Optimization Strategies

General search techniques to solve combinatorial optimization problems

Branch-and-Bound: Exhaustive search

– Branch over all possible solutions, pruning the search by computing bounds

Simulated Annealing: Hill climbing algorithm

– Model the physical annealing process which results in increasing the "toughness" of metals

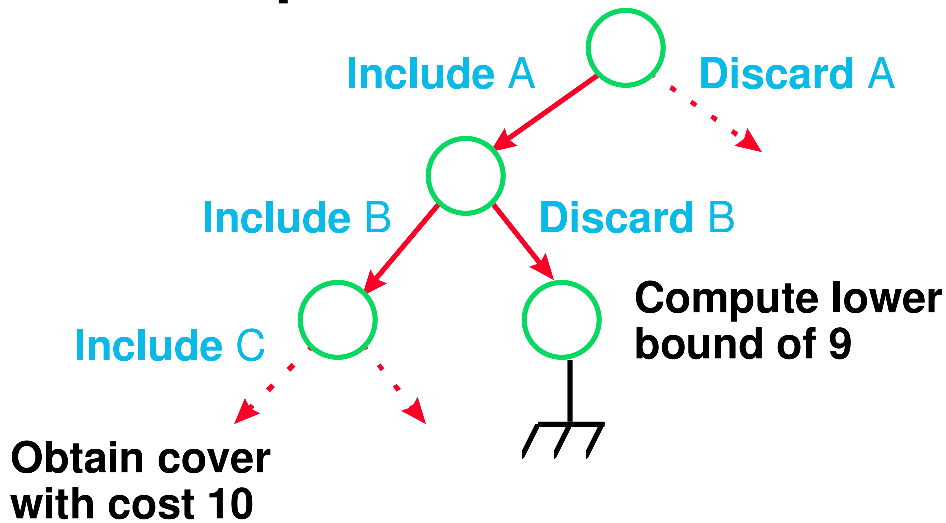
Simulated Evolution or Genetic Algorithms:

– Model the (human) evolution process which results in perfecting living beings

and other contenders …

Branch-and-Bound

Enumerate all possible solutions in search tree



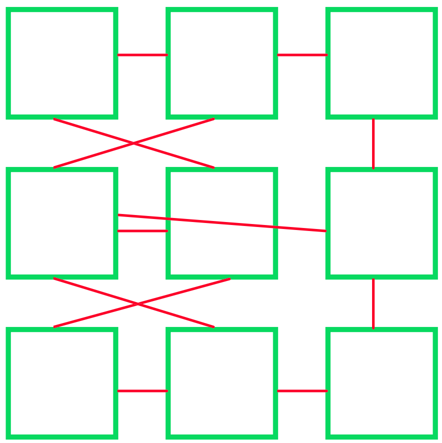
Backtrack to most recently made decision, and explore other possibilities

Reduce computational effort by lower bounding

Force-Directed Methods

Based on classical mechanics

Used mainly for cell placement problems

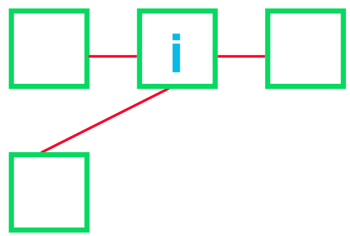


Find placement to minimize <u>total</u> net length, or other cost function reflecting routability

Formulating Force Equations

Modules 1, 2, …, N with

- m_i mass of module i
- X_i X position of module i (continuous variable)
- K_{ij} Attractive constant between module i and j
- F_i Net force on module i from rest of modules



Proportional to number of nets between modules i and j

Hooke's law:

$$m_i \frac{d^2 X_i}{dt^2} = F_i = -\sum_{j=1}^{N} K_{ij} (X_i - X_j)$$

Formulating Force Equations - 2

In the previous formulation, all modules will collapse to a single point!

- Need to place constraint that $X_i \neq X_j$

Use a repulsive force between modules that prevents overlap

repulsive force

attractive force

$$F_i = \sum_{j=1}^{N} \left[\delta_{ij} R - K_{ij} (X_i - X_j) \right]$$

$$\delta_{ij} = 1 / (X_i - X_j)$$

R is a critical constant, set heuristically

Solving the Equations

Transient simulation: Solve simultaneous non-linear differential equations by integration methods

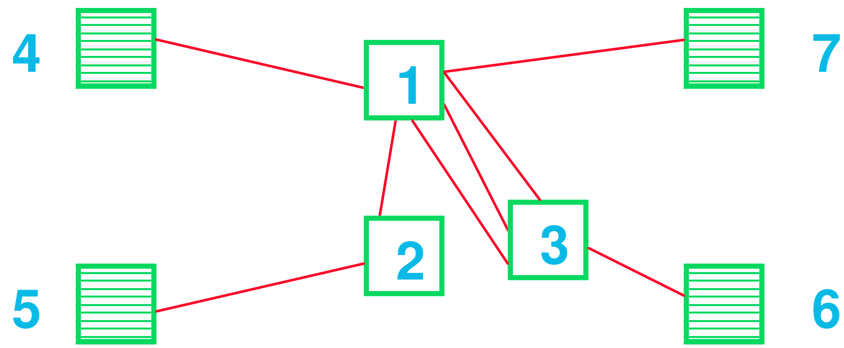
- **Effectively simulate equations from random starting point to steady state**

Steady state: Given that $\frac{d^2 X_i}{dt^2} = 0$ **at** $t = \infty$ **,**

solve the non-linear equations using Newton's method

faster

Example



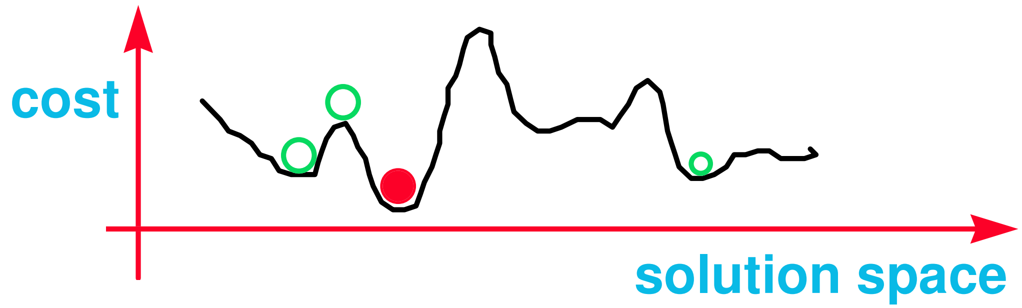
$m_1 = m_2 = m_3 = 1$
$m_4 = m_5 = m_6 = m_7 = \infty$

$X_4 = X_5 = 0$
$X_6 = X_7 = L$

$$\frac{d^2 X_1}{dt^2} = R\left[\frac{1}{X_1} + \frac{1}{X_1 - L} + \frac{1}{X_1 - X_2} + \frac{1}{X_1 - X_3} \right] - \left[X_1 + (X_1 - L) + 3\,(X_1 - X_3) + (X_1 - X_2) \right]$$

Hill-Climbing Algorithms

In order not to get trapped in local minima, hill-climbing in an algorithm is desirable



Need to accept worse solutions, i.e., make "bad" moves in a search strategy, so as to get to the global minimum

Probabilistic hill-climbing algorithms occasionally accept bad moves

Physical Annealing

- Take a metal and heat it to a high temperature.
- Allow it to cool slowly ; Metal is annealed to a low temperature.
- This makes metal tougher, because the atoms in the metal are in low-energy states after annealing.
- Brownian motion of atoms may result in atom moving from lower to higher energy states, especially at high temperatures.

Simulated Annealing

Optimization based on physical annealing.

Cost function analogous to energy of atoms.

<u>Generate</u> random moves.

Initially, at high temperatures, <u>accept</u> moves that increase cost.

As temperature decreases, the probability of accepting “bad” moves decreases.

Stop when cost does not decrease for a while.

The Algorithm

```
T = T0 ;   X = X0 ;
while ( stopping criterion is not satisfied ) {
   for ( a certain number of moves ) {
      Y = generate( X );
      if ( accept( C(Y), C(X), T ) )
         X = Y ;
   }
   T = Update( T ) ;
}
```

X, Y: solutions   T: temperature   C(): cost function

Accepting or Rejecting a Move

```
accept( C(X), C(Y), T ) {
    ΔC = C(Y) - C(X) ;
    if ( ΔC ≤ 0 ) return( YES ) ;
    else {
        y = exp( – ΔC / T ) ;
        r  = random( 0, 1 ) ;
        if  ( r < y ) return( YES ) ;
        else return( NO ) ;
    }
}
```

Cell Placement Problem

One of the first successful implementations of simulated annealing

Cost function: Total net length
Penalty for cell overlap

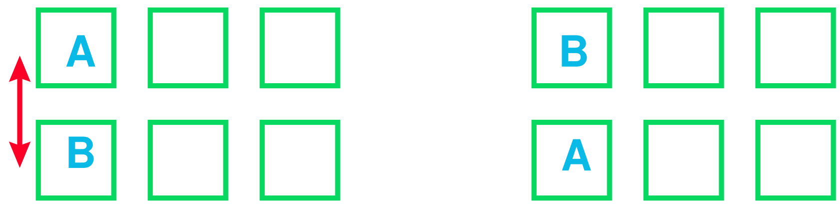
Generation: Cell interchanges
Cell displacements

Inner loop: 10 times number of cells at each temperature point

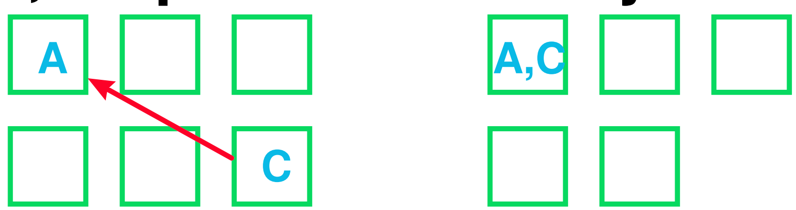
Outer loop: Stop when cost has not changed for 3 temperature points

Penalty Function

Begin with a placement with no overlap and interchange cells



Explore space better using displacements; however, displacements may cause overlap



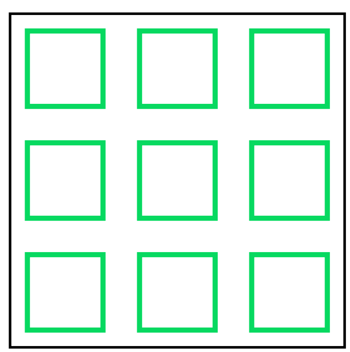
To avoid overlap in final placement, penalize overlap using penalties in cost function

Range Limiting

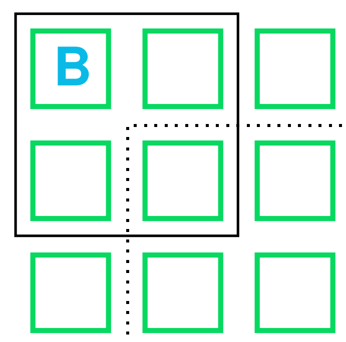
Simplistic strategy for generating moves is random interchange or displacement

- Interchanges of faraway cells, and large displacements are more likely to increase cost than smaller changes

Implement a range limiter to generate moves that are likely to be accepted



high temperature, all interchanges



low temperature,only neighboring

Genetic Algorithms

Simulate the evolution process.

Begin with a population of size p.

Reproduce to produce a new population, also of size p.

Mutate m randomly selected members of the population.

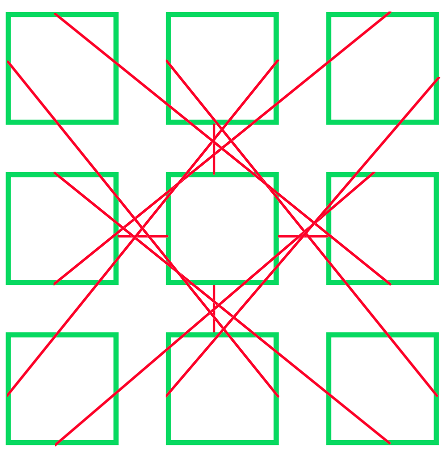
Iterate for g generations.

Select the fittest member of the population.

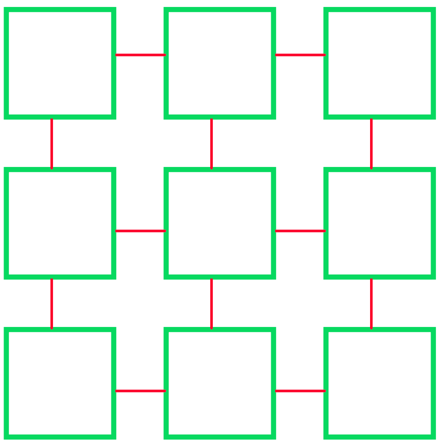
Fitness of Members

Each member of the population is merely a solution of the optimization problem

The fitness of a member is inversely proportional to the cost of the solution



Cost = 28

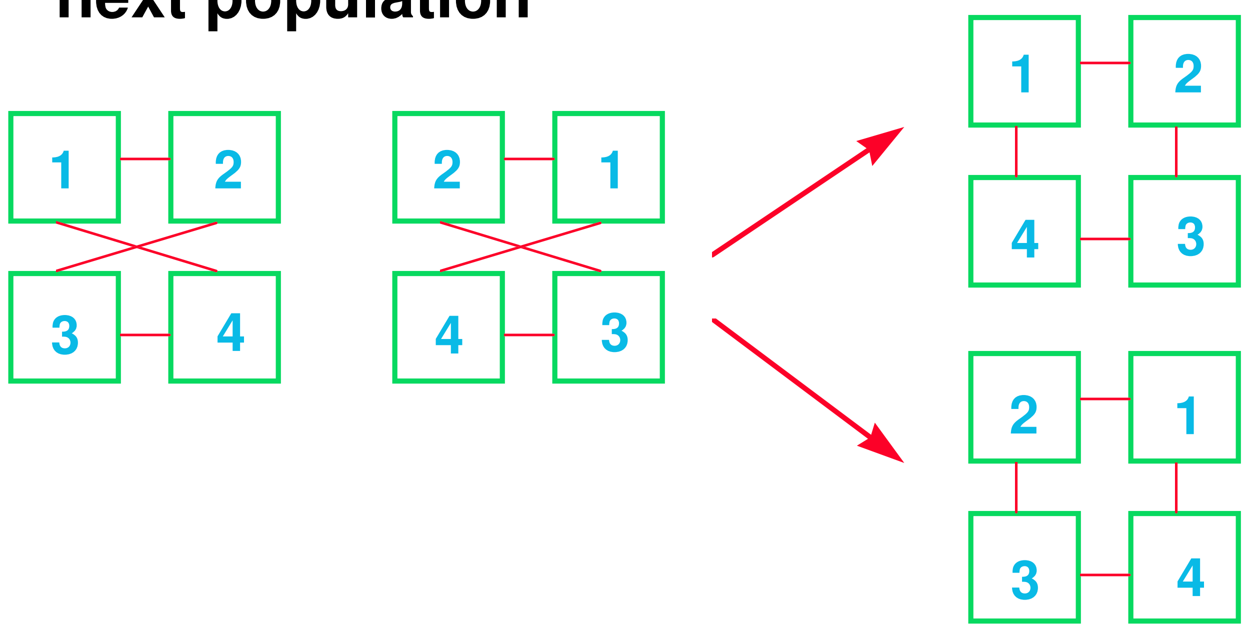


Cost = 12

Reproduction Function

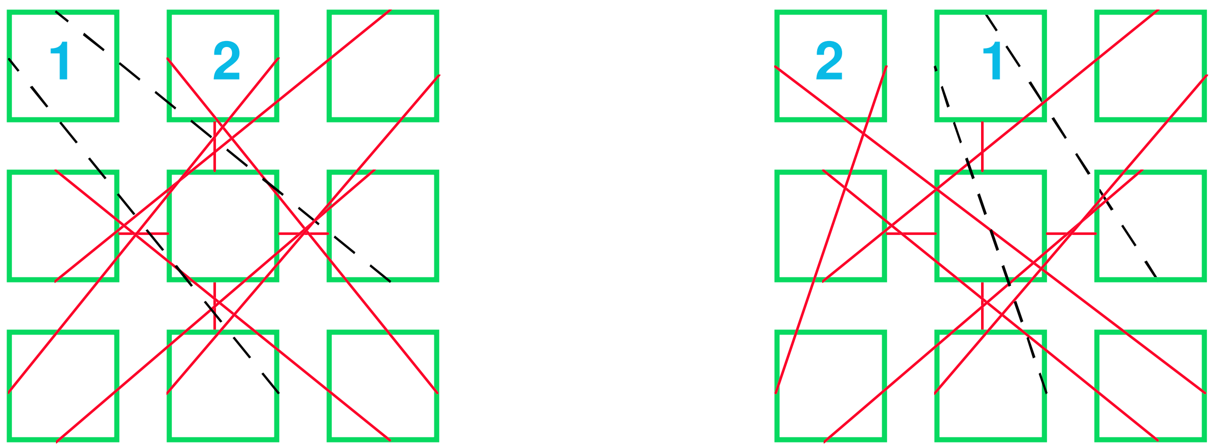
Randomly pair members of the current population, and generate p new elements

Pick the best fit half of the 2p elements for the next population



Mutation

Each time you generate a new member, with probability m, change the member's configuration



Cost = 28

Cost = 25

The Lowdown ...

General combinatorial optimization techniques are easy to implement, but there is “no free lunch” !

In order to get an efficient heuristic algorithm, need to tailor the basic technique for specific optimization problem, and problem instance

- Branch-and-bound needs efficient bounding
- Simulated annealing needs fast, incremental cost evaluation, good move set
- Genetic algorithms??