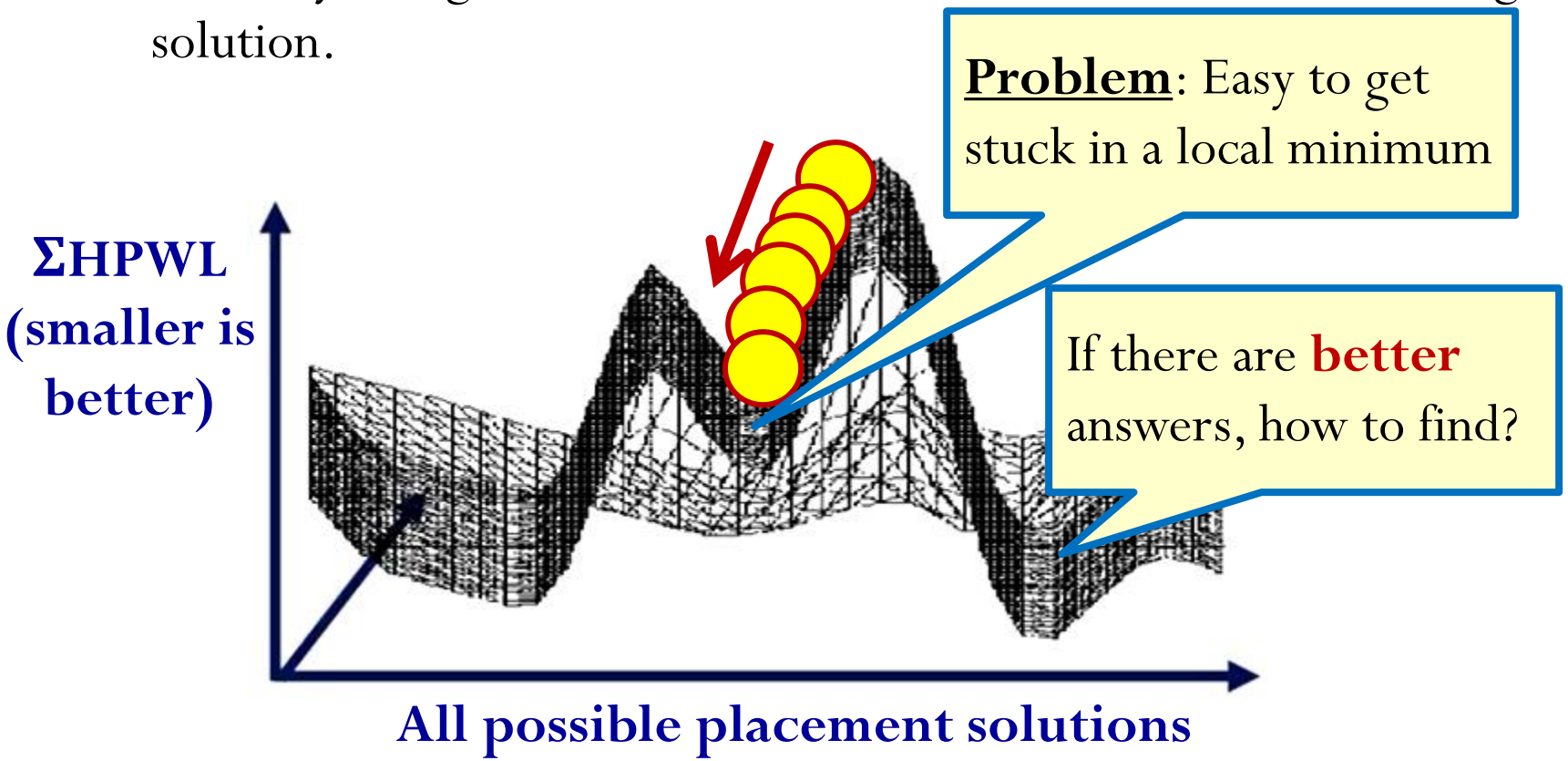
ECE6703J

Computer-Aided Design of Integrated Circuits

Simulated Annealing Placement

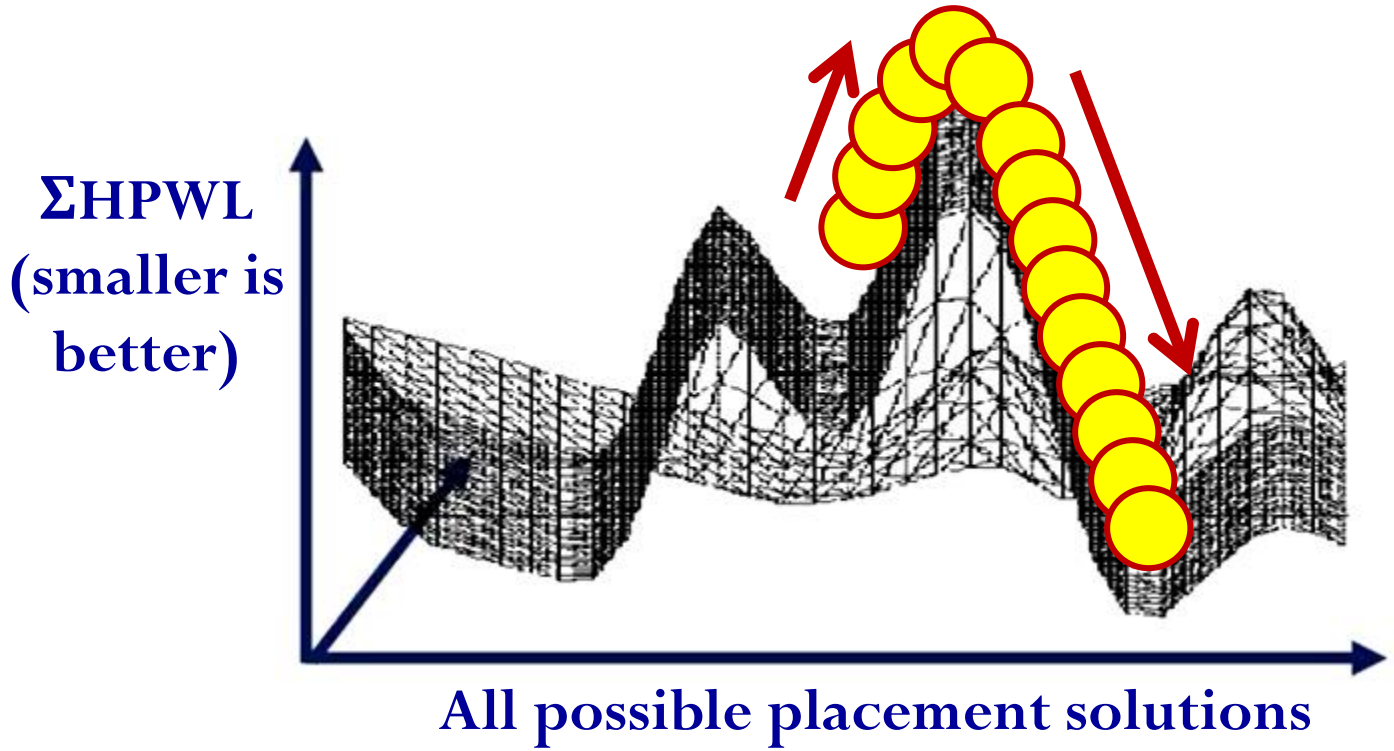
Iterative Improvement Placer: Problem?

- Imagine you could plot ΣHPWL for **all** possible placements.
  - Our random method only takes swaps that **improve** wirelength.
  - Can only find good answers **downhill** from our random starting solution.



Getting Stuck: Downhill vs Uphill Methods

- If we could go **uphill**, we could maybe get better placements.
- **New problem**: how to **control** this? Such changes make placement **worse**!

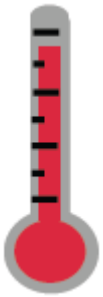


Random Iterative Improvement is Greedy

1. Randomly swap two gates
2. Evaluate ΔL = Change in ΣHPWL
3. Is $\Delta L < 0$?
   1) **Yes. Keep swap.** Go to 1.
   2) **No. UNDO swap.**

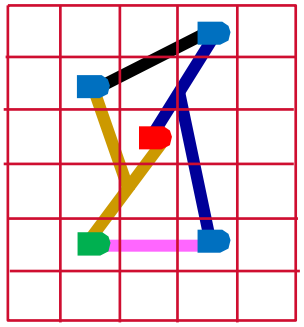
New Idea: Improvement with Hill-Climbing

1. Randomly swap two gates
2. Evaluate ΔL = Change in ΣHPWL
3. Is $\Delta L < 0$?
   1) **Yes. Keep swap.** Go to 1.
   2) **No…**
      - Evaluate function $P(\Delta L, T)$.
        Accept swap with probability P.

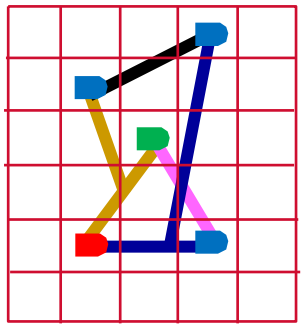


New:
Hill-climbing
control parameter:
T=temperature

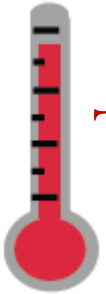
Idea: Probabilistic Swap Accept Criterion



Swap **Green** with **Red**



Change in length $\Delta L > 0$



T

$$e^{-\frac{\Delta L}{T}}$$

A probability value $0 < P < 1$

R = **uniform** random number between 0 and 1.

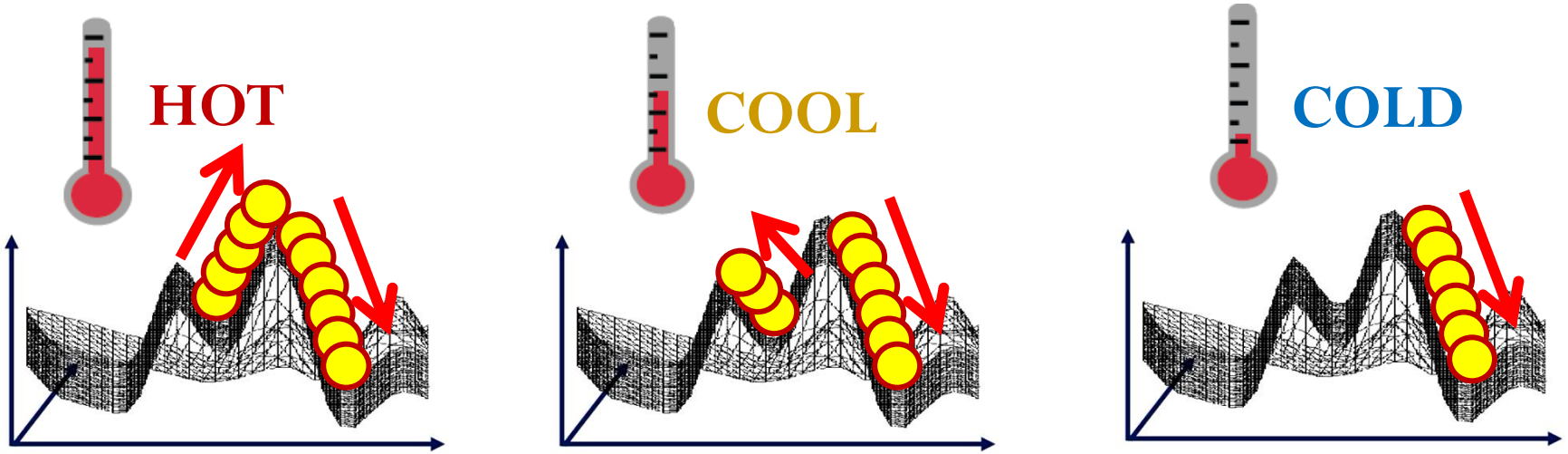
if(R < P) then **keep** swap
else **undo** swap

New Random Improvement Algorithm

- Almost same as old, greedy one, with 2 **big** changes.
  - Hill-climbing temperature T, start with T=Hot=BIG, **slowly** reduce T over many swaps.
    - Only after many swaps you reduce T, then do many swaps, reduce T again … (repeat).
  - If a swap makes wirelength **worse**, **randomly accept it**, with probability $P(\Delta L, T) = e^{-\frac{\Delta L}{T}}$.
    - Note: when T is BIG (initially), $e^{-\frac{\Delta L}{T}} \approx 1$, many worse improvements are accepted; when T is SMALL, $e^{-\frac{\Delta L}{T}} \approx 0$, many worse improvements are rejected.

New Random Improvement Algorithm

- Accepting probability $P(\Delta L, T) = e^{-\frac{\Delta L}{T}}$ versus T

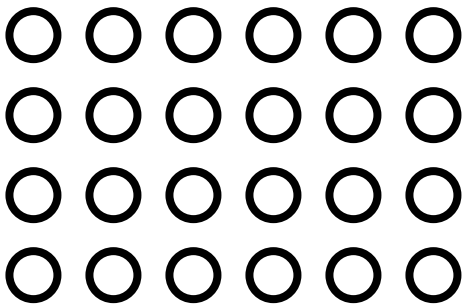


New Famous Algorithm: **Simulated Annealing**

Physical Analogy: Annealing

- Suppose you want to make a perfect crystal from a material.
  - **Perfect** = all atoms lined up on crystal lattice sites; **lowest energy** "state" of the atoms.

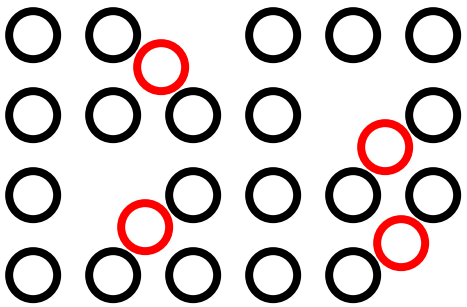
Perfect order,
has **minimum** energy



Physical Analogy: Annealing

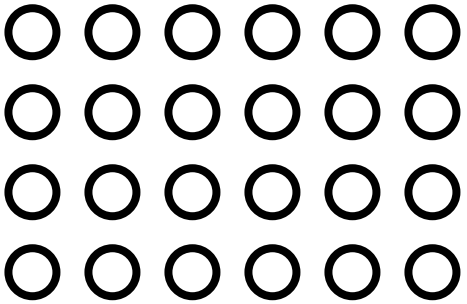
- How you get to the **perfect** order?
  - Get the material really **hot** (so atoms have energy to move around, even to bad places)…
  - … then, **cool** very, very slowly (so atoms settle in "good" spots).
- This process has a name: **Annealing**.

Imperfect order, has higher energy



Anneal

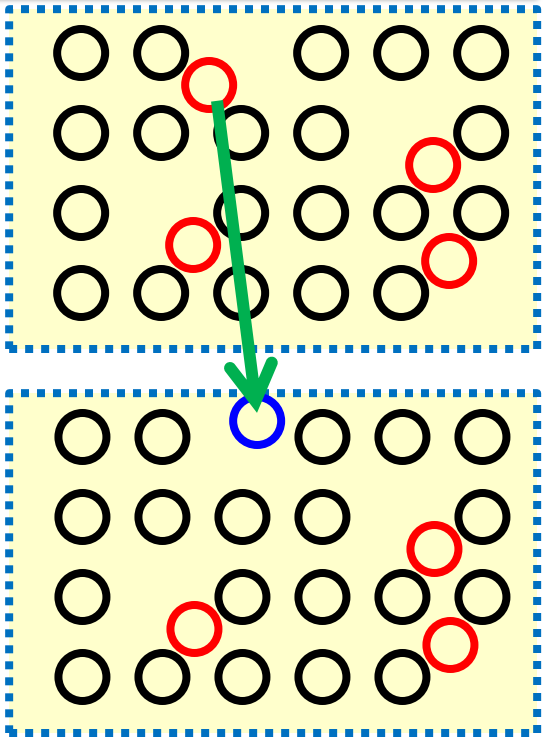
Perfect order, has minimum energy



Physical Analogy: Annealing

- How do you **simulate** this crystal with a computer program?

Move 1 atom.
Evaluate ΔE = **change in Energy**



Two situations: ΔE<0 and ΔE>0

In physics, what is probability of this atomic move, if ΔE<0?

Answer: 1

What is probability if ΔE>0?

Answer: $e^{-\frac{\Delta E}{KT}}$

(K is **Boltzmann constant**. In real physics, K converts units of Temperature to units of Energy)

Annealing versus Placement

- Annealing process in physics
  - Optimize energy E
  - Perturbation: move of a single atom
  - Temperature T
  - $\Delta E < 0$, keep the move
  - $\Delta E > 0$, accept move with probability $e^{-\frac{\Delta E}{KT}}$
- Placement
  - Optimize the total HPWL L
  - Perturbation: swap two gates
  - Temperature T
  - $\Delta L < 0$, keep the swap
  - $\Delta L > 0$, accept swap with probability $e^{-\frac{\Delta L}{T}}$

This New Method: Simulated Annealing

- General optimization method, used widely in VLSI CAD
  - Invented at IBM in early 1980s.
  - S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". Science 220 (4598): 671–680, 1983.

Annealing Placer: Pseudo-code

```
Start with a random initial placement with wirelength = ∑HPWL(net);
T = temperature = hot;     frozen = false;
while ( ! frozen ) {
    for (s=1; s< M* #gates in layout;  s++) { // M is how many swaps-per-gate
        Swap two random gates Gi and Gj;
        Compute ΔL = [∑HPWL(net) after swap] - [∑HPWL(net) before swap];
        if (ΔL < 0 ) then accept this swap;  //this is a new, better placement
        else {
            if( uniform_random() < exp( -ΔL/T ) )
            then accept this 'uphill' swap; //this is a new, worse placement
            else undo this 'uphill' swap;
        }
    }
    if (∑HPWL(net) still decreasing over the last few temperatures)
    then T = 0.9 * T;    // cool the temperature; do more gate swaps
    else  frozen = true;
}
return (final placement as best solution);
```

Randomly accept swap

Annealing temperature cooling outer loop

Probabilistic Acceptance of Swaps

```
if( uniform_random() < exp( -ΔL/T ) )
   then accept this 'uphill' swap; //this is a new, worse placement
   else undo this 'uphill' swap;
```

- This little piece of code implements a **famous idea**
  - **Idea**: "randomly accepting" a perturbation of system, with specific probability, will correctly simulate the real physics of that real system.
  - **Name**: **Metropolis Criterion**

Metropolis Criterion: Behavior

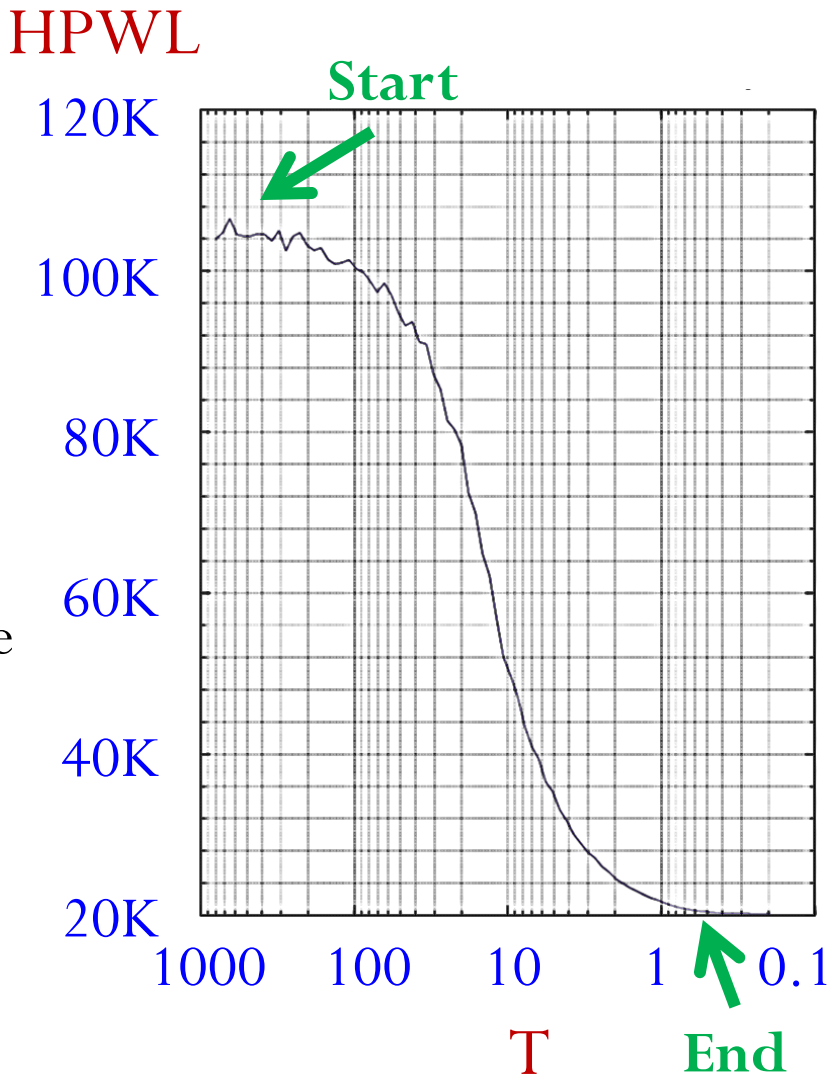
if(uniform_random() < exp(-ΔL/T))
then accept this 'uphill' swap; *//this is a new, **worse** placement*
else undo this 'uphill' swap;

- How does this really work?
  - It really is **random**.
  - You might accept swap, you might not. Depends on **random number** you generate.
  - Suppose $T = 10$ and $\Delta L = +20$. $\exp(-\Delta L/T) = 0.14$. So, 0.14 probability to accept swap with $\Delta L = +20$.
  - Suppose that at this temperature T, you try 100,000 **different** swaps, among which 5,000 evaluate to $\Delta L = +20$. Then, roughly $0.14 \times 5000 = 700$ of them will be accepted.
  - If you **change** the random number seed and run code again, you get a **different** result…
  - …but, if you still have 5,000 moves with $\Delta L = +20$, still expect ~700 (**different**) accepts.

How Well Does Annealing Work?

- It works great
  - Same ~2500 gate benchmark
  - HOT=800, M=100 moves/gate
  - Do 100×2500 swaps/temperature
  - About 30% less HPWL wirelength!

- Typical annealing "cost" curve
  - X axis is T=temperature, **LOG** scale
  - Y axis is ΣHPWL
  - Cool fact: curves **always** look like this!



A Few Facts About Simulated Annealing

- Does annealing always find the **best global** optimum?
  - NO. It is just good at avoiding a lot of local minimums.
- Does annealing work on **every type** of optimization problem?
  - NO. But it does work on many. However, it is not always the most efficient option.
- Is annealing always **slow**– doing all those many swaps over many temperatures?
  - NO. Lots of engineering tricks to speed it up.

A Few Facts About Simulated Annealing

- Do you need to set all the parameters – HOT, M swaps/gate – by **trial and error**?
  - NO. There are fancy adaptive techniques to determine these automatically.

- What happens if running annealer several **different** times, with different random seeds?
  - You get a **different** (random) answer each time.
  - To get a better solution, run it **multiple** times!

Simulated Annealing Placement: Summary

- **Simulated annealing** is a very famous, successful method.
  - Much better than simple random iterative improvement.
  - Dominant method for placers in 1980s, 1990s
  - And still today – used in lots of other VLSI CAD tasks.
  - Important to understand!

- But… **not** how we really do placers today
  - Annealing works well for up to 100K – 500K gates
  - Annealing too inefficient for things with 1M+ gates
  - Need yet **another** new set of ideas…