Fast generation of domino portraits

Hadrien Cambazard, John Horan, Eoin O’Mahony, Barry O’Sullivan

Cork Constraint Computation Center
SFI grant number 05/IN/1886
Outline

• The domino portrait problem
• The Integer linear programming approach of Robert Bosch
• A two-step approach
  – flow-based formulation of the problem
• Applications
Domino portraits

$$k$$ sets of double nine dominos

[Knowlton, Representation of design 1983]
[Knuth, The Stanford GraphBase, 1993]
[E. Berlekamp and T. Rogers, The mathemagician and pied puzzler, 1999]
[Bosh, constructing domino portraits, 2004]
$k = s^2$

55 x $k$ dominoes

110 x $k$ cells
Domino $k$ is denoted $d_k = [p_k^1, p_k^2]$

The grayscale value of cell $(i,j)$ is denoted $g_{i,j}$

Domino $d_k$ is placed on cells $(i_k^1, j_k^1) (i_k^2, j_k^2)$

Pb: $\text{Minimize} \sum_{d_k \in D} (p_k^1 - g_{i_k^1,j_k^1})^2 + (p_k^2 - g_{i_k^2,j_k^2})^2$

Use each domino exactly once
The integer linear programming model

[R. Bosch. Constructing domino portraits. Tribute to a Mathemagician, 2004]

• The model:
  - Boolean variables to specify if a given domino is placed with a given orientation with its reference square in a given cell of the grid.
  - Each domino has to be used once
  - Each cell is covered by a unique domino

• Very large models
  - $k=49$ gives 1,063,300 variables and around 2 hours of computation
A two-step approach

1. Pattern generation:
   cover the grid with empty dominoes (rectangles)

2. Assignment:

   grid of gray values

   pattern

   $(6 - 4)^2 + (3 - 1)^2 = 8$
A two-step approach

• Pattern: an arrangement of rectangles that covers the picture.

• Once the pattern is known the remaining problem is polynomial.

• Claim: any random pattern provide a good upperbound
Generate a random pattern

• Randomly assign rectangles vertically or horizontally with simple propagation

• Contradiction detection: a connected region of with an odd number of cells

• Partial restart by wiping out part of the grid
Optimal assignment

- The assignment problem is represented as a bipartite graph.
- The cost of assigning a domino to a rectangle is given by its best orientation.
- The **Hungarian** algorithm computes a minimum weight bipartite matching.
Optimal assignment

- Solving the assignment is polynomial
- Hungarian algorithm works in $O(n^3)$ [Kuhn 55]
- $k=100$ gives 5500 dominos. The Hungarian does not scale!
  
  ~10 minutes of computation
From $n^3$ to constant time

• Take advantage of symmetries:
  - Each domino is repeated $k$ times
  - There are only 55 kinds of rectangles defining areas

• An area is a set of rectangles of same cost
From $n^3$ to constant time

- Take advantage of symmetries:
  - Each domino is repeated $k$ times
  - There are only 55 kinds of rectangles defining areas

- An area is a set of rectangles of same cost
From $n^3$ to constant time

• Take advantage of **symmetries**:
  - Each domino is repeated **$k$** times
  - There are only **55 kinds** of rectangles defining **areas**

• **An area** is a set of rectangles of same cost
The min cost flow formulation
Optimal assignment

- There exists flow algorithm whose complexity does not depend on the capacities nor flow amount:
  - The Enhanced Capacity Scaling algorithm $O((m \log(n))(m + n \log(n)))$
  - The Successive Shortest Path is enough for our needs $O(n^2mU)$ where $U$ is the maximum capacity
Optimal assignment: min cost flow vs hungarian

- The flow formulation provides a constant time answer to the assignment problem!
First results

- Upper bounds obtained with a single random pattern

<table>
<thead>
<tr>
<th>k</th>
<th>ILP Opt Cost</th>
<th>Time (s)</th>
<th>Two steps (100 runs) Avg Cost</th>
<th>Min Cost</th>
<th>Avg Time (s)</th>
<th>Gap Avg Gap (%)</th>
<th>Min Gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1192</td>
<td>1.04</td>
<td>1260</td>
<td>1222</td>
<td>0.02</td>
<td>5.96</td>
<td>2.52</td>
</tr>
<tr>
<td>4</td>
<td>4844</td>
<td>13.8</td>
<td>5228</td>
<td>5139</td>
<td>0.04</td>
<td>7.99</td>
<td>6.09</td>
</tr>
<tr>
<td>9</td>
<td>11255</td>
<td>65.9</td>
<td>12183</td>
<td>12013</td>
<td>0.07</td>
<td>8.26</td>
<td>6.73</td>
</tr>
<tr>
<td>25</td>
<td>33673</td>
<td>325.62</td>
<td>36265</td>
<td>35998</td>
<td>0.12</td>
<td>7.71</td>
<td>6.90</td>
</tr>
<tr>
<td>49</td>
<td>69585</td>
<td>7030.29</td>
<td>74075</td>
<td>73639</td>
<td>0.13</td>
<td>6.45</td>
<td>5.83</td>
</tr>
<tr>
<td>121</td>
<td>171961</td>
<td>9797.55</td>
<td>181768</td>
<td>180991</td>
<td>0.16</td>
<td>5.72</td>
<td>5.25</td>
</tr>
<tr>
<td>225</td>
<td>376176</td>
<td>44895.86</td>
<td>386870</td>
<td>386326</td>
<td>0.17</td>
<td>2.84</td>
<td>2.69</td>
</tr>
</tbody>
</table>

- Very good upperbounds but a visually detectable gap to the optimal solution
Gap between ILP and random pattern + flow

Optimal solution

Upper bound (5.8 %)
Searching among patterns

• We now know how to solve the problem very efficiently (constant time) once the pattern is known.

  How do we find the right pattern?

• Main Idea: the pattern only matters where the grey values are varying.

• A change of the pattern that would not affect the size of the areas on the flow graph has no effect over the optimal assignment.

• Our approach consists in slightly perturbing the pattern in a local search manner to affect the capacities of the areas and improve the flow.
A LNS algorithm

1. Identify the regions of the grid where the cost varies
2. Randomly select a point from those areas and remove M dominos around it
3. Enumerate all possible patterns (LNS step) that can fill the hole:
   - Update incrementally the flow *(sensitivity analysis* of the flow)
   - Store new solution if improvement
4. Return to 2 until a stopping criterion is met
Selecting the points of interest

Result of the FAST (Features from Accelerated Segment Test) algorithm on the “Girl with a Pearl Earring”
Final results

- High quality portraits (gap around 2%)
- Orders of magnitude of speed up (seconds vs hours)

<table>
<thead>
<tr>
<th>k</th>
<th>ILP Opt Cost</th>
<th>Time (s)</th>
<th>LNS patterns Cost</th>
<th>Time (s)</th>
<th>Gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1192</td>
<td>1.04</td>
<td>1207</td>
<td>8.32</td>
<td>1.26</td>
</tr>
<tr>
<td>4</td>
<td>4844</td>
<td>13.8</td>
<td>4903</td>
<td>14</td>
<td>1.22</td>
</tr>
<tr>
<td>9</td>
<td>11255</td>
<td>65.9</td>
<td>11512</td>
<td>14.54</td>
<td>2.28</td>
</tr>
<tr>
<td>25</td>
<td>33673</td>
<td>325.62</td>
<td>34498</td>
<td>15.72</td>
<td>2.45</td>
</tr>
<tr>
<td>49</td>
<td>69585</td>
<td>7030.29</td>
<td>70977</td>
<td>17.66</td>
<td>2</td>
</tr>
<tr>
<td>121</td>
<td>171961</td>
<td>9797.55</td>
<td>175669</td>
<td>27.34</td>
<td>2.16</td>
</tr>
<tr>
<td>225</td>
<td>376176</td>
<td>44895.86</td>
<td>380408</td>
<td>32.71</td>
<td>1.13</td>
</tr>
</tbody>
</table>
Gap between ILP and LNS

Optimal solution (k = 49) ~7030s

LNS solution (k = 49) ~18s
Applications

- Children love it!

Science discovery event 2007 in Cork
Applications

- People finally know what you are doing at work
Applications

• Teaching OR with fun:
  • graph algorithms (Hungarian, Min cost flow and sensitivity analysis)
  • search techniques (depth first search with simple propagation, LNS)
  • algorithm from computer vision (FAST)
Conclusion

- An efficient and scalable approach based on a reformulation of the problem as a min cost flow problem
- Orders of magnitude of improvements compared to the integer linear approach
- Massive success with kids and great teaching tool
- [http://4c.ucc.ie/~hcambaza/](http://4c.ucc.ie/~hcambaza/)
Questions?
Applications

- Packing with positioning cost?