Appendix C
Implementation Details of Problem Instances

This appendix gives implementation details for the optimization problems used in the experiments of Chapter 4. For further information, including the complete datasets used in the bin-packing, Bayes net structure-finding, and cartogram domains, please access the following web page:

$$\texttt{http://www.cs.cmu.edu/\~{}AUTON/stage/} \tag{C.1}$$

C.1 Bin-packing

The bin-packing problem was introduced in Sections 3.3.2 and 4.2. We are given a *bin capacity* C and a list $L = (a_1, a_2, ...a_n)$ of *items*, each having a *size* $s(a_i) > 0$. The goal is to pack the items into as few bins as possible, i.e., partition them into a minimum number m of subsets $B_1, B_2, ..., B_m$ such that for each B_j, $\sum_{a_i \in B_j} s(a_i) \leq C$.

In the STAGE experiments, value function approximation was done with respect to two state features: the objective function itself, and the variance in bin fullness levels. In terms of the above notation, given a packing $x = \{B_1, B_2, \ldots, B_{M_x}\}$ (with all bins B_j assumed non-empty), we have

$$\begin{aligned}
\mathrm{Obj}(x) &\stackrel{\text{def}}{=} M_x \\
\mathrm{Var}(x) &\stackrel{\text{def}}{=} \left(\frac{1}{M_x} \sum_{j=1}^{M_x} \mathrm{fullness}(B_j)^2 \right) - \left(\frac{1}{M_x \cdot C} \sum_{i=1}^{n} s(a_i) \right)^2
\end{aligned}$$

where

$$\mathrm{fullness}(B_j) \stackrel{\text{def}}{=} \frac{1}{C} \sum_{a_i \in B_j} s(a_i) .$$

The illustrative example of Section 3.3.2 consisted of the following 30 items, to be packed into bins of capacity 100:

(27, 23, 23, 23, 27, 26, 26, 51, 26, 23,
51, 23, 51, 23, 23, 51, 23, 23, 27, 23,
51, 27, 51, 26, 27, 23, 26, 27, 26, 23)

These particular item sizes were motivated by Figure 2.5 of [Coffman *et al.* 96], which depicts a template of a worst-case example for the "First Fit Decreasing" offline bin-packing heuristic. The optimal packing fills 9 bins exactly to capacity.

The twenty instances of the u250 class were contributed to the Operations Research Library by Falkenauer [96]. They may be downloaded from the web page referenced at the start of this section, or directly from the OR-Library web site at

http://www.ms.ic.ac.uk/info.html .

C.2 VLSI Channel Routing

My implementation of channel routing follows that of the SACR system [Wong *et al.* 88]. This system allows a restricted form of doglegging, whereby a net may be split horizontally only at columns containing a pin belonging to that net.

Most of the experiments in this dissertation were conducted on the instance YK4. That instance is specified by the following pin columns:

Upper: 17 9 23 33 0 17 34 33 32 31 32 20 9 10 21 34 0 31 22 10 0 22 1 3 16 0 0 0 9 19 7 0 16 14 7 51 43 57 67 0 51 68 67 66 65 66 54 43 44 55 68 0 65 56 44 0 56 35 37 50 0 0 0 43 53 41 0 50 48 41 85 77 91 101 0 85 102 101 100 99 100 88 77 78 89 102 0 99 90 78 0 90 69 71 84 0 0 0 77 87 75 0 84 82 75 119 111 125 135 0 119 136 135 134 133 134 122 111 112 123 136 0 133 124 112 0 124 103 105 118 0 0 0 111 121 109 0 118 116 109

Lower: 0 0 0 24 10 0 4 2 21 2 4 23 1 4 24 1 4 2 0 1 4 0 3 19 2 3 20 2 3 14 3 1 9 24 0 0 0 0 58 44 0 38 36 55 36 38 57 35 38 58 35 38 36 0 35 38 0 37 53 36 37 54 36 37 48 37 35 43 58 0 0 0 0 92 78 0 72 70 89 70 72 91 69 72 92 69 72 70 0 69 72 0 71 87 70 71 88 70 71 82 71 69 77 92 0 0 0 0 126 112 0 106 104 123 104 106 125 103 106 126 103 106 104 0 103 106 0 105 121 104 105 122 104 105 116 105 103 111 126 0

These pin columns correspond to Example 1 (Figure 25) of [Yoshimura and Kuh 82], but multiplied four times and placed side by side. By "cloning" the problem in this manner, we maintain the known global optimum (in this case, 10 tracks when restricted doglegging is allowed), but make the problem much more difficult for local search.

In Section 6.2, I applied STAGE to eight additional channel routing instances, HYC1 through HYC8. The pin columns for these problems may be found in [Chao and Harper 96] and are also available on the STAGE web page.

C.3 Bayes Network Learning

For the experiments with the Bayes-net structure-finding domain (Sections 4.4 and 6.1.5), three datasets were used: MPG, ADULT2, and SYNTH125K.

- The MPG dataset contains information on the horsepower, weight, gas mileage, and other such data (10 total attributes) for 392 automobiles. It is derived from the "Auto-Mpg" dataset available from the UCI Machine Learning repository [Merz and Murphy 98], but modified by coarsely discretizing all continuous variables.

- The ADULT2 dataset was also obtained from the UCI repository. It consists of census data related to the job, wealth, nationality, etc. (15 total attributes) on 30,162 individuals.

- The SYNTH125K dataset was generated synthetically from the probability distribution given by the Bayes net in Figure 4.7 (p. 86), designed by Moore and Lee [98].

All three datasets are available for downloading from the STAGE web page.

C.4 Radiotherapy Treatment Planning

The radiotherapy treatment planning domain of Section 4.5 is too complex to explain in full detail here. Please refer to the web page for more information. Here, I describe the domain in enough detail to illustrate its complexity—in particular, to show why we must resort to using local search rather than, say, linear programming to solve it. The form of my objective function was based on discussions with domain experts; however, I did not have access to a medically accurate implementation of the dose calculations and penalty functions. Thus, I claim only that the optimization problem solved here retains most of the overall structure and geometry of tradeoffs found in the true medical domain.

I formulated the problem as follows. The treatment area is discretized into an 80×80 rectangular grid. The radiation dosage $\mathrm{dose}_p(x)$ at each pixel p can then be calculated from the current plan x according to a known forward model. Pixels within the area of the tumor t have a target dose targ_t and incur a penalty based on the ratio $r_p = \frac{\mathrm{dose}_p(x)}{\mathrm{targ}_t}$:

$$\mathrm{Penalty}(p) = \begin{cases} \exp(1/\max(r_p, 0.1)) & \text{if } r_p < 1 \text{ (underdose)} \\ r_p - 1 & \text{if } r_p \geq 1 \text{ (overdose)} \end{cases}$$

Pixels within a sensitive structure s have a maximum acceptable dose accep_s and incur a penalty based on the ratio $r_p = \frac{\text{dose}_p(x)}{\text{accep}_s}$:

$$\text{Penalty}(p) = \begin{cases} r_p & \text{if } r_p \leq 1 \text{ (safe dose)} \\ \exp(r_p) & \text{if } r_p > 1 \text{ (overdose)} \end{cases}$$

These two penalty functions are plotted in Figure C.1. Finally, the overall objective function is calculated as a weighted sum of all the per-pixel penalties. The weights are fixed and reflect the relative importance of the various structures being targeted or protected.

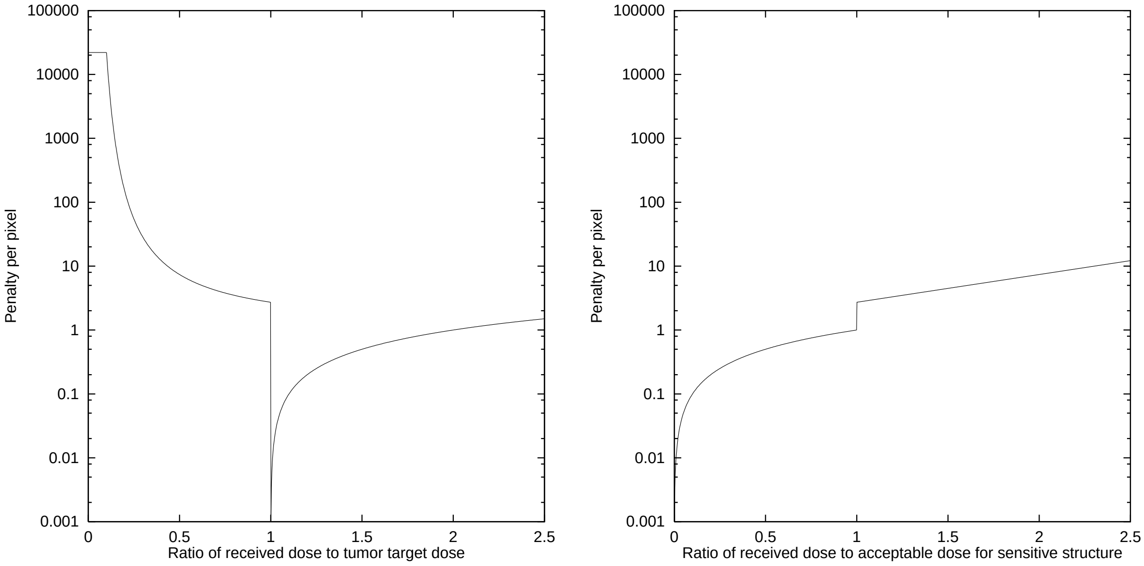

FIGURE C.1. The per-pixel penalty for pixels within a tumor (left) or within a sensitive structure (right). Note the logarithmic scale on the y-axis.

This problem formulation is nearly suitable for a linear programming solution: the dose incurred at each pixel is a linear combination of the beam intensities, and the objective function is a linearly weighted sum of penalty terms. If the penalty functions of Figure C.1 had been defined to be piecewise linear and convex, and the beam intensities were allowed to vary continuously over $[0, 1]$, then the optimal treatment plan x^* would be obtainable by linear programming. However, with 0/1 discrete beam intensities and non-convex penalty functions, as assumed in the experiments of this thesis, we must resort to heuristic search.

C.5 Cartogram Design

The cartogram design problem was introduced in Section 4.6. This thesis investigated a single instance, US49. Instance US49 is defined by the 49 polygons of the continental U.S. map (48 states plus the District of Columbia) and their respective target areas, which are based on the 1990–2000 electoral vote for U.S. President. This dataset is available from the STAGE web page.

Local search moves in this domain consist of choosing one of the 162 points and perturbing it slightly. These points are chosen randomly, but with a bias toward those points that contribute most to the current map error function. To be precise: first a state is chosen with probability proportional to its contribution to $\mathrm{Obj}(x)$, and then one point on that state is chosen uniformly at random. The perturbation is then generated by adding uniformly random numbers in $[-1.5, 1.5]$ to each coordinate.

The objective function is defined as the sum of four penalty terms:

$$\mathrm{Obj}(x) = \Delta\mathrm{Area}(x) + \Delta\mathrm{Gape}(x) + \Delta\mathrm{Orient}(x) + \Delta\mathrm{Segfrac}(x)$$

The penalty terms are defined as follows:

$$\begin{aligned}
\Delta\mathrm{Area}(x) &= \frac{10}{\#\mathrm{States}} \sum_{s \in \mathrm{States}} \big(\max(\frac{\mathrm{Area}_x(s)}{\mathrm{Area}_{\mathbf{targ}}(s)}, \frac{\mathrm{Area}_{\mathbf{targ}}(s)}{\max(\mathrm{Area}_x(s), 0.001)}) - 1\big)^2 \\
\Delta\mathrm{Gape}(x) &= \frac{1}{\#\mathrm{Bends}} \sum_{\angle ABC \in \mathrm{Bends}} \big(\mathrm{measure}_x(\angle ABC) - \mathrm{measure}_{\mathbf{orig}}(\angle ABC)\big)^2 \\
\Delta\mathrm{Orient}(x) &= \frac{1}{\#\mathrm{Bends}} \sum_{\angle ABC \in \mathrm{Bends}} \big(\mathrm{measure}_x(\angle AB__) - \mathrm{measure}_{\mathbf{orig}}(\angle AB__)\big)^2 \\
\Delta\mathrm{Segfrac}(x) &= \frac{10}{\#\mathrm{Bends}} \sum_{\angle ABC \in \mathrm{Bends}} \big(\frac{\mathrm{length}_x(\overline{AB})}{\mathrm{perim}_x(\mathrm{State}(\angle ABC))} - \frac{\mathrm{length}_{\mathbf{orig}}(\overline{AB})}{\mathrm{perim}_{\mathbf{orig}}(\mathrm{State}(\angle ABC))}\big)^2
\end{aligned}$$

The "Bends" above index each angle of each polygon in the map. The notation "$\angle AB\_\_$" refers to the angle made between line $\overleftrightarrow{AB}$ and the x-axis (a fixed horizontal line). Finally, the constants 1 and 10 in these penalty terms were chosen by trial and error to create aesthetically appealing cartograms.

C.6 Boolean Satisfiability

The difficult "32-bit parity function learning" instances tested in Section 4.7 are described in [Crawford 93]. The complete formulas are available for downloading from the STAGE web page, and also from the DIMACS satisfiability archive:

`ftp://dimacs.rutgers.edu/pub/challenge/sat/benchmarks/cnf/` .