Artificial Intelligence

CS4365 --- Fall 2022

Local Search

Instructor: Yunhui Guo

Improvements to Basic Local Search

- Issue: How to move more quickly to successively higher plateaus and avoid getting “stuck” / local minima

- Idea: Introduce uphill moves (“noises”) to escape from long plateaus (or true local minima).
- Strategies:
  - Simulated Annealing
  - Random-restart hill-climbing
  - Tabu search
  - Local beam search
  - Genetic Algorithms

Variation on Hill-Climbing

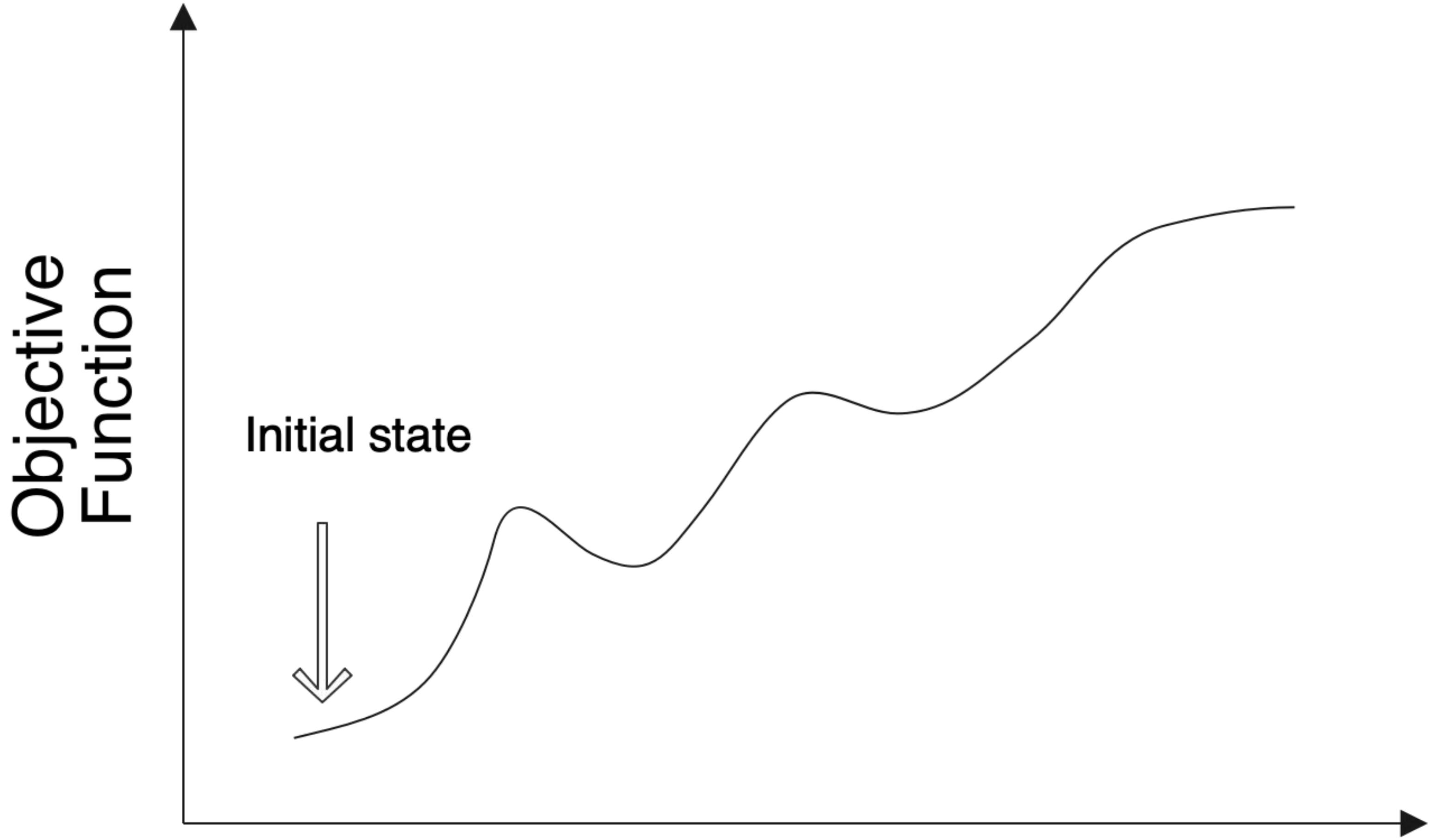
- Random restarts: simply restart at a new random state after a pre-defined number of local steps

- Tabu: prevent returning quickly to the same state.

  - Implementation: Keep fixed length queue (“tabu list”): add most recent step to queue; drop “oldest” step. Never make step that’s currently on the tabu list

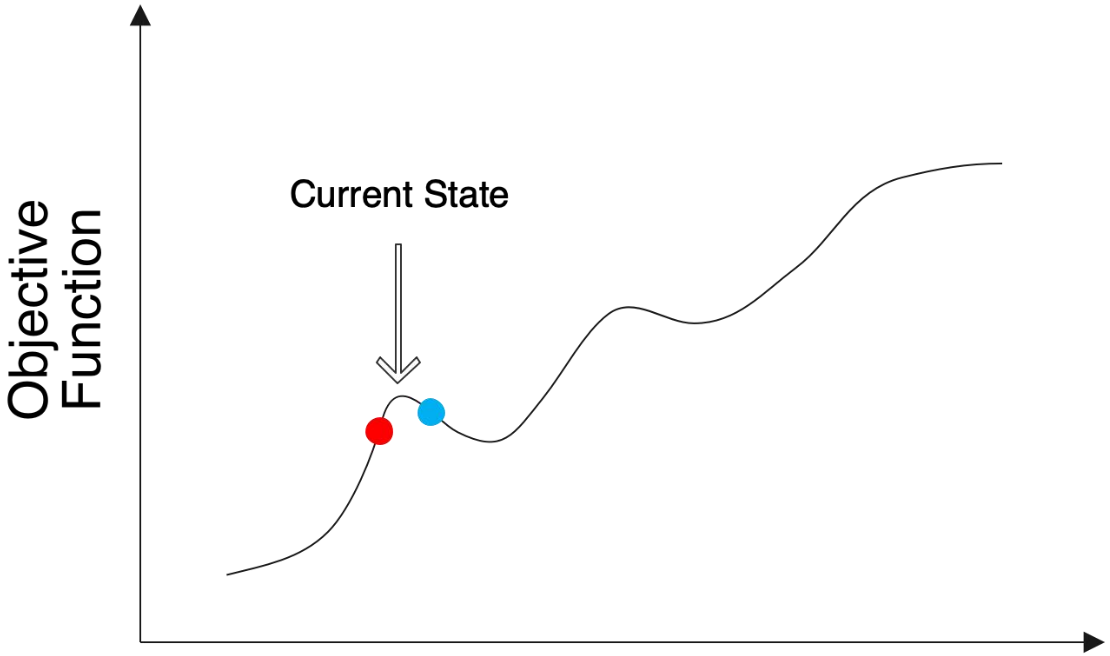
  - Uphill moves are acceptable if no downhill moves are available

Simulated Annealing



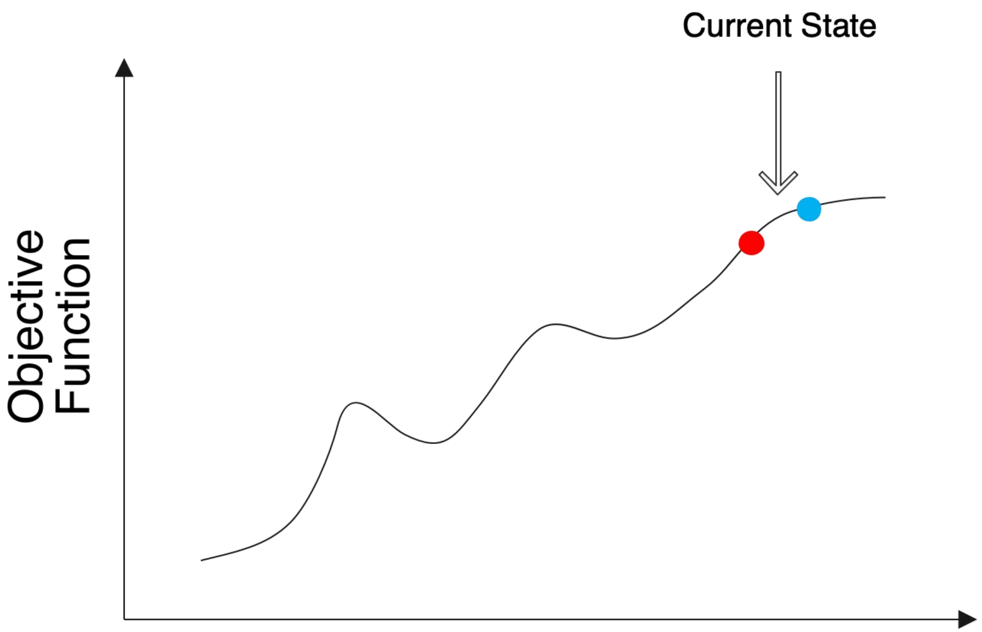
Simulated Annealing

- Idea:
  - Use conventional hill-climbing techniques, but occasionally take a step in a direction other than that in which the rate of change is maximal.



Simulated Annealing

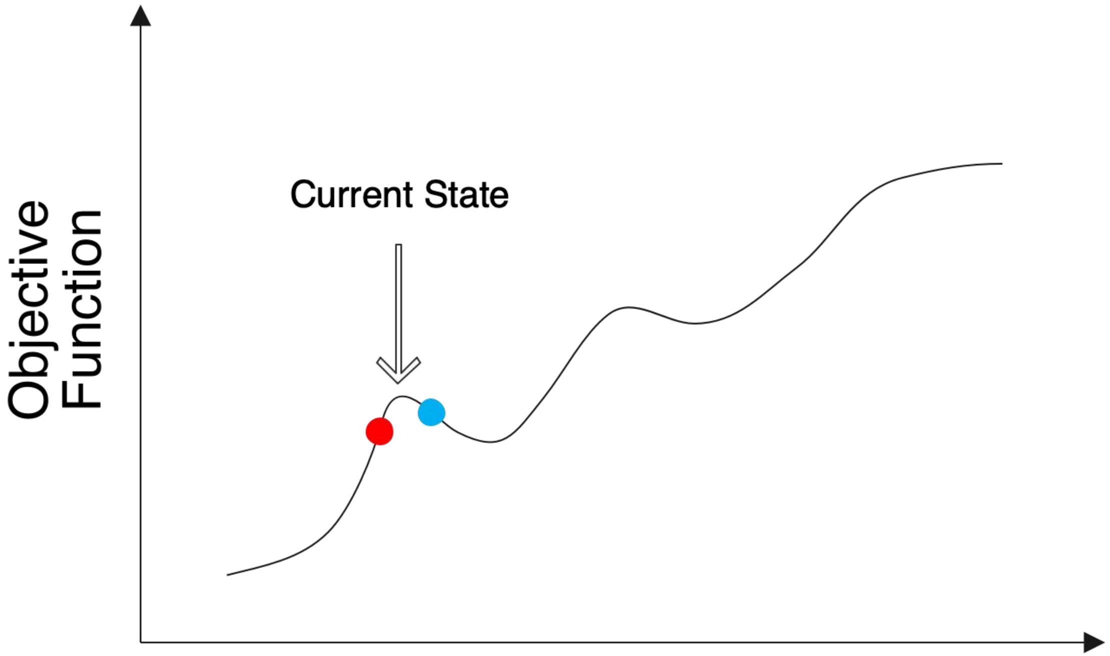
- Idea:
  - Use conventional hill-climbing techniques, but occasionally take a step in a direction other than that in which the rate of change is maximal.



Simulated Annealing

As time passes,

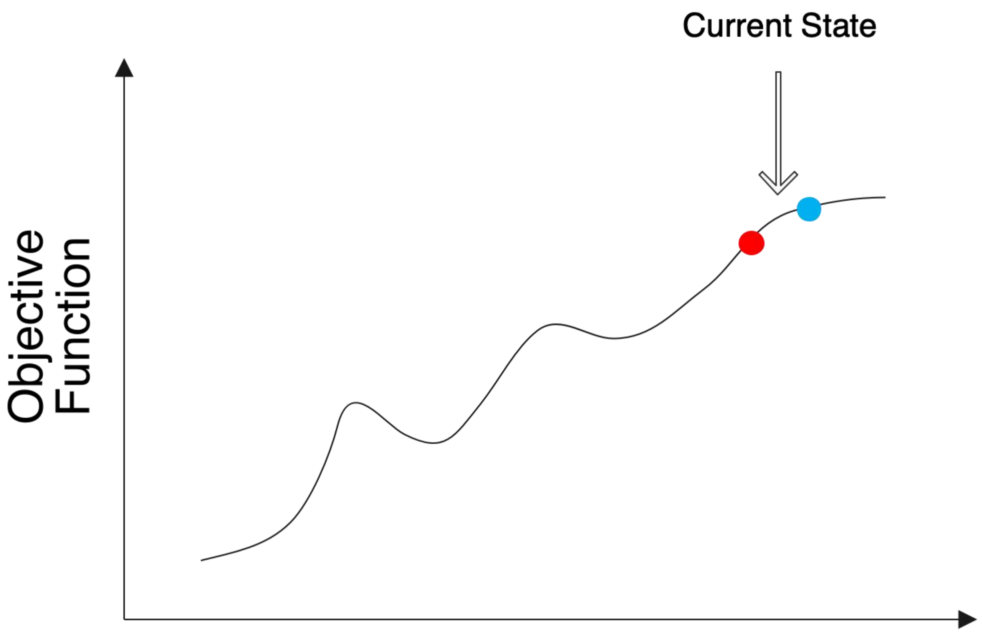
- The size of any down-hill step taken is decreased.



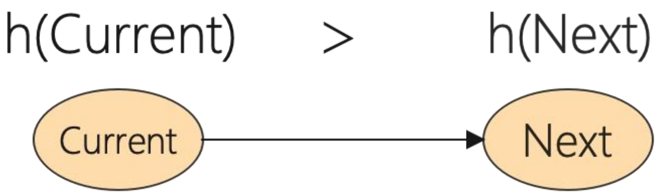
Simulated Annealing

As time passes,

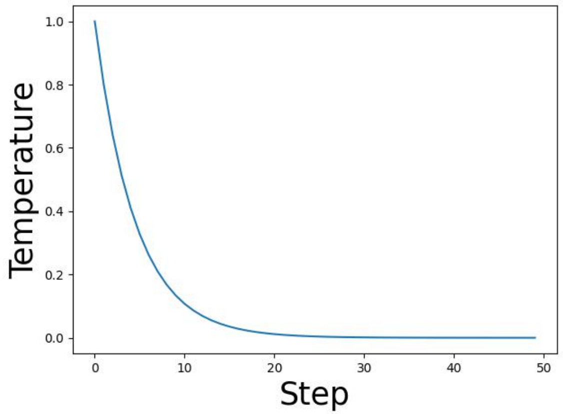
- The size of any down-hill step taken is decreased.
- The probability that a down-hill step is taken is gradually reduced



Simulated Annealing



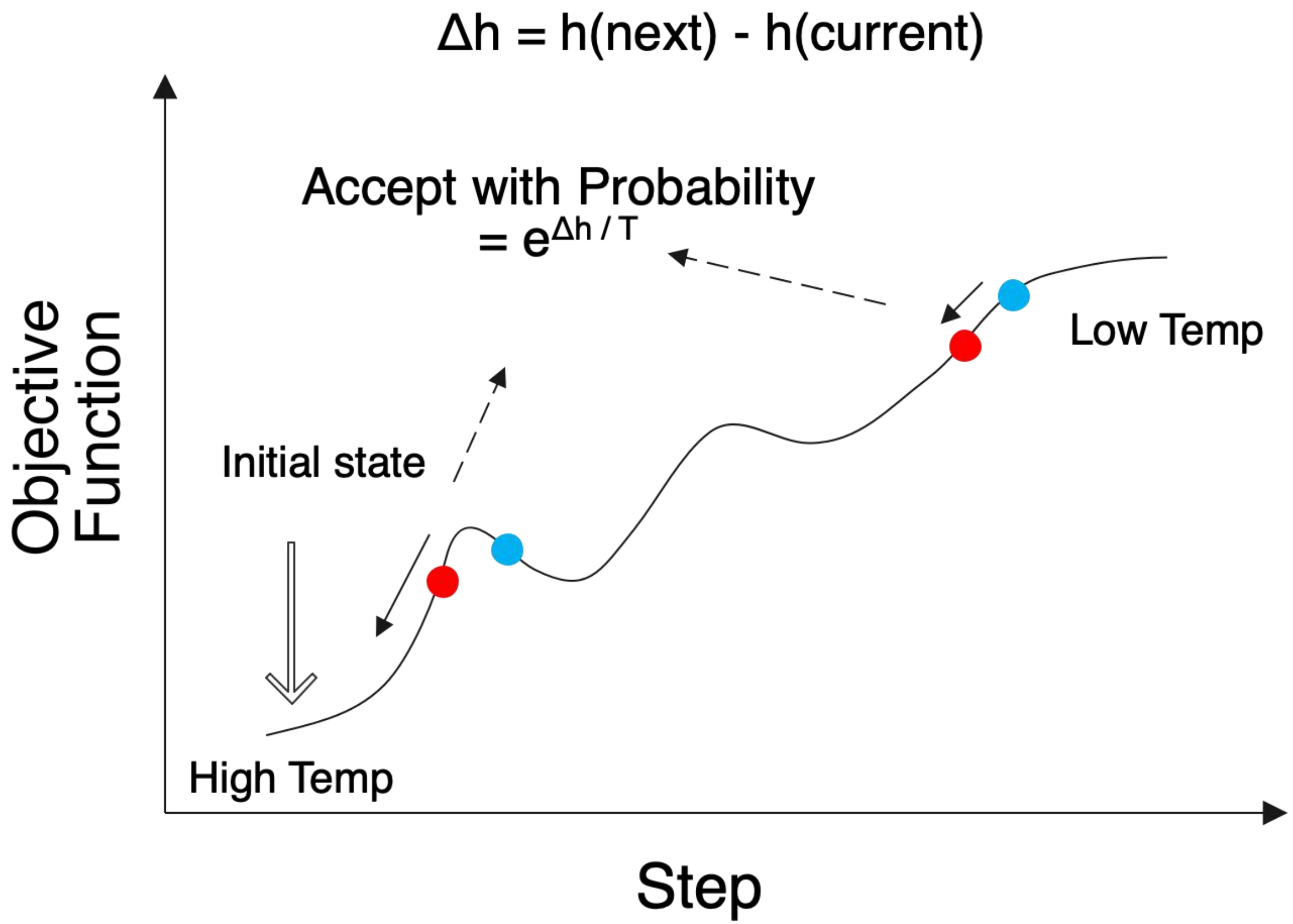
- Intuition:
  - The probability of move to a bad state should decrease exponentially with the “badness” h(Next) - h(Current)
  - The probability decreases as the “temperature” T goes down
- How to schedule the “temperature”?
  - Initially with a high temeprature and gradually decays
  - e.g. $T_k = T_0 \cdot \beta^k$ ($\beta < 1$)

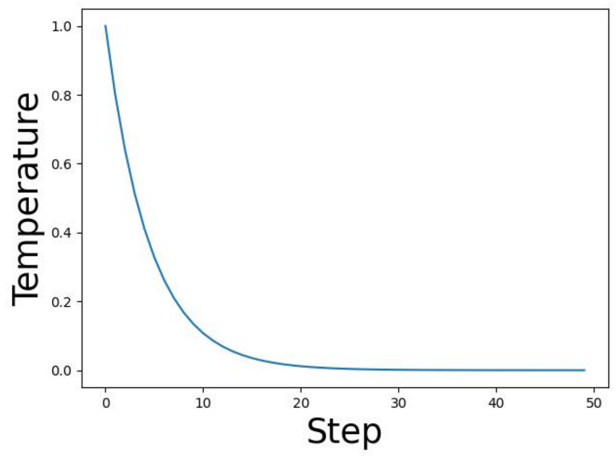


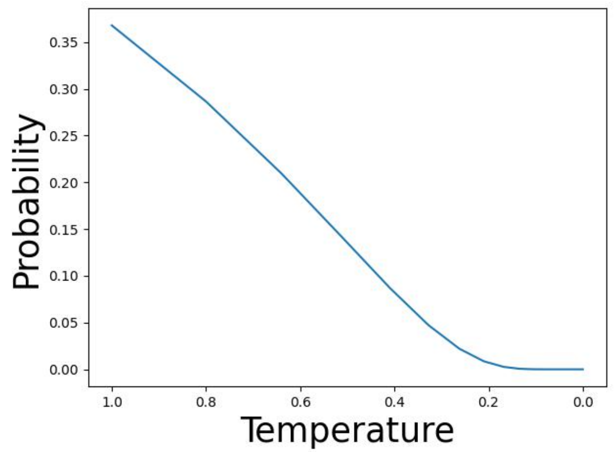
SA Algorithm

- Current, Next: nodes/states
- T: “temperature” controlling probability of downward steps
- Schedule: mapping from time to “temperature”
- H: heuristic evaluation function

SA Algorithm







SA Algorithm

current ← initial state

for t ← 1 to inf do

T ← schedule[t]

if T = 0 then return current

next ← **randomly selected** successor of current

$\Delta h \leftarrow h(\text{next}) - h(\text{current})$

if $\Delta h > 0$ then **current** ← **next** uphill

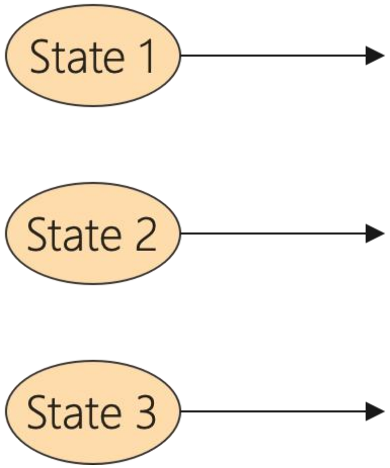
else **current** ← **next** only with probability $e^{\Delta h/T}$ downhill

SA Algorithm: Convergence

- If the schedule lowers T slowly enough, SA will find a global minimum with probability approaching 1
- In practice, reaching the global minimum could take a large number of iterations.

Local Beam Search

- Instead of maintaining one current state, local beam search keep track of k states
- All of the successors of the k states are generated
- Local beam search selects the best k states from all the successors and repeat until the goal state is found.



Local Beam Search

- Instead of maintaining one current state, local beam search keep track of k states
- Useful information is passed among the parellel search threads
- Disadvantage:
  - All k states can become stuck in a small region of the state space

Example: Satisfiability

- A wide variety of key CS problems can be translated into a propositional logical formalization

  e.g., $(A \vee B \vee C) \wedge (\neg B \vee C \vee D) \wedge (A \vee \neg C \vee D)$

- Solved by finding a truth assignment to the propositional variables $(A, B, C, ...)$ that makes it true, i.e., a model.

- If a formula has a model, we say that it is “satisfiable”

Random Walk SAT

Random walk SAT algorithm: $(A \vee B) \wedge (\neg B \vee C) \wedge (A \vee D)$

I Pick random truth assignment.

II Repeat until all clauses satisfied:

Flip variable from any unsatisfied clause.

Solve 2-SAT (2 variables per clause) in $O(n^2)$ flips

Does not work at all for hard k-SAT (k >= 3)

WalkSAT: Mixing Random Walk w/ Greedy Search (Selman et al. 1996)

- With probability p, walk,

  i.e., pick a variable in some unsatified clause and flip it;
- With probability (1-p) make a greedy flip,

  i.e., one that makes greatest decrease in number of unsatisfied clauses.

- Cannot detect unsatisfiability.

Experimental Results: Hard Random 3CNF

formula		GSAT						Simul. Ann.	
		basic		walk		noise			
vars	clauses	time	flips	time	flips	time	flips	time	flips
100	430	.4	7554	.2	2385	.6	9975	.6	4748
200	860	22	284693	4	27654	47	396534	21	106643
400	1700	122	2.6×10^6	7	59744	95	892048	75	552433
600	2550	1471	30×10^6	35	241651	929	7.8×10^6	427	2.7×10^6
800	3400	*	*	286	1.8×10^6	*	*	*	*
1000	4250	*	*	1095	5.8×10^6	*	*	*	*
2000	8480	*	*	3255	23×10^6	*	*	*	*

- Noise: different from WalkSAT, the selected randomly fipped variable is not restricted to be in an unsatified clause

Experimental Results: Hard Random 3CNF

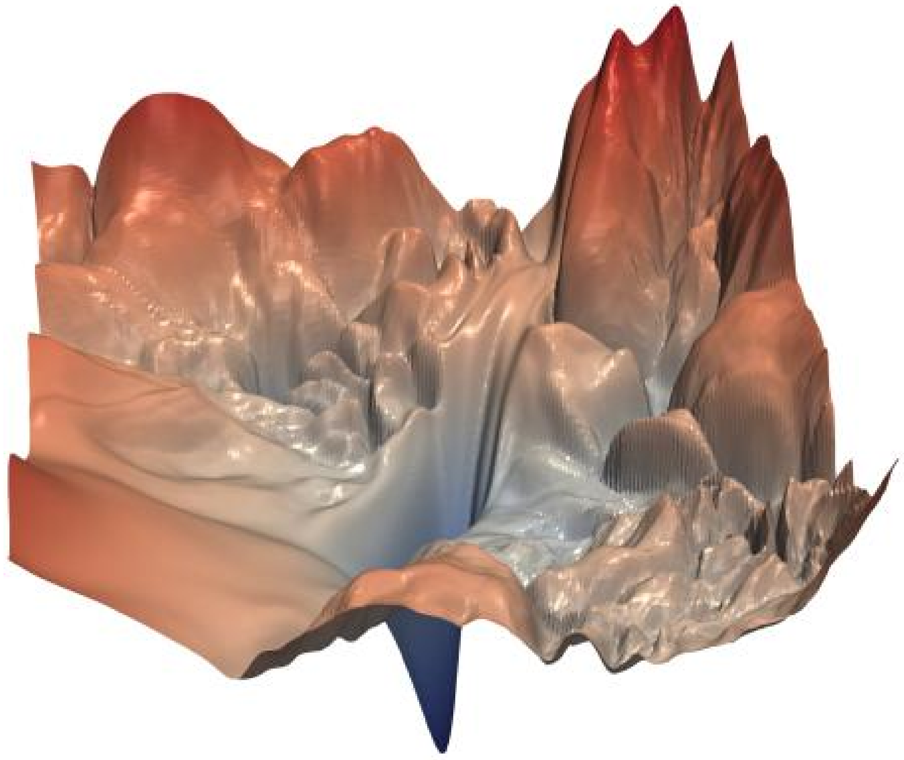
- Complete methods, such as DP, up to 400 variables

- WalkSAT better than
  Simulated Annealing better than
  Basic GSAT better than
  Backtracking (Davis-Putnam).

Local Search in Continous Spaces

- Originated in the 17th century, after the development of calculus by Newton and Leibniz

$$\min f(x_1, x_2, ..., x_n)$$



Local Search in Continous Spaces

- Originated in the 17th century, after the development of calculus by Newton and Leibniz

- Example:
  - Place a storage center (x, y) that are close to n cities (a_i, b_i)

$$f(x, y) = \sum_{i}^{n} \sqrt{(x - a_i)^2 + (y - b_i)^2}$$

Gradient Descent

- Similar to hill-climbing search but the states are continuous
- Basic idea:
  - Use the gradient of the cost function for updating the unknowns.

$$\nabla f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})$$

  - The gradient gives the direction for decreasing the objective function.

$$(x_{i+1}, y_{i+1}) = (x_i, y_i) - \alpha \nabla f$$

  - For small enough α, we have

$$f(x_{i+1}, y_{i+1}) \leq f(x_i, y_i)$$