CSCI 550: Advanced Algorithms
4. Local Search

- *gradient descent*
- *Metropolis algorithm*
- *Hopfield neural networks*
- *maximum cut*

Local search

Local search. Algorithm that explores the space of possible solutions in sequential fashion, moving from a current solution to a "nearby" one.

Neighbour relation. Let $S \sim S'$ be a neighbour relation for the problem.

Gradient descent. Let S denote the current solution. If there is a neighbour S' of S with a strictly lower cost, replace S with the neighbour whose cost is as small as possible. Otherwise, terminate the algorithm.



A funnel



a jagged funnel

Gradient descent: vertex cover

Vertex cover. Given a graph $G = (V, E)$, find a subset of nodes S of minimal cardinality such that for each $(u, v) \in E$, either u or v (or both) are in S.

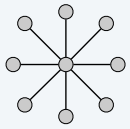
Neighbour relation. $S \sim S'$ if S' can be obtained from S by adding or deleting a single node. Each vertex cover S has at most n neighbours.

Gradient descent. Start with $S = V$. If there is a neighbour S' that is a vertex cover and has lower cardinality, replace S with S'.

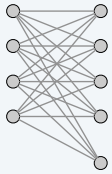
Remark. Algorithm terminates after at most n steps since each update decreases the size of the cover by one.

Gradient descent: vertex cover

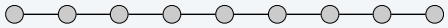
Local optimum. No neighbour is strictly better.



optimum = center node only
local optimum = all other nodes



optimum = all nodes on left side
local optimum = all nodes on right side



optimum = even nodes
local optimum = omit every third node

CSCI 550: Advanced Algorithms
4. Local Search

- *gradient descent*
- *Metropolis algorithm*
- *Hopfield neural networks*
- *maximum cut*

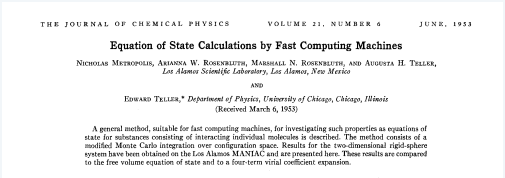
Metropolis algorithm

Key idea.

- Simulate the behaviour of a physical system according to principles of statistical mechanics.
- Globally biased toward "downhill" steps, but occasionally makes "uphill" steps to break out of local minima.



Nicholas Metropolis



THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, and Augusta H. Teller,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico
and
Edward Teller,* *Department of Physics, University of Chicago, Chicago, Illinois*
(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

Thermodynamics

Gibbs-Boltzmann function. The probability of finding a physical system in a state with energy E is proportional to

$$e^{-E/(kT)},$$

where $T > 0$ is temperature and k is the Boltzmann constant.

- For any temperature $T > 0$, the G-B function is a monotone decreasing function of energy E.
- System is more likely to be in a lower energy state than a higher one.
  - T large $\Rightarrow$ high- and low-energy states have roughly same probability.
  - T small $\Rightarrow$ low-energy states are much more probable.

Thermodynamics

Metropolis algorithm.

- Given a fixed temperature T, maintain the current state S.
- Randomly perturb the current state S to a new state $S' \in N(S)$.
- If $E(S') \leq E(S)$, update the current state to S'.
  Otherwise, update the current state to S' with probability $e^{-\Delta E/(kT)}$, where $\Delta E = E(S') - E(S) > 0$.

Theorem. Let $f_S(t)$ be the fraction of the first t steps in which the simulation is in state S. Then, assuming some technical conditions, with probability 1:

$$\lim_{t \to \infty} f_S(t) = \frac{1}{Z} e^{-E(S)/(kT)},$$

$$\text{where } Z = \sum_{S \in N(S)} e^{-E(S)/(kT)}.$$

Intuition. Simulation spends roughly the right amount of time in each state, according to the Gibbs-Boltzmann equation.

Simulated annealing

Consider how we might approximate the global optimum of some function.

Key idea.

- T large $\Rightarrow$ probability of accepting an uphill move is large.
- T small $\Rightarrow$ uphill moves are almost never accepted.
- Solution: turn a "knob" to control T.
- Cooling schedule: $T = T(i)$ at iteration i.

Physical analog: annealing.

- Take room-temp steel and heat it to a high temperature; we do not expect its molecules to maintain a nice crystal structure.
- Take glowing steel and cool it very abruptly; we do not expect its molecules to maintain a nice crystal structure.
- Anneal the steel by cooling it gradually from a high temperature, allowing it to reach equilibrium via a succession of intermediate lower temperatures.

Simulated annealing

Finding a local maximum.



CSCI 550: Advanced Algorithms
4. Local Search

- *gradient descent*
- *Metropolis algorithm*
- *Hopfield neural networks*
- *maximum cut*

Hopfield neural networks

Hopfield networks. Simple model of an associative memory, in which a large collection of units are connected by an underlying network, and neighbouring units try to correlate their states.

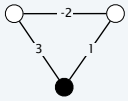


John Hopfield

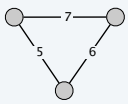
Input. Graph $G = (V, E)$ with integer (positive or negative) edge weights w.

Output. Node assignment (or configuration) $s_u = \pm 1$.

Intuition. If $w_{uv} < 0$, then u and v want to have the same state; if $w_{uv} > 0$, then u and v want different states.



Note. In general, no configuration respects all constraints.



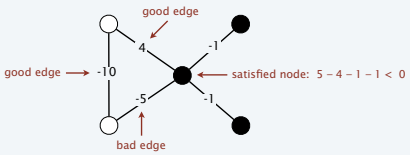
Hopfield neural networks

Def. With respect to a configuration S, an edge $e = (u, v)$ is good if $w_e \times s_u \times s_v < 0$. That is, if $w_e < 0$, then $s_u = s_v$; if $w_e > 0$, then $s_u \neq s_v$.

Def. With respect to a configuration S, a node u is satisfied if the weight of incident good edges ≥ the weight of incident bad edges.

$$\sum_{v:\, e=(u,v)\in E} w_e s_u s_v \;\leq\; 0$$

Def. A configuration is stable if all nodes are satisfied.



Goal. Find a stable configuration, if such a configuration exists.

State-flipping algorithm

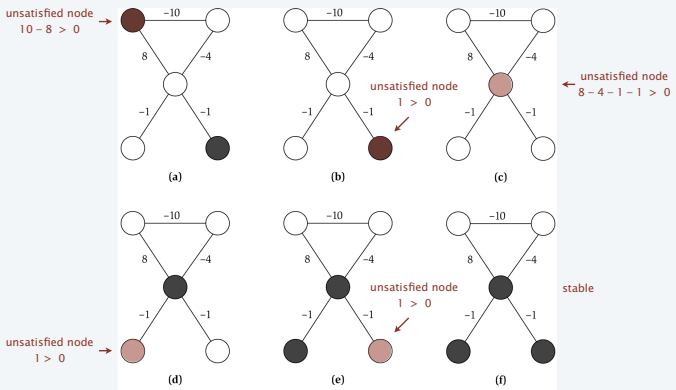
Goal. Find a stable configuration, if such a configuration exists.

Idea. Repeatedly flip the state of an unsatisfied node.

HOPFIELD-FLIP *(G, w)*

$S \leftarrow$ arbitrary configuration
WHILE (current configuration is not stable)
  $u \leftarrow$ unsatisfied node
  $s_u \leftarrow -s_u$
RETURN S

State-flipping algorithm: example



State-flipping algorithm: proof of correctness

Theorem. The state-flipping algorithm terminates with a stable configuration after at most $W = \Sigma_e |w_e|$ iterations.

Pf attempt.

Consider measure of progress $\Phi(S) = \#$ satisfied nodes.

State-flipping algorithm: proof of correctness

Theorem. The state-flipping algorithm terminates with a stable configuration after at most $W = \Sigma_e |w_e|$ iterations.

Pf.

Consider measure of progress $\Phi(S) = \Sigma_{e \text{ good}} |w_e|$.

- Clearly, $0 \le \Phi(S) \le W$.
- We show $\Phi(S)$ increases by at least 1 after each flip.

  When u flips state:
  - all good edges incident to u become bad
  - all bad edges incident to u become good
  - all other edges remain the same

$$\Phi(S') = \Phi(S) - \sum_{\substack{e:\ e=(u,v)\in E \\ e \text{ is bad}}} |w_e| + \sum_{\substack{e:\ e=(u,v)\in E \\ e \text{ is good}}} |w_e| \ge \Phi(S) + 1$$

u is unsatisfied

Complexity of Hopfield neural networks

Hopfield network search problem. Given a weighted graph, find a stable configuration if one exists.

Hopfield network decision problem. Given a weighted graph, does there exist a stable configuration?

Remark. The decision problem is trivially solvable (always yes), but there is no known poly-time algorithm for the search problem.

polynomial in n and $\log(W)$

CSCI 550: Advanced Algorithms
4. Local Search

- *gradient descent*
- *Metropolis algorithm*
- *Hopfield neural networks*
- *maximum cut*

Maximum cut

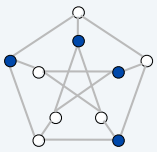
Maximum cut. Given an undirected graph $G = (V, E)$ with positive integer edge weights w_e, find a cut (A, B) such that the total weight of edges crossing the cut is maximized.

$$w(A, B) := \sum_{u \in A,\, v \in B} w_{uv}$$

Toy application.

- n activities, m people.
- Each person wants to participate in two of the activities.
- Schedule each activity in the morning or afternoon to maximize the number of people that can enjoy both activities.

Real applications. Circuit layout, statistical physics.



Maximum cut: local search

Single-flip neighbourhood. Given a cut (A, B), move one node from A to B (or one from B to A) if it improves the solution.

Greedy algorithm.

```
MAX-CUT-LOCAL (G, w)

(A, B) ← random cut
WHILE (there exists an improving node v)
  IF (v ∉ A)
    A ← A ∪ { v }
    B ← B − { v }
  ELSE
    B ← B ∪ { v }
    A ← A − { v }
RETURN (A, B)
```

Maximum cut: local search analysis

Theorem. Let (A, B) be a locally optimal cut and let (A^*, B^*) be an optimal cut. Then $w(A, B) \geq \frac{1}{2} \sum_e w_e \geq \frac{1}{2} w(A^*, B^*)$.

weights are nonnegative

Pf.

- Local optimality implies that for all $u \in A$: $\sum_{v \in A} w_{uv} \leq \sum_{v \in B} w_{uv}$

  Adding up all these inequalities yields:

$$2 \sum_{\{u,v\} \subseteq A} w_{uv} \leq \sum_{u \in A,\, v \in B} w_{uv} = w(A, B)$$

- Similarly, we have that $2 \sum_{\{u,v\} \subseteq B} w_{uv} \leq \sum_{u \in A,\, v \in B} w_{uv} = w(A, B)$

- Now,

each edge counted once

$$\sum_{e \in E} w_e = \underbrace{\sum_{\{u,v\} \subseteq A} w_{uv}}_{\leq \frac{1}{2} w(A, B)} + \underbrace{\sum_{u \in A,\, v \in B} w_{uv}}_{w(A, B)} + \underbrace{\sum_{\{u,v\} \subseteq A} w_{uv}}_{\leq \frac{1}{2} w(A, B)} \leq 2w(A, B) \quad \blacksquare$$

Maximum cut: big improvement flips

Local search. Within a factor of 2 for MAX-CUT, but not poly-time!

Big-improvement-flip algorithm. Only choose a node which, when flipped, increases the cut value by at least $\frac{2\varepsilon}{n} w(A, B)$.

Claim. Upon termination, big-improvement-flip algorithm returns a cut (A, B) such that $(2 + \varepsilon)\, w(A, B) \geq w(A^*, B^*)$.

Pf idea. Add $\frac{2\varepsilon}{n} w(A, B)$ to each inequality in the original proof. ▪

Claim. Big-improvement-flip algorithm terminates after $O(\varepsilon^{-1} n \log(W))$ flips, where $W = \sum_e w_e$.

Pf idea.

- Each flip improves the cut value by at least a factor of $(1 + \varepsilon / n)$.
- After n / ε iterations, the cut value improves by a factor of 2.
- Cut value can be doubled at most $\log_2(W)$ times. ▪

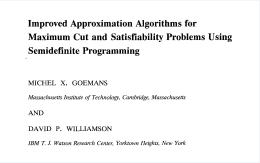
if $x \geq 1$, $(1 + 1/x)^x \geq 2$

Maximum cut: context

Theorem. [Sahni-Gonzales, 1976] There exists a ½-approximation algorithm for MAX-CUT.

Theorem. [Goemans-Williamson, 1995] There exists a 0.878-approximation algorithm for MAX-CUT.

Theorem. [Håstad, 2001] Unless **P** = **NP**, there does not exist any 0.942-approximation algorithm for MAX-CUT.



Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming

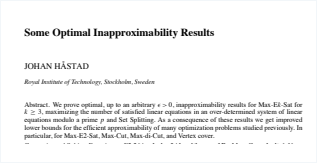
MICHEL X. GOEMANS

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

DAVID P. WILLIAMSON

IBM T. J. Watson Research Center, Yorktown Heights, New York



Some Optimal Inapproximability Results

JOHAN HÅSTAD

Royal Institute of Technology, Stockholm, Sweden

Abstract. We prove optimal, up to an arbitrary $\epsilon > 0$, inapproximability results for Max-Ek-Sat for $k \geq 3$, maximizing the number of satisfied linear equations in an over-determined system of linear equations modulo a prime p and Set Splitting. As a consequence of these results we get improved lower bounds for the efficient approximability of many optimization problems studied previously. In particular, for Max-E2-Sat, Max-Cut, Max-di-Cut, and Vertex cover.