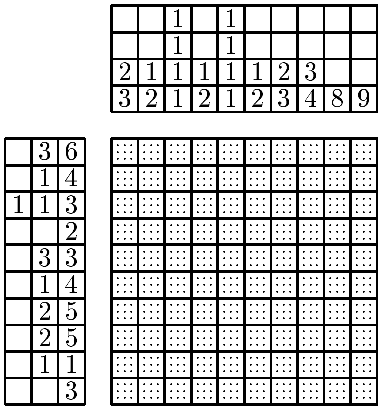

Figure 1

Small paint-by-numbers puzzles are easy to solve. One useful strategy is to alternate between analyzing the row cluster-size sequences and the column cluster-size sequences. For the puzzle displayed in Figure 1, a first pass through the rows enables us to paint 15 pixels black and one pixel white, as displayed in Figure 2. (Since row 1's cluster-size sequence is $3, 6$, its first three pixels must be black, its fourth pixel must be white, and its final six pixels must be black. Since rows 7 and 8 have cluster-size sequence $2, 5$, their second clusters must occupy pixels 4 through 8, 5 through 9, or 6 through 10; in each case they occupy pixels 6, 7, and 8.) A subsequent pass through the columns enables us to paint 19 more pixels black and 19 more pixels white, as displayed in Figure 3.

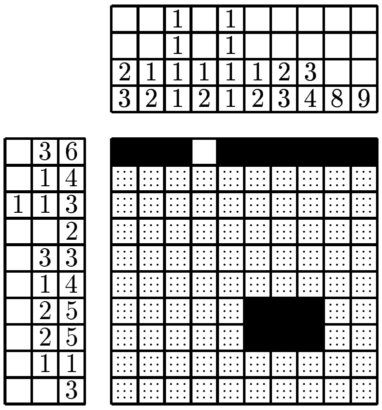

Figure 2

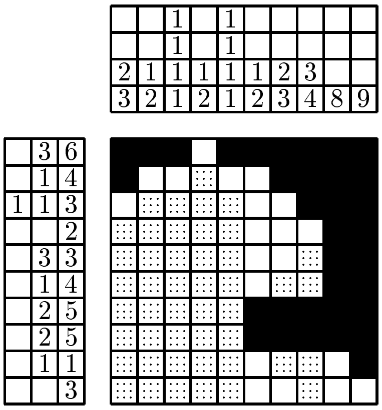

Figure 3

Four more passes through the rows and columns enable us to paint the remaining 46 pixels. The unique solution, a picture of a spaceman, is displayed in Figure 4. It should be noted that only well designed paint-by-numbers puzzles have unique solutions.

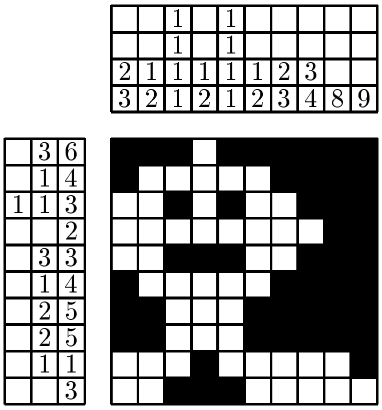

Figure 4

An IP Formulation

In this section, we describe how integer programming can be used to solve paint-by-numbers puzzles. Our approach is to think of a paint-by-numbers puzzle as a problem comprised of two interlocking tiling problems: one involving the placement of the row clusters on the canvas, and the other involving the placement of the column clusters. Note that if a pixel is painted black, it must be covered by both a row cluster and a column cluster; if it is painted white, it must be left uncovered by the row clusters and column clusters.

Notation

Let

$$
\begin{aligned}
m &= \text{the number of rows,}\\
n &= \text{the number of columns,}\\
k^r_i &= \text{the number of clusters in row } i,\\
k^c_j &= \text{the number of clusters in column } j,\\
s^r_{i,1}\,,\ s^r_{i,2}\,,\ \ldots\,,\ s^r_{i,k^r_i} &= \text{the cluster-size sequence for row } i,\\
s^c_{j,1}\,,\ s^c_{j,2}\,,\ \ldots\,,\ s^c_{j,k^c_j} &= \text{the cluster-size sequence for column } j.
\end{aligned}
$$

In addition, let

$$
\begin{aligned}
e^r_{i,t} &= \text{the smallest value of } j \text{ such that row } i\text{'s } t^{\text{th}} \text{ cluster can be}\\
&\qquad \text{placed in row } i \text{ with its leftmost pixel occupying pixel } j,\\
l^r_{i,t} &= \text{the largest value of } j \text{ such that row } i\text{'s } t^{\text{th}} \text{ cluster can be}\\
&\qquad \text{placed in row } i \text{ with its leftmost pixel occupying pixel } j,\\
e^c_{j,t} &= \text{the smallest value of } i \text{ such that column } j\text{'s } t^{\text{th}} \text{ cluster can be}\\
&\qquad \text{placed in column } j \text{ with its topmost pixel occupying pixel } i,\\
l^c_{j,t} &= \text{the largest value of } i \text{ such that column } j\text{'s } t^{\text{th}} \text{ cluster can be}\\
&\qquad \text{placed in column } j \text{ with its topmost pixel occupying pixel } i.
\end{aligned}
$$

(The letters "e" and "l" stand for "earliest" and "latest".) In our example puzzle, row 7's first cluster must be placed so that its leftmost pixel occupies pixel 1, 2, or 3; row 7's second cluster must be placed so that its leftmost pixel occupies pixel 4, 5, or 6. In other words, $e^r_{7,1} = 1$, $l^r_{7,1} = 3$, $e^r_{7,2} = 4$, and $l^r_{7,2} = 6$.

Variables

Our formulation uses three sets of variables. One set specifies which pixels are painted black. For each $1 \le i \le m$ and $1 \le j \le n$, let

$$z_{i,j} = \begin{cases} 1 & \text{if row } i\text{'s } j^{\text{th}} \text{ pixel is painted black,} \\ 0 & \text{if row } i\text{'s } j^{\text{th}} \text{ pixel is painted white.} \end{cases}$$

The other two sets of variables are concerned with the placements of the row and column clusters. For each $1 \le i \le m$, $1 \le t \le k_i^r$, and $e_{i,t}^r \le j \le l_{i,t}^r$, let

$$y_{i,t,j} = \begin{cases} 1 & \text{if row } i\text{'s } t^{\text{th}} \text{ cluster is placed in row } i \\ & \quad \text{with its leftmost pixel occupying pixel } j, \\ 0 & \text{if not.} \end{cases}$$

For each $1 \le j \le n$, $1 \le t \le k_j^c$, and $e_{j,t}^c \le i \le l_{j,t}^c$, let

$$x_{j,t,i} = \begin{cases} 1 & \text{if column } j\text{'s } t^{\text{th}} \text{ cluster is placed in column } j \\ & \quad \text{with its topmost pixel occupying pixel } i, \\ 0 & \text{if not.} \end{cases}$$

Cluster Constraints

To ensure that row i's t^{th} cluster appears in row i exactly once, we impose

$$\sum_{j=e_{i,t}^r}^{l_{i,t}^r} y_{i,t,j} = 1.$$

The sum is over all pixels j such that row i's t^{th} cluster can be placed in row i with its leftmost pixel occupying pixel j. For the two clusters of row 7 of our example puzzle, we impose

$$y_{7,1,1} + y_{7,1,2} + y_{7,1,3} = 1$$

and

$$y_{7,2,4} + y_{7,2,5} + y_{7,2,6} = 1.$$

To guarantee that row i's $(t+1)^{\text{th}}$ cluster is placed to the right of its t^{th} cluster, we impose

$$y_{i,t,j} \le \sum_{j'=j+s_{i,t}^r+1}^{l_{i,t+1}^r} y_{i,t+1,j'} \quad \text{for each } e_{i,t}^r + 1 \le j \le l_{i,t}^r.$$

For row 7 of our example puzzle, we impose

$$y_{7,1,2} \leq y_{7,2,5} + y_{7,2,6}$$

and

$$y_{7,1,3} \leq y_{7,2,6}.$$

Similar constraints (involving the $x_{j,t,i}$'s) are needed for the column clusters.

Double Coverage Constraints

To guarantee that each black pixel is covered by both a row cluster and a column cluster, we impose, for each $1 \leq i \leq m$ and $1 \leq j \leq n$, the following pair of inequalities:

$$z_{i,j} \;\leq\; \sum_{t=1}^{k_i^r} \sum_{j'=\min\{e_{i,t}^r, j-s_{i,t}^r+1\}}^{\max\{l_{i,t}^r, j\}} y_{i,t,j'},$$

$$z_{i,j} \;\leq\; \sum_{t=1}^{k_j^c} \sum_{i'=\min\{e_{j,t}^c, i-s_{j,t}^c+1\}}^{\max\{l_{j,t}^c, i\}} x_{j,t,i'}.$$

The first inequality states that if row i's j^{th} pixel is painted black, then at least one of row i's clusters must be placed in such a way that it covers row i's j^{th} pixel. (The upper and lower limits of the second summation make sure that j' satisfies the inequalities $e_{i,t}^r \leq j' \leq l_{i,t}^r$ and $j - s_{i,t}^r + 1 \leq j' \leq j$. The first two hold if and only if row i's t^{th} cluster can be placed in row i with its leftmost pixel occupying pixel j'. The last two hold if and only if row i's j^{th} pixel is covered when row i's t^{th} cluster is placed in row i with its leftmost pixel occupying pixel j'.) The other one makes sure that if row i's j^{th} pixel is painted black, then at least one of column j's clusters covers it. In our example problem, for row 7's fifth pixel, we impose

$$z_{7,5} \leq y_{7,2,4} + y_{7,2,5}$$

and

$$z_{7,5} \leq x_{5,3,7} + x_{5,4,7}.$$

Finally, we include constraints that prevent white pixels from being covered by the row clusters or column clusters. For each $1 \leq i \leq m$, each $1 \leq j \leq n$, each $1 \leq t \leq k_i^r$, and each j' that satisfies the inequalities $j - s_{i,t}^r + 1 \leq j' \leq j$ and $e_{i,t}^r \leq j' \leq l_{i,t}^r$, we impose

$$z_{i,j} \geq y_{i,t,j'}.$$

And for each $1 \leq i \leq m$, $1 \leq j \leq n$, $1 \leq t \leq k_j^c$, and each i' that satisfies the inequalities $e_{j,t}^c \leq i' \leq l_{j,t}^c$ and $i - s_{j,t}^c + 1 \leq i' \leq i$, we impose

$$z_{i,j} \geq x_{j,t,i'}.$$

In our example problem, for row 7's fifth pixel, we impose

$$\begin{aligned} z_{7,5} &\geq y_{7,2,4}, \\ z_{7,5} &\geq y_{7,2,5}, \\ z_{7,5} &\geq x_{5,3,7}, \\ z_{7,5} &\geq x_{5,4,7}. \end{aligned}$$

Objective Function

There is no need for an objective function here.

Problems

Interested readers may enjoy trying to solve the following problems:

1. Try to improve the IP formulation. To obtain our code for constructing Cplex ".lp" files for the IP formulation, point your browser to

   `www.oberlin.edu/~math/faculty/people/bosch.html`

   and scroll down to the *Optima Mindsharpener* section.

2. Is there a better method for solving paint-by-numbers puzzles?

3. Solve the paint-by-numbers puzzle displayed in Figure 5. This puzzle was designed by Won Yoon Jo. Its solution is quite lovely, and it is perhaps the most difficult 20×20 puzzle on the Paint-by-Numbers Web Page (much more difficult than many of the 30×30 and 40×40 puzzles). To reach the Paint-by-Numbers Web Page, point your browser to

   `www02.so-net.ne.jp/~kajitani/cgi-bin/pbn.cgi/index.html`

	3	1																	
1	1	4	1	1		5						3							
1	2	2	3	4	1	1				6		1	3	2	2	3	3	2	
1	1	1	2	6	11	6		7	7	1	9	1	1	1	1	2	3	3	2
2	1	1	4	1	1	2	14	2	2	1	2	1	3	3	5	2	2	2	6

			7	1
		1	1	2
		2	1	2
		1	2	2
		4	2	3
		3	1	4
		3	1	3
		2	1	4
			2	9
		2	1	5
			2	7
				14
			8	2
		6	2	2
	2	8	1	3
	1	5	5	2
1	3	2	4	1
3	1	2	4	1
1	1	3	1	3
	2	1	1	2

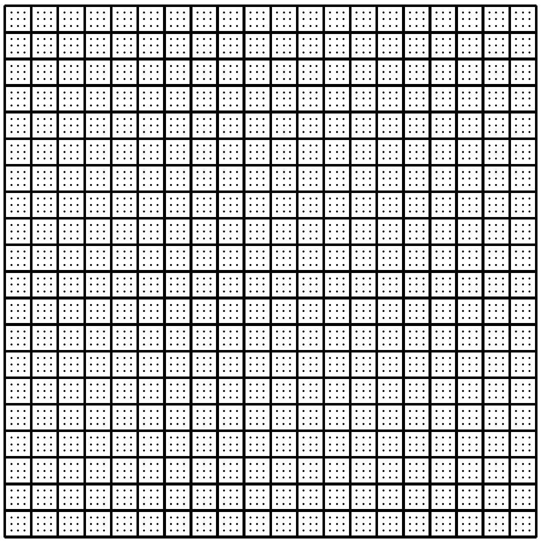

Figure 5