ESE535:
Electronic Design Automation

Day 10: February 18, 2013
Placement II
(Simulated Annealing)

Preclass

- Squared wirelength for top placement?
- Squared wirelength for bottom placement?
- Number of swaps?
- Squared wirelength after swap for each of these cases?
  - Sample from class
  - (everyone assigned 1 or 2)

Preclass Lesson

- Why can't we find an improving swap?

Today

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

Behavioral
(C, MATLAB, …)
↓ Arch. Select, Schedule
RTL
↓ FSM assign, Two-level, Multilevel opt., Covering, Retiming
Gate Netlist
↓ Placement, Routing
Layout
↓
Masks

Simulated Annealing

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

Key Benefit

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

Simulated Annealing

- At high temperature can move around
  - not trapped to only make "improving" moves
  - free energy from "temperature" allows exploration of non-minimum states
  - avoid being trapped in local minima
- As temperature lowers
  - less energy available to take big, non-minimizing moves
  - more local / greedy moves

Design Optimization

Components:

1. "Energy" (Cost) function to minimize
   – represent **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

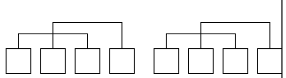
- Pick an initial solution
- Set temperature (T) to initial value
- while ($T>T_{min}$)
  - for time at T
    - pick a move at random
    - compute Δcost
    - if less than zero, accept
    - else if (RND<$e^{-\Delta cost/T}$), accept
  - update T

Cost Function

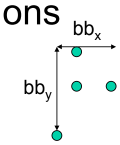
- Can be very general
  – Combine area, timing, energy, routability…
- Desirable characteristics:
  1. drive entire solution in right direction
     - reward **every** good move
  2. cheap to compute delta costs
     - *e.g.* FM
     - Ideally O(1)

Bad Cost Functions

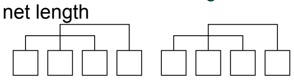
- Not reward every move:
  - size < threshold ?
  - Anything using max
    - channel width
    - critical path delay
    - How apply to example at right?
- Expensive update cost
  - rerun router on every move
  - rerun static timing analysis
    - *E.g.* recalculate critical path delay



Example Cost Functions

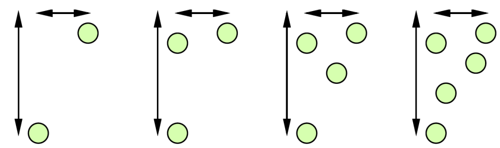


- Total Wire Length
  - Linear, quadratic…
- Σ Bounding Box (semi-perimeter)
  - Surrogate for routed net length
- Σ ($e^{channel\_density}$)
  - Dominate by largest density → approximate max
  - Rewards improvement in non-maximum channel
  - But reward is larger for denser channels
  - Can be computed incrementally



Bounding Box

- Why might not be satisfied with bounding box?
  - Should these all contribute same cost?
    - 2, 3, 4, or 5 terminals on same net



Example: VPR Wire Costs

- VPR Bounding Box

$$Cost = \sum_{i=1}^{Nets} \left(q(i) \times \left[bb_x(i) + bb_y(i) \right] \right)$$

Swartz, Betz, & Rose FPGA 1998

Original table: Cheng ICCAD 1994

Num Terminals	Correction Factor	Num Terminals	Correction Factor
1 ~ 3	1.00	15	1.69
4	1.08	20	1.89
5	1.15	25	2.07
6	1.22	30	2.23
7	1.28	35	2.39
8	1.34	40	2.54
9	1.40	45	2.66
10	1.45	50	2.79

Example: VPR Timing Costs

- Criticality(e)=1-Slack(e)/Dmax
- TCost(e)=Delay(e)*Criticality(e)^{CriticalityExp}
- Keep all edge delays in a table
- Recompute Net Criticality at each Temperature

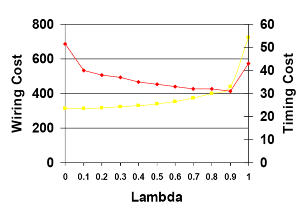
Criticality Exponent	Placement Estimated Critical Path (ns) (20 Circuit Geometric Average)	Wiring Cost (20 Circuit Geometric Average)
1	38.9	342.0
2	37.1	343.4
3	35.9	344.0
4	34.8	344.7
5	34.7	343.7
6	34.8	341.6
7	34.3	339.6
8	34.3	340.1
9	33.8	339.6
10	34.3	337.9
11	34.3	336.3

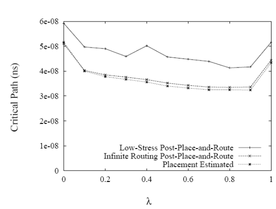
Marquardt, Betz, & Rose FPGA2000

Different Costs

- How might we deal with multiple costs?
  - E.g. Wire cost and Timing costs ?

VPR Balance Wire and Time Cost





Anneal Cost is weighted linear sum of Wire and Time

$$\Delta Cost = \lambda \left(\frac{\Delta TCost}{OldTCost} \right) + (1-\lambda) \left(\frac{\Delta WCost}{OldWCost} \right)$$

Marquardt, Betz, & Rose FPGA2000

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<$e^{-\Delta cost/T}$), accept
  - update T

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<$e^{-\Delta cost/T}$), accept
  - update T

Initial Solution

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

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<$e^{-\Delta cost/T}$), accept
  - update T

Details

- Initial Temperature
  - $T_0=-\Delta avg/\ln(P_{accept})$
  - $e^{-\Delta cost/T}$
  - $e^{-\Delta cost/T0}=e^{-\Delta cost/(-\Delta avg/\ln(Paccept))}$
  - Average move → $e^{\ln(Paccept)}$
    - Accepted with Probability P_{accept}
- When $P_{accept}=1$, moves randomize

Details

- Cooling schedule: options
  - fixed ratio: $T=\lambda T$
    - (*e.g.* λ=0.85)
  - temperature dependent
  - function of both temperature and acceptance rate
    - example to come
- Time at each temperature: options
  - 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
  - $Tnew=Told \times \gamma$
  - **Idea:** advance slowly in good α range
  - α is measured acceptance rate

α	γ
$\alpha > 0.96$	0.5
$0.8 < \alpha \leq 0.96$	0.9
$0.15 < \alpha \leq 0.8$	0.95
$\alpha \leq 0.15$	0.8

Betz, Rose, & Marquardt Kluwer 1999

Basic Algorithm Sketch

- 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 ($RND<e^{-\Delta cost/T}$), accept
  - update T

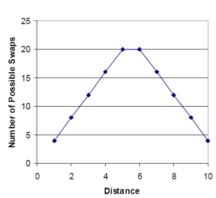
What happens when $T \rightarrow 0$?

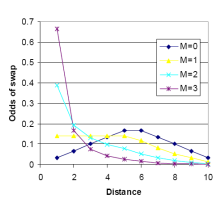
Range Limit

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

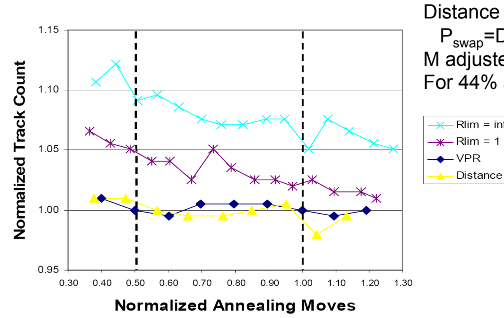
Range Limiting?

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





Range Limiting



Distance is $P_{swap}=D^{-M}$
M adjusted For 44% accept

Eguro, Hauck, & Sharma DAC 2005

Recall: VPR Timing Costs

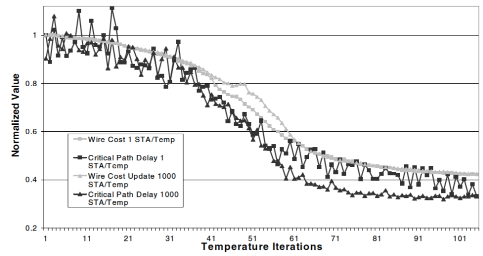
- Criticality(e)=1-Slack(e)/Dmax
- TCost(e)=Delay(e)*Criticality(e)$^{\text{CriticalityExp}}$
- Keep all edge delays in a table
- Recompute Net Criticality at each Temperature
- Why might be a problem?

Criticality Exponent	Placement Estimated Critical Path (ns) (20 Circuit Geometric Average)	Wiring Cost (20 Circuit Geometric Average)
1	38.9	342.0
2	37.1	343.4
3	35.9	344.0
4	34.8	344.7
5	34.7	343.7
6	34.8	341.6
7	34.3	339.6
8	34.3	340.1
9	33.8	339.6
10	34.3	337.9
11	34.3	336.3

Marquardt, Betz, & Rose FPGA2000

Stale Criticality

- Criticality becomes stale during moves



Eguro & Hauck DAC 2008

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 ($RND<e^{-\Delta cost/T}$), 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….

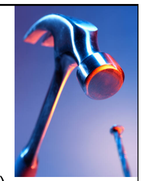
Practice

- 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

Big "Hammer"



- Costly, but general
- Works for most all problems
  - (part, placement, route, retime, 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

Big Ideas:

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

Admin

- Reading for Wednesday online
- Assignment 3 due Wednesday