Kapitel 5:
Local Search

Inhalt:

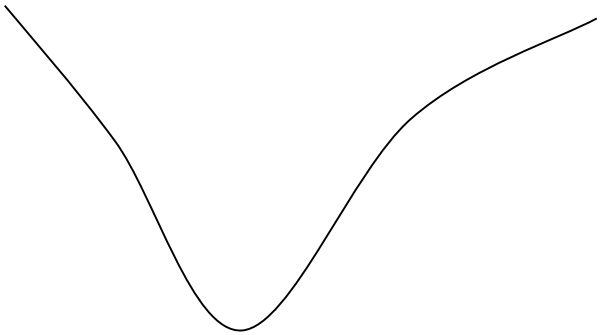
- Gradient Descent (Hill Climbing)
- Metropolis Algorithm and Simulated Annealing
- Local Search in Hopfield Neural Networks
- Local Search for Max-Cut
  - Single-flip neighborhood
  - K-flip neighborhood
  - KL-neighborhood
- Nash Equilibria

Local Search

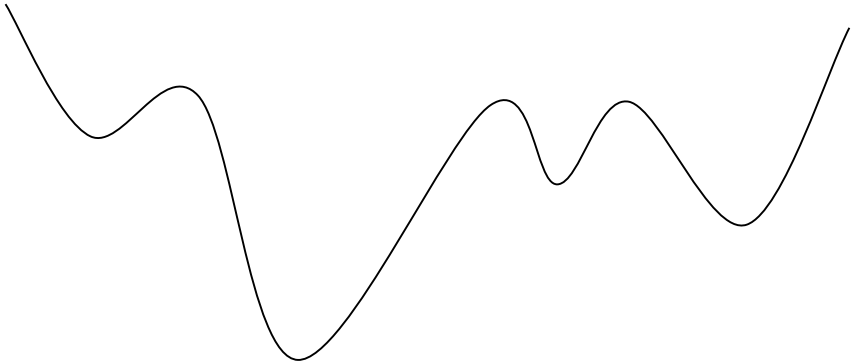
Local search. Algorithm that explores the space of possible solutions in sequential fashion, moving from a current solution to a "nearby" one.

1) Neighbor relation. Let $S \sim S'$ be a neighbor relation for the problem.
2) Choice Rule. Rule for choosing a neighboring solution at each step.

Gradient descent. Let S denote current solution. If there is a neighbor S' of S with strictly lower cost, $C(S') < C(S)$, replace S with the neighbor 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.

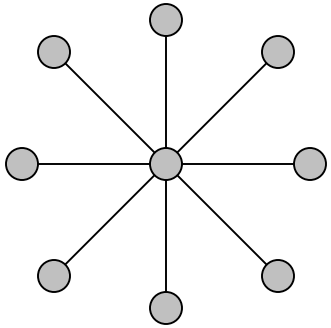
Neighbor relation. $S \sim S'$ if S' can be obtained from S by adding/deleting a single node to/from the cover. Each vertex cover S has at most n neighbors.

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

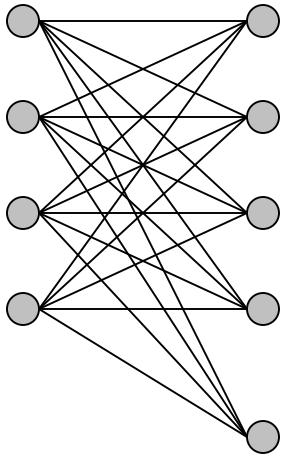
Remark. Algorithm terminates after at most n steps since each update decreases the size of the cover by one.

Gradient Descent: Vertex Cover

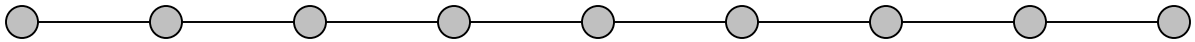
Local optimum. No neighbor is strictly better.



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

Kapitel 5:
Local Search

Inhalt:

- Gradient Descent (Hill Climbing)
- Metropolis Algorithm and Simulated Annealing
- Local Search in Hopfield Neural Networks
- Local Search for Max-Cut
  - Single-flip neighborhood
  - K-flip neighborhood
  - KL-neighborhood
- Nash Equilibria

Metropolis Algorithm

Metropolis algorithm. [Metropolis, Rosenbluth, Rosenbluth, Teller, Teller 1953]

- Simulate behavior 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.

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 a constant.

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

Metropolis Algorithm

Metropolis algorithm.

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

Theorem. Let $f_S(t)$ be fraction of first t steps in which 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 Gibbs-Boltzmann equation.

Simulated Annealing

Simulated annealing.

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

Physical analog.

- Take solid and raise it to high temperature, we do not expect it to maintain a nice crystal structure.
- Take a molten solid and freeze it very abruptly, we do not expect to get a perfect crystal either.
- Annealing: cool material gradually from high temperature, allowing it to reach equilibrium at succession of intermediate lower temperatures.

Kapitel 5:
Local Search

Inhalt:

- Gradient Descent (Hill Climbing)
- Metropolis Algorithm and Simulated Annealing
- Local Search in Hopfield Neural Networks
- Local Search for Max-Cut
  - Single-flip neighborhood
  - K-flip neighborhood
  - KL-neighborhood
- Nash Equilibria

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 neighboring units try to correlate their states.

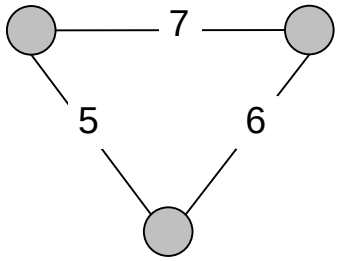
Input: Graph $G = (V, E)$ with integer edge weights w.

positive or negative

Configuration. Node assignment $s_u = \pm 1$ for all $u \in V$.

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.



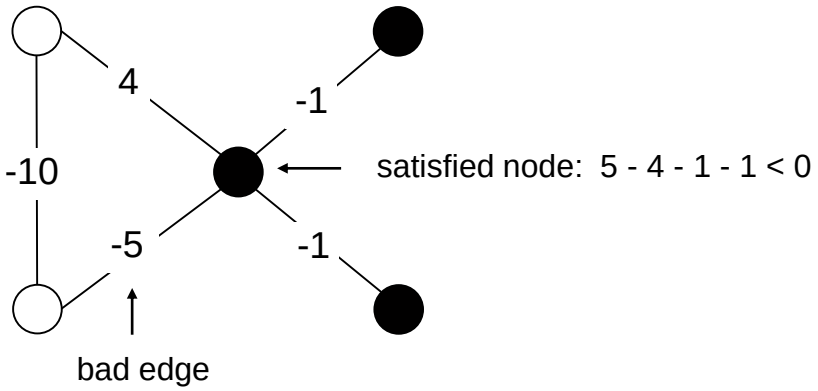
Hopfield Neural Networks

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

Def. With respect to a configuration S, a node u is satisfied if the total absolute weight of incident good edges $\geq$ total absolute 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.

Hopfield Neural Networks

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

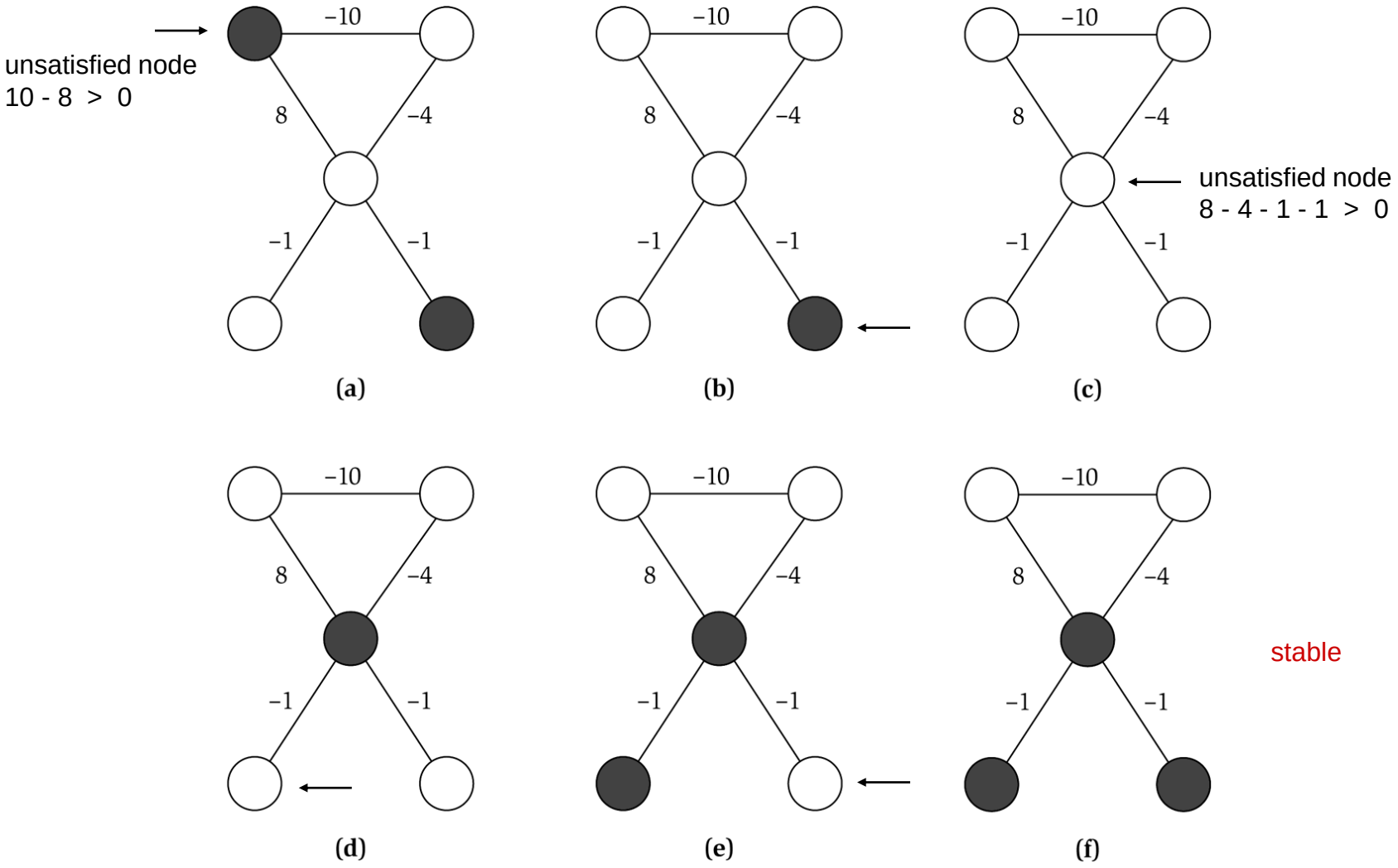
State-flipping algorithm. Repeated flip state of an unsatisfied node.

```
Hopfield-Flip(G, w) {
    S ← arbitrary configuration

    while (current configuration is not stable) {
        u ← unsatisfied node
        s_u = -s_u
    }

    return S
}
```

State Flipping Algorithm



Hopfield Neural Networks

Claim. 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.

Hopfield Neural Networks

Claim. 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 $0 \leq \Phi(S) \leq W$.
- 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

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

u is unsatisfied

Fragen?