ESE535: Electronic Design Automation

Day 20: April 13, 2009
Placement II
(Simulated Annealing)

Today

- Placement
- Improving Quality
  - Cost functions
  - Avoiding local minima
- Technique:
  - Simulated Annealing

Simulated Annealing

- Physically motivated approach
- Physical world has similar problems
  - objects/atoms seeking minimum cost arrangement
  - at high temperature (energy) can move around
    - \textit{E.g.} it melts
  - at low temperature, no free energy to move
  - cool quickly \rightarrow freeze in defects (weak structure)
    - glass
  - cool slowly \rightarrow allow to find minimum cost
    - crystal

Key Benefit

- Avoid Local Minima
  - Allowed to take locally non-improving moves in order to avoid being stuck

Design Optimization

\textbf{Components:}

1. “Energy” (Cost) function to minimize
   - represent \textit{entire} state, drives system forward
2. Moves
   - local rearrangement/ transformation of solution
3. Cooling schedule
   - initial temperature
   - temperature steps (sequence)
   - time at each temperature
Basic Algorithm Sketch

• Pick an initial solution
• Set temperature \( T \) to initial value
• while \( T > T_{\text{min}} \)
  – for time at \( T \)
    • pick a move at random
    • compute \( \Delta \text{cost} \)
    • if less than zero, accept
    • else if (\( \text{RND} < e^{\Delta \text{cost} / T} \)), accept
  – update \( T \)

Details

• Initial Temperature
  \[ T_0 = \frac{\Delta \text{avg}}{\ln(P_{\text{accept}})} \]
  \[ e^{\Delta \text{cost} / T} \]
  \[ e^{\Delta \text{cost} / T_0} = e^{\Delta \text{cost} / \left( \frac{\Delta \text{avg}}{\ln(P_{\text{accept}})} \right)} \]
  • Average move \( \rightarrow e^{\ln(P_{\text{accept}})} \)
  • Assume increasing cost is negative \( \Delta \text{avg} \)
  • Accepted with Probability \( P_{\text{accept}} \)

Details

• Cooling schedule
  – fixed ratio: \( T = \lambda T \)
  \( \text{e.g. } \lambda = 0.85 \)
  – temperature dependent
  – function of both temperature and acceptance rate
  • example to come

• Time at each temperature
  – fixed number of moves?
  – fixed number of rejected moves?
  – fixed fraction of rejected moves?

VPR Cooling Schedule

• Moves at Temperature = \( cN^{4/3} \)
• Temperature Update
  \[ T_{\text{new}} = T_{\text{old}} \times \gamma \]
  – Idea: advance slowly in good \( \alpha \) range
  \( \square \alpha \) is measured acceptance rate

\( \begin{array}{cc}
\alpha > 0.96 & 0.5 \\
0.8 < \alpha \leq 0.96 & 0.9 \\
0.15 < \alpha \leq 0.8 & 0.95 \\
\alpha \leq 0.15 & 0.8 \\
\end{array} \)

Kluwer 1999

Cost Function

• Can be very general
  – Combine area, timing, energy, routability…
• Desirable characteristics:
  1. drive entire solution in right direction
     • reward each good move
  2. cheap to compute delta costs
     • e.g. FM
     • Ideally \( O(1) \)

Bad Cost Functions

• Not reward every move:
  – size < threshold ?
  – Anything using max
    • channel width
    • critical path delay
• Expensive update cost
  – rerun router on every move
  – rerun static timing analysis
    • E.g. recalculate critical path delay
**Example Cost Functions**

- **Total Wire Length**
  - Linear, quadratic...
  - $\sum \text{Bounding Box (semi-perimeter)}$
  - Surrogate for routed net length

- $\sum (\text{channel density})$
  - Dominate by largest density
  - Rewards improvement in non-maximum channel
  - But reward is larger for denser channels
  - Can be computed incrementally

**VPR Wire Costs**

- VPR Bounding Box
  \[
  \text{Cost} = \sum_{i=1}^{\text{Nets}} q(i) \times [(bb_x(i) + bb_y(i))]
  \]

- Original table:
  - Swartz, Betz, & Rose
  - FPGA 1998

**VPR Timing Costs**

- Criticality(e) = $1 - \frac{\text{Slack(e)}}{D_{\text{max}}}$
- $T\text{Cost(e)} = \text{Delay(e)} \times \text{Criticality(e)}^{\text{CriticalityExp}}$
- Keep all edge delays in a table
- Recompute Net Criticality at each Temperature

**VPR Balance Wire and Time Cost**

\[
\Delta \text{Cost} = \lambda \left( \frac{\Delta T\text{Cost}}{\text{OldTCost}} \right) + (1 - \lambda) \left( \frac{\Delta W\text{Cost}}{\text{OldWCost}} \right)
\]

- Marquardt, Betz, & Rose
- FPGA2000

**Stale Criticality**

- Criticality becomes stale during moves
  - Strategy of updating STA infrequently problematic for highly pipelined logic

**Initial Solution**

- Spectral Placement
- Random
- Constructive Placement
  - Fast placers start at lower temperature; assume constructive got global right.
Moves

- Swap two cells
  - Within some distance limit? (ex. to come)
- Swap regions
  - ...rows, columns, subtrees, cluster
- Rotate cell (when feasible)
- Flip (mirror) cell
- Permute cell inputs (equivalent inputs)

Legality Constraints

- Examples:
  - Limit on number of gates/cluster (position)
  - Limit on number of Inputs/cluster (region)
- Options:
  - Force all moves to be legal
    - Force initial placement to be legal
    - Illegal moves rejected
  - Allow illegal placement/moves
    - Set cost function to make undesirable
    - Make less desirable (more costly) over time

Basic Algorithm Sketch (review)

- Pick an initial solution
- Set temperature (T) to initial value
- While (T > T_min)
  - For time at T
    - Pick a move at random
    - Compute Δcost
    - If less than zero, accept
    - Else if (RND < e^-Δcost/T), accept
    - Update T

Variant: “Rejectionless”

- Order moves by cost
  - compare FM
- Pick random number first
- Use random to define range of move costs will currently accept
- Pick randomly within this range
- Idea: never pick a costly move which will be rejected

Simulated Annealing Theory

- If stay long enough at each cooling stage
  - will achieve tight error bound
- If cool long enough
  - will find optimum
- ...but is it any less work than exhaustive exploration?
  - Good to have a continuum....

Practice

- Good results
  - ultimately, what most commercial tools use...what vpr uses...
- Slow convergence
- Tricky to pick schedules to accelerate convergence
  - Too slow → runs too long
  - Too fast → freezes prematurely → local min → low quality
Pragmatic Approach

- Good way to find out what optimization is possible
  - Run for long time and cool slowly
  - If can slow down cooling and get improvement
    - Demonstration haven’t found optimum, yet
- Once know good result this way
  - Can try to accelerate convergence w/out sacrificing quality

Range Limit

- Want to tune so accepting 44% of the moves – Lam and Delosme DAC 1988
- VPR
  - Define Rlimit – defines maximum ∆x and ∆y accepted
  - Tune Rlimit to maintain acceptance rate
  - Rlimit^{new} = Rlimit^{old} \times (1 - 0.44 + α)
  - α is measured acceptance rate

Range Limiting?

- Eguro alternate [DAC 2005]
  - define P = D - M
  - Tune M to control α

Range Limiting

- Eurgo, Hauck, & Sharma DAC 2005
  - Distance is P_{swap} = D - M
  - M adjusted
  - For 44% accept

Big Hammer

- Costly, but general
- Works for most all problems
  - (part, placement, route, retiming, schedule…)
- Can have hybrid/mixed cost functions
  - as long as weight to single potential
  - (e.g. wire/time from VPR)
- With care, can attack multiple levels
  - place and route
- Ignores structure of problem
  - resignation to finding/understanding structure

Summary

- Simulated Annealing
  - use randomness to explore space
  - accept “bad” moves to avoid local minima
  - decrease tolerance over time
- General purpose solution
  - costly in runtime
Admin

- Reading for Monday online

Big Ideas:

- Use randomness to explore large (non-convex) space
  - Sample various parts of space
  - Avoid becoming trapped in local minimum
- Technique
  - Simulated Annealing