# Stochastic Search

Hill Climbing, Simulated Annealing,
WALKSAT, Ant Colony Optimization

Vincent A. Cicirello

Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

cicirello@ri.cmu.edu

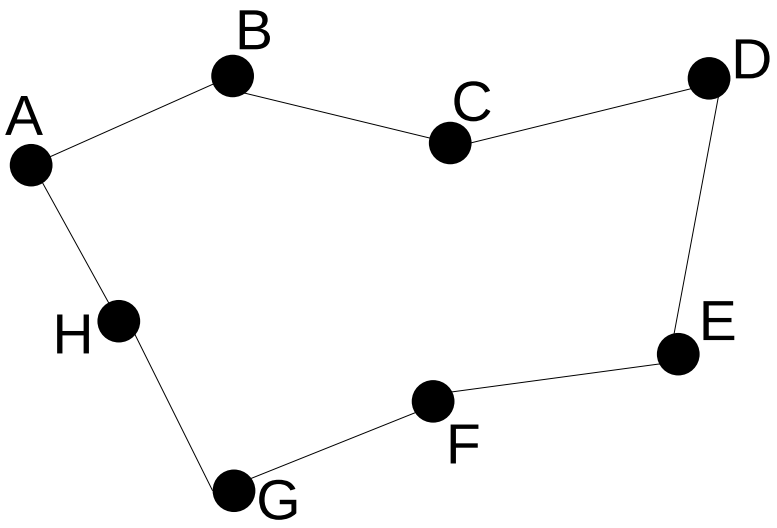
# Example Problems

## SAT

$$A \vee \neg B \vee C \ \wedge \ \neg A \vee C \vee D \ \wedge$$
$$B \vee D \vee \neg E \ \wedge \ \neg C \vee \neg D \vee \neg E \ \wedge$$
$$\neg A \vee \neg C \vee E$$

hundreds of variables, thousands of clauses

## TSP




hundreds or even thousands of cities

## Informal Problem Characteristics

Problems in which:

- There is some combinatorial structure to optimize

- There is some cost function: Structure $\rightarrow$ Real, to be optimized or at least a reasonable solution is to be found

- Searching all possible structures is intractable

- DFS approaches are too expensive.

- There is no known algorithm for finding optimal solutions efficiently

- Similar solutions have similar costs.

## Iterative Improvement

Intuition: Consider the possible configurations to be laid out on the surface of a landscape. The height of each configuration on that landscape is determined by a measure of its quality. We want to find the highest point.

We don't particularly care how we get there.

"Iterative Improvement" methods:

- Start at random configuration
- Iteratively consider various moves
- Accept some and reject some
- When stuck, restart.

We must invent a MoveSet that describes what moves are allowed from a given configuration.

## Hill-climbing

Hill-climbing: Try to maximize Eval(X) by moving to the highest configuration given our moveset. If all successors are lower than we are at a "local optimum."

1. Let $X$ := initial config
2. Let $E$ := Eval($X$)
3. Let $N$ := moveset-size($X$)
4. For $i$ := 1 ... $N$
   Let $E_i$ := Eval(move($X$,$i$))
5. If all $E_i \leq E$, terminate, return $X$
6. Else let $i^*$ = argmax($E_i$)
7. Let $X$ := move($X$,$i^*$)
8. Let $E$ := $E_{i^*}$
9. Goto 3

## Randomized Hill-climbing

1. Let $X$ := initial config
2. Let $E$ := Eval($X$)
3. Let $i$ := random move from moveset
4. Let $E_i$ := Eval(move($X$,$i$))
5. If $E < E_i$ then
   Let $X$ := move($X$,$i$)
   Let $E$ := $E_i$
6. Goto 3 until bored

What stopping criterion should we use?

What are some pros and cons compared with previous hill climber?

## Hill-climbing Issues

- Trivial to program
- Requires no memory (no backtracking)
- MoveSet design is critical
- Evaluation function design is critical (avoid dense local optima and plateauz when possible)
- Enormous MoveSet may result in inefficiency

- Tiny MoveSet may result in getting stuck easily
- What if its cheaper to evaluate an incremental change of a previously evaluated object? Can we do that with hill-climbing?

## Hill-climbing example: GSAT

$$A \vee \neg B \vee C$$
$$\wedge$$
$$\neg A \vee C \vee D$$
$$\wedge$$
$$B \vee D \vee \neg E$$
$$\wedge$$
$$\neg C \vee \neg D \vee \neg E$$
$$\wedge$$
$$\neg A \vee \neg C \vee E$$

Configuration: (1,0,1,0,1)

Maximize: Eval(Config) = # of satisfied clauses

MoveSet: flip any 1 variable

## WALKSAT: Randomized GSAT

- Pick a random unsatisfied clause

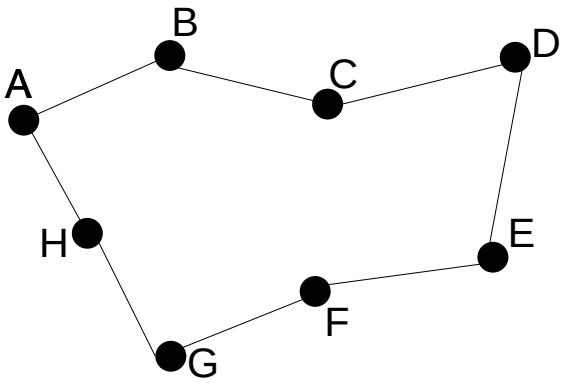
- Consider 3 moves: flipping each variable.

- If any improve Eval, accept the best.

- If none improve Eval, then 50% of the time pick the least bad move and 50% of the time pick a random one.

# Hill-climbing Example: TSP

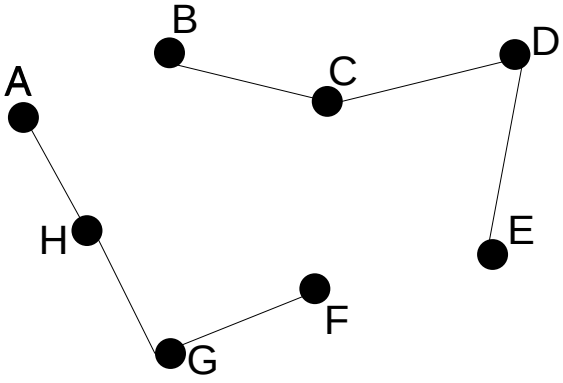
Configuration:

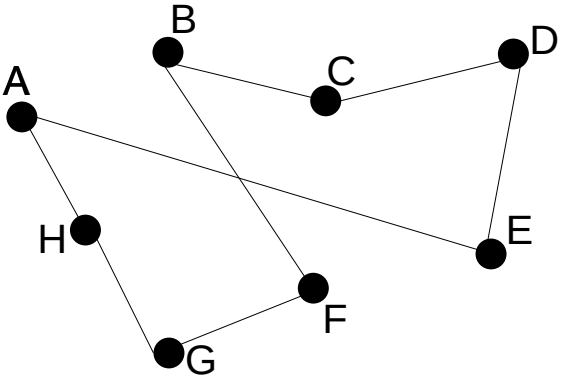



Minimize: Eval(Config) = length of tour

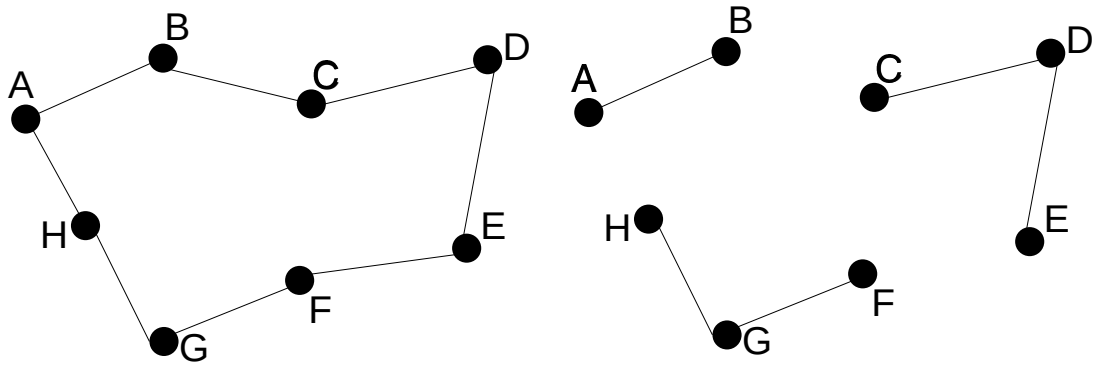
MoveSet: 2-change, ..., $k$-change

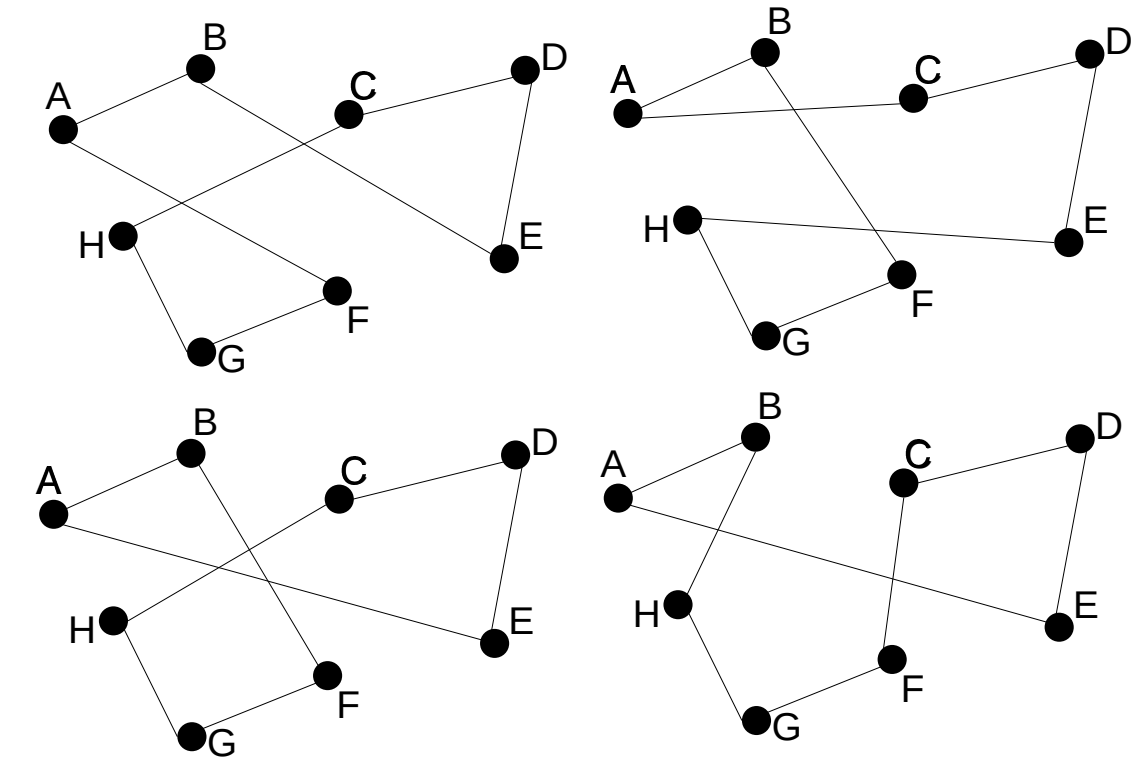
Example: 2-change



# 3-change Example



## Hill-climbing Example: TSP

This class of algorithms for the TSP is usually referred to as $k$-opt (MoveSet: 2-change, ..., $k$-change) for some constant $k$.

Lin showed empirically:

- 3-opt solutions are much better than 2-opt

- 4-opt solutions are not sufficiently better than 3-opt to justify the extra compute time

- A 3-opt tour for the 48-city problem of Held and Karp has about a probability of 0.05 of being optimal (100 random restarts will yield the optimal solution with probability 0.99)

- Further for his particular class of TSP, a 3-opt tour is optimal with probability $2^{-n/10}$ where $n$ is number of cities.

There is no theoretical justification for any of these results.

## Simulated Annealing

1. Let $X$ := initial config
2. Let $E$ := Eval($X$)
3. Let $i$ := random move from moveset
4. Let $E_i$ := Eval(move($X$,$i$))
5. If $E < E_i$ then
   Let $X$ := move($X$,$i$)
   Let $E$ := $E_i$
   Else with some probability, accept the move anyway
   Let $X$ := move($X$,$i$)
   Let $E$ := $E_i$
6. Goto 3 until bored

This may make the search fall out of mediocre local minima and into better local maxima.

## Simulated Annealing

How should we choose probability of accepting a worsening move? Here are some ideas:

1. Probability = 0.1
2. Probability decreases over time
3. Probability decreases over time and also as $E - E_i$ increases

## Simulated Annealing

If $E_i \geq E$ then definitely accept the change

If $E_i < E$ then accept the change with probability:

$$\exp(\frac{-(E-E_i)}{T_i})$$

(called the Boltzman distribution)

where $T_i$ is a "temperature" parameter that gradually decreases. Typical cooling schedule: $T_i = T_0 \cdot r$

High temp: accept all moves (random walk)
Low temp: Stochastic hill-climbing

When no improvement after enough iterations, terminate.

## Simulated Annealing Issues

- Trivial to program
- Requires no memory (no backtracking)
- MoveSet design is critical
- Evaluation function design is often critical
- Annealing schedule often critical
- What if its cheaper to evaluate an incremental change of a previously evaluated object? Can we do that with simulated annealing?

## Simulated Annealing Discussion

Simulated Annealing was introduced in 1953 by Metropolis and is based on an analogy with the way that alloys manage to find a nearly global minimum energy level when they are cooled slowly.

Simulated Annealing is sometimes empirically much better at avoiding local minima than hill-climbing.

Not much opportunity to say anything formal about it (there is a proof that with an infinitely slow cooling rate, you'll find the global optimum).

## Ant Colony Optimization (ACO)

Idea: Set loose on your problem a colony of ants. Each ant stochastically builds a solution. A shared environment is updated to reflect quality of solutions and promising parts of search space. Repeat until all ants build same solution or until bored.

Based on an analogy of ant pheromone trail following.

A large part of "Swarm Intelligence: From Natural to Artificial Systems" (Bonabeau, Dorigo, Theraulaz, 1999) is dedicated to it.

## Traveling Sales-Ants

1. Let $\tau_{i,j} = C$ be the initial amount of pheromone between cities $i$ and $j$
2. Let $\eta_{i,j} = \frac{1}{d_{i,j}}$ where $d_{i,j}$ is the distance between cities $i$ and $j$
3. For each ant $k$
4. Let $J_k$ initially consist of all cities
5. Let $i$ := random start city
6. Remove $i$ from $J_k$
7. While $J_k$ not empty
8. Transition from $i$ to $j$ with probability
$$p_{i,j}^k(t) = \frac{(\tau_{i,j}(t))^\alpha (\eta_{i,j}(t))^\beta}{\sum_{h \in J_k} (\tau_{i,h}(t))^\alpha (\eta_{i,h}(t))^\beta}$$
9. Let $i = j$ and remove $j$ from $J_k$
10. Let $\tau_{i,j} = (1-\rho)\tau_{i,j} + \sum_{k \in A_{i,j}} \frac{1}{D_k}$ where $D_k$ is length of tour found by ant $k$ and $A_{i,j}$ are the ants that used edge $(i,j)$

# Traveling Sales-Ants

**Distance matrix**

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | | | | | | |
| B | | | | | | |
| C | | | | | | |
| D | | | | | | |
| E | | | | | | |
| F | | | | | | |

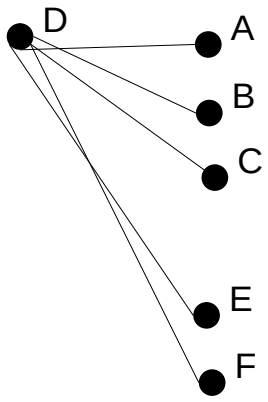
**Pheromone matrix**

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | | | | | | |
| B | | | | | | |
| C | | | | | | |
| D | | | | | | |
| E | | | | | | |
| F | | | | | | |

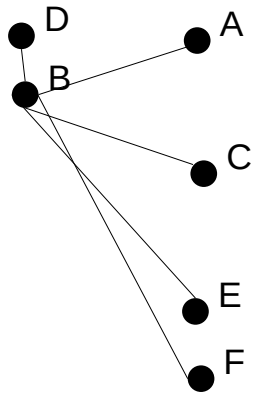
**Tour** $J_k$

A
B
C
D
E
F

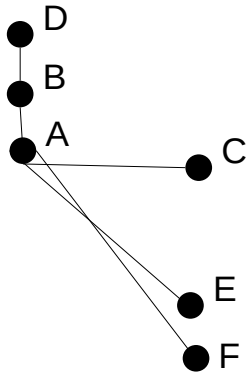
**Tour** $J_k$



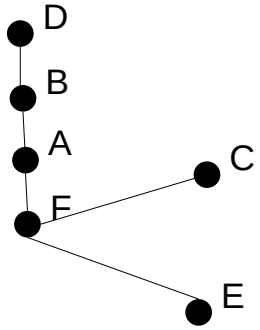

**Tour** $J_k$



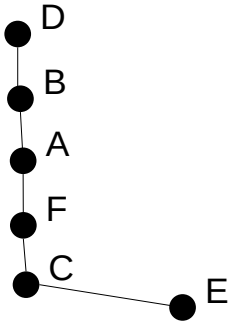

**Tour** $J_k$



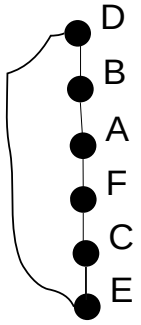

**Tour** $J_k$




**Tour** $J_k$




**Tour** $J_k$



## ACO Issues

- Transforming your problem into one of finding a path through some sort of graph structure is often critical.
- Parameter tuning is often critical to avoid either early convergence with bad results or very long slow convergence.
- There exist different variations of city transition rule. Which to use?
- There exist variations that allow only the best ant to update the tour, the $k$ best ants to update the tour, ants with bad solutions to negatively reinforce the tour, and so forth. Which to use?
- What if our problem has no obvious transformation to a TSP-like problem? Can we still use ACO? What about for SAT?