Assignment 4

Constraint programming

2013

Mahboobeh Abdal Mahmood Abadi & Mehran Nasseri

KTH

5/22/2013

Table of Contents

[1. Branching for Square Packing 3](#_Toc356771972)

[1.1. Experiment 3](#_Toc356771973)

[2. Maximum density still life 4](#_Toc356771974)

# Branching for Square Packing

For interval branching we decided to split the coordinate interval into two subintervals. In this way, only the first alternative forces the obligatory part to a certain percentage p \*w where ‘p’ is a double number between 0.0 and 1.0 and ‘w’ is width.

For every square ‘i’ we need to ensure that if coordinate interval is large enough so that obligatory parts are generated and we can split. This can be checked by following equation:

(x[i].min() + w[i]) - x[i].max()< p\*w[i]

If the above condition is true, so branching is possible on square ‘i’, and we need to calculate the split point such that the interval is at least p\*w[start] . The split point can be calculated as follow:

Start :=i

The split point is the smallest value ‘v’ in domain x-coordinates in which:

x[start].min() + w[start] - v >= p\*w[start]

Now we can split x-coordinate into two subinterval [x[start].min(), v] and [v+1, x[start].max()]

# Experiment

The new branching strategy significantly improved the performance as it reduced both the number of explored nodes and failures dramatically. Moreover, the performance improvement depends on the parameter ‘p’. The result of experiment (table 1) illustrates that, the best result, especially for the larger problems, can be obtained when parameter ‘p’ is set to 0.7 or 0.8.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No-overlap propagator with interval**  **P=0.7** | | | **No-overlap propagator with interval**  **P=0.8** | | | **No-overlap propagator without interval** | | |
| N | Time | Nodes | Failures | Time | Nodes | Failures | Time | Nodes | Failures |
| 14 | 90ms | 64 | 21 | 115ms | 121 | 50 | 192ms | 787 | 382 |
| 15 | 123ms | 25 | 2 | 125ms | 24 | 1 | 104ms | 99 | 38 |
| 16 | 123ms | 479 | 232 | 134ms | 792 | 387 | 307ms | 5311 | 2645 |
| 17 | 221ms | 1492 | 735 | 271ms | 184 | 79 | 33s | 558155 | 279065 |
| 18 | 1s | 20958 | 10318 | 3s | 48444 | 24209 | 3m24s | 3779396 | 1889686 |
| 19 | 172ms | 738 | 361 | 222ms | 1828 | 906 | 2s | 43695 | 21837 |
| 20 | 851ms | 7913 | 3945 | 281ms | 2638 | 1308 | 1m | 935128 | 467553 |
| 21 | 2s | 29707 | 14842 | 4s | 44533 | 22257 | 3m40s | 3139868 | 1569923 |
| 22 | 16s | 162413 | 81197 | 20s | 217277 | 108629 | 1h4m22s | 47531512 | 23765747 |
| 23 | 1m10s | 835422 | 417700 | 1m6s | 705955 | 352965 | - | - | - |

Table 1: the performance comparison

# Maximum density still life

According to the figure 1 the maximum density value that we have found for the size 8\*8 is 36.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Figure 1: maximum density pattern for size 8

According to the figure 2 the maximum density value that we have found for the size 9\*9 is 43.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Figure 2: maximum density pattern for size 9