Appendix B
Simulated Annealing

B.1 Annealing Schedules

Simulated annealing (SA) is described by the template in Table B.1. The main implementation challenge is to choose an effective annealing schedule for the temperature parameter. The temperature t_i controls the probability of accepting a step to a worse neighbor: when $t_i = +\infty$, SA accepts any step, acting as a random walk; whereas when $t_i = 0$, SA rejects all worsening steps, acting as stochastic hillclimbing with equi-cost moves.

Simulated-Annealing(X, S, N, Obj, TotEvals, *Schedule*):

Given:

- a state space X
- starting states $S \subset X$
- a neighborhood structure $N : X \to 2^X$
- an objective function, $\mathrm{Obj} : X \to \Re$, to be minimized
- TotEvals, the number of state evaluations allotted for this run
- an annealing *Schedule* which determines the temperature on each iteration

1. Let $x_0 \in S$ be a random starting state for search;
   let $t_0 :=$ the initial temperature of *Schedule*
2. For $i := 0$ to $\textsc{TotEvals} - 1$, do:
   (a) Choose $x' :=$ a random element from $N(x_i)$
   (b) $x_{i+1} := \begin{cases} x' & \text{if } \textsc{rand}[0,1) < e^{[\mathrm{Obj}(x_i) - \mathrm{Obj}(x')]/t_i} \\ x_i & \text{otherwise} \end{cases}$
   (c) Update temperature t_{i+1} according to *Schedule*
3. Return the best state found.

Table B.1. A template for the simulated annealing algorithm

The temperature annealing schedule has been the subject of extensive theoretical and experimental analysis in the simulated annealing literature [Boese 96]. The possibilities include the following:

- Logarithmic: $t_i = \gamma / \log(i+2)$. For sufficiently high γ, a schedule of this form guarantees that SA will converge with probability one to the globally optimal solution [Mitra *et al.* 86, Hajek 88]. Unfortunately, this guarantee only applies in the limit as TotEvals $\to \infty$. Logarithmic schedules are generally too slow for practical use.

- Geometric: for a fixed *initial temperature* t_0, *cooling rate* $\alpha \in (0,1)$ and *round length* $L \geq 1$, define $t_i = t_0 \cdot \alpha^{\lfloor i/L \rfloor}$. This is the original schedule proposed by Kirkpatrick [83], and is still widely used in practice [Johnson *et al.* 89, Johnson *et al.* 91, Press *et al.* 92]. However, the parameters t_0, L, and α must be tuned for each problem instance.

- Adaptive: a variety of adaptive schedules have been proposed, with both theoretical and practical motivations. [Boese 96] summarizes the most notable of these, including the schedules of [Aarts and van Laarhoven 85], [Huang *et al.* 86], and [Lam and Delosme 88]. According to [Ochotta 94],

  > Results for the Lam schedule are quite impressive. When compared with other general-purpose annealing schedules such as [Huang *et al.* 86] and even hand-crafted schedules like the one in [Sechen and Sangiovanni-Vincentelli 86], it often provides speed-ups of 50% while actually improving the quality of the final answers [Lam and Delosme 88]. Simulated annealing with the Lam schedule even compares favorably with heuristic combinatorial optimization methods tuned to specific problems like partitioning and the travelling salesman problem [Lam 88].

B.2 The "Modified Lam" Schedule

For the purposes of comparing against STAGE, I sought to implement simulated annealing with a cooling schedule which would both perform very well and would require little tuning from one problem instance to the next. After some experimentation, I settled on an adaptive schedule similar to that used in [Ochotta 94], which in turn was based on Swartz's modification of the Lam schedule [Swartz and Sechen 90]. Swartz observed that for each of a large number of annealing runs using the Lam schedule, the accept ratio—that is, the ratio of moves accepted to moves considered—followed an almost identical pattern with respect to the move counter i [Ochotta 94, p. 137]:

The accept rate starts at about 100%, decreases exponentially until it stabilizes (about 15% of the way through the run) at about 44%, and remains there until about 65% of the way through the run. It then continues its exponential decline until the end of the annealing process. Based on this observation, Swartz duplicated the effect of the Lam schedule by using a simple feedback loop to control the temperature and *force* the accept ratio to follow the curve in Figure [B.1]. In comparing the modified schedule to the original schedule, Swartz reported almost no difference in the quality of the final answers [Swartz and Sechen 90]. One additional benefit of the modified Lam schedule is that, in contrast to the original schedule, the total number of moves in the annealing run can be specified by the user.

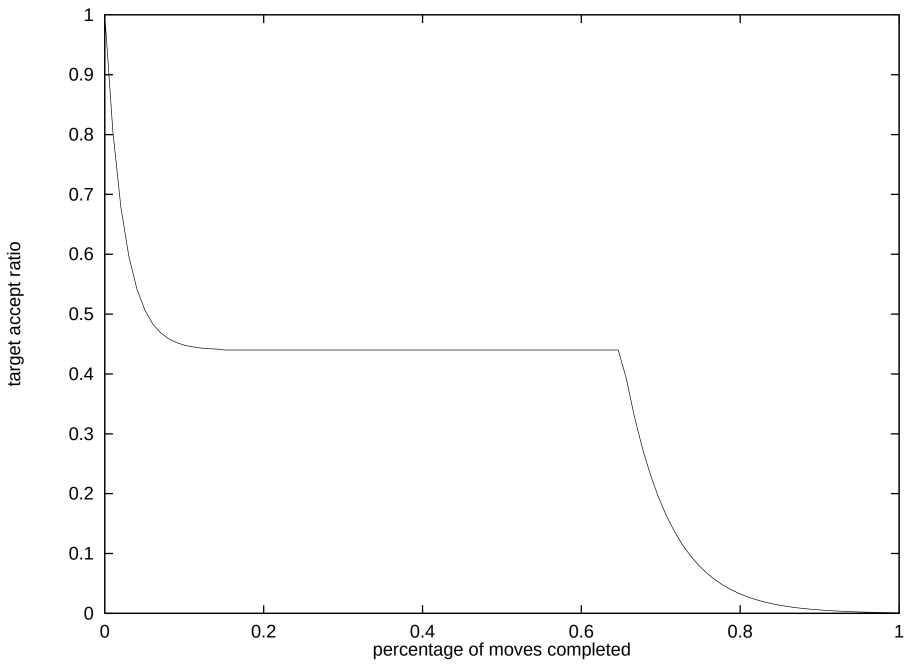

FIGURE B.1. Accept rate targets for the modified Lam schedule

All details of my implementation of Swartz's modified Lam schedule are given below in Table B.2. I do not dynamically readjust the neighborhood structure $N(x)$ over the course of search, as the original Lam schedule did, since such readjustments cannot be specified in a problem-independent manner.

Simulated-Annealing(X, S, N, Obj, TotEvals, **Modified-Lam-Schedule**):

1. Let $x_0 \in S$ be a random starting state for search;
   let AcceptRate:=0.5, $t_0 := 0.5$.

2. For i := 0 to TotEvals − 1, do:

   (a) Choose $x' :=$ a random element from $N(x_i)$

   (b) $x_{i+1} := \begin{cases} x' & \text{if } \textsc{rand}[0,1) < e^{[\mathrm{Obj}(x_i) - \mathrm{Obj}(x')]/t_i} \\ x_i & \text{otherwise} \end{cases}$

   (c) Update temperature t_{i+1} as follows:

      i. let $\textsc{AcceptRate} := \begin{cases} \frac{1}{500}(499 \cdot \textsc{AcceptRate} + 1) & \text{if } x' \text{ was accepted} \\ \frac{1}{500}(499 \cdot \textsc{AcceptRate}) & \text{if } x' \text{ was rejected} \end{cases}$

      ii. let $d := i/\textsc{TotEvals}$

      iii. let $\textsc{TargRate} := \begin{cases} 0.44 + 0.56 \cdot 560^{-d/0.15} & \text{if } 0 \leq d < 0.15 \\ 0.44 & \text{if } 0.15 \leq d < 0.65 \\ 0.44 \cdot 440^{-(d-0.65)/0.35} & \text{if } 0.65 \leq d < 1 \end{cases}$

      iv. let $t_{i+1} := \begin{cases} t_i \cdot 0.999 & \text{if } \textsc{AcceptRate} > \textsc{TargRate} \\ t_i/0.999 & \text{otherwise} \end{cases}$

3. Return the best state found.

Table B.2. Details of the "modified Lam" adaptive simulated annealing schedule, instantiating the template of Figure B.1. The TargRate function is plotted in Figure B.1.

B.3 Experiments

I certainly do not claim that the modified Lam schedule is perfectly optimized for every problem attempted in Chapter 4. Getting the best performance from simulated annealing on any given problem is an art that involves refining the temperature schedule, dynamically adjusting the search neighborhood, and tuning the cost function coefficients. However, empirically, the simulated annealing algorithm defined by Table B.2 does seem to perform very well on a wide variety of problems without requiring further tuning.

The following experiments illustrate the effectiveness of my implementation. I consider four of the optimization domains of Chapter 4: bin-packing (§4.2), VLSI channel routing (§4.3), Bayes net learning (§4.4), and cartogram design (§4.6). For each of these domains, I compare the performance of the modified Lam schedule with that of 12 different geometric cooling schedules, defined by $t_i = t_0 \cdot \alpha^i$ for all combinations of

$$\begin{aligned} \text{initial temperature} \;\; & t_0 \in \{10, 1, 0.1, 0.01\} \\ \text{cooling rate} \;\; & \alpha \in \{0.9999, 0.99999, 0.999999\}. \end{aligned}$$

The total length of the schedules is set to TotEvals, the same setting used in the comparative experiments of Chapter 4: 10^5 for bin-packing and Bayes net learning, $5 \cdot 10^5$ for channel routing, and 10^6 for cartogram design.

Results are shown in Figures B.2–B.5. Each figure plots thirteen boxes: the left-most box corresponds to the modified Lam schedule results (copied from Chapter 4), and the other 12 boxes correspond to the performance of the various geometric cooling schedules (averaged over 10 runs each). In all cases, the modified Lam schedule is nearly as effective as, if not more effective than, the best of the geometric schedules.

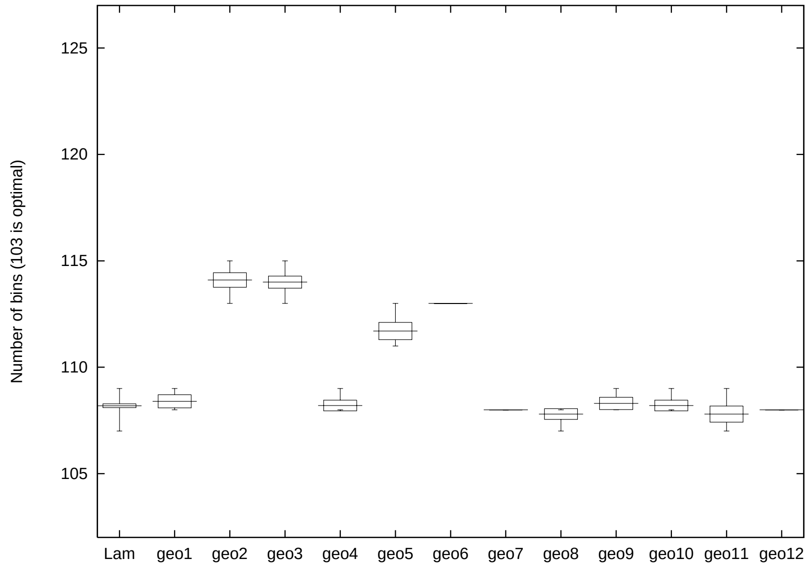

FIGURE B.2. Simulated annealing schedules for bin-packing instance u250_13: Modified Lam schedule (leftmost) versus 12 geometric cooling schedules

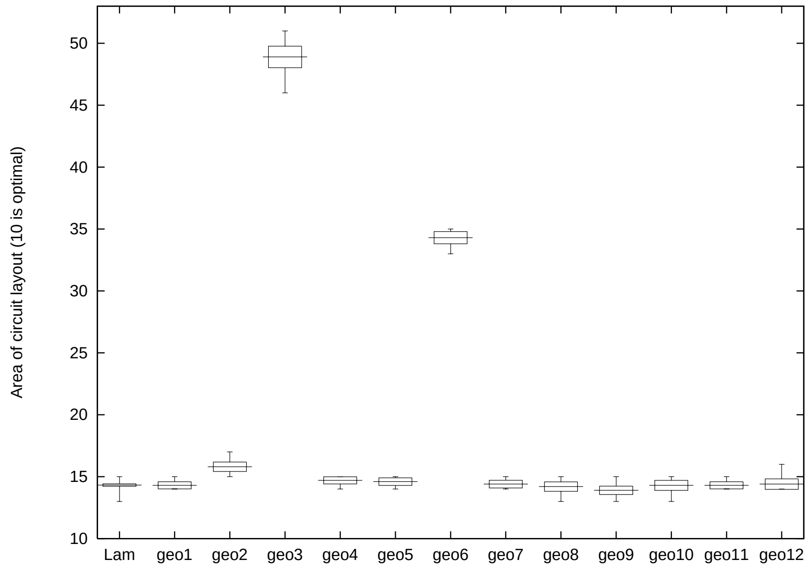

FIGURE B.3. Simulated annealing schedules for channel routing instance YK4: Modified Lam schedule (leftmost) versus 12 geometric cooling schedules

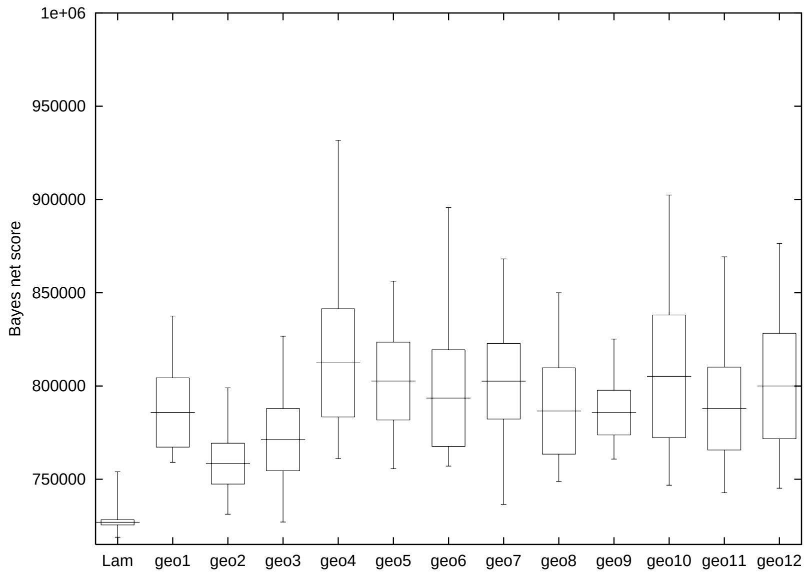

FIGURE B.4. Simulated annealing schedules for Bayes net structure-finding instance SYNTH125K: Modified Lam schedule (leftmost) versus 12 geometric cooling schedules

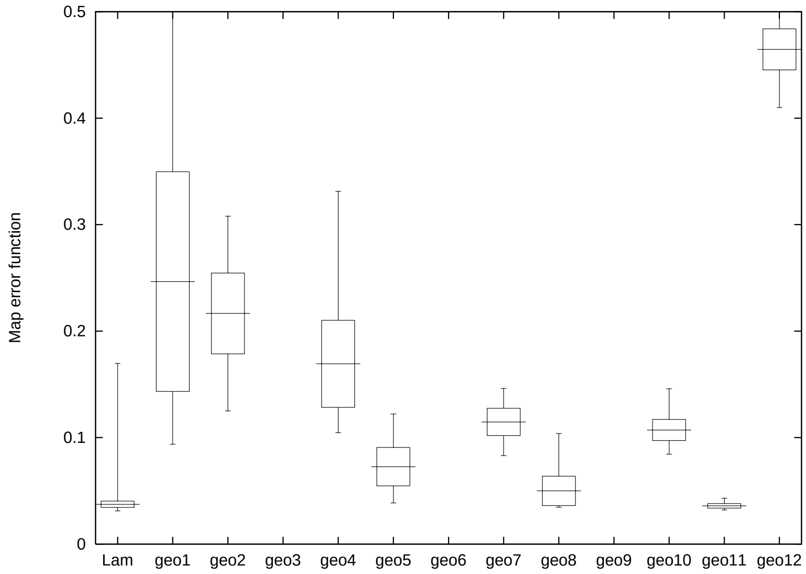

FIGURE B.5. Simulated annealing schedules for cartogram design instance US49: Modified Lam schedule (leftmost) versus 12 geometric cooling schedules