

# Tópicos de Análise de Algoritmos

Busca local:  
algoritmo Metropolis e simulated annealing

Secs 12.1 e 12.2 do KT

# Busca local

Problema de otimização combinatória:

conjunto de **instâncias**

para cada instância  $I$ ,

um conjunto  $\text{Sol}(I)$  (não vazio) de soluções (viáveis) e

uma função  $\text{val}(S)$  para cada solução  $S$  em  $\text{Sol}(I)$

# Busca local

Problema de otimização combinatória:

conjunto de instâncias

para cada instância  $I$ ,

um conjunto  $Sol(I)$  (não vazio) de soluções (viáveis) e

uma função  $val(S)$  para cada solução  $S$  em  $Sol(I)$

**Objetivo:** Dada uma instância  $I$  viável,  
encontrar solução  $S$  em  $Sol(I)$  tq  $val(S)$  é mínimo/máximo.

# Busca local

Problema de otimização combinatória:

conjunto de instâncias

para cada instância  $I$ ,

um conjunto  $Sol(I)$  (não vazio) de soluções (viáveis) e

uma função  $val(S)$  para cada solução  $S$  em  $Sol(I)$

**Objetivo:** Dada uma instância  $I$  viável,  
encontrar solução  $S$  em  $Sol(I)$  tq  $val(S)$  é mínimo/máximo.

Tipicamente  $Sol(I)$  tem tamanho exponencial.

# Busca local

Problema de otimização combinatória:

conjunto de **instâncias**

para cada instância  $I$ ,

um conjunto  $\text{Sol}(I)$  (não vazio) de soluções (viáveis) e

uma função  $\text{val}(S)$  para cada solução  $S$  em  $\text{Sol}(I)$

**Objetivo:** Dada uma instância  $I$  viável,  
encontrar solução  $S$  em  $\text{Sol}(I)$  tq  $\text{val}(S)$  é mínimo/máximo.

Tipicamente  $\text{Sol}(I)$  tem tamanho exponencial.

Em uma busca local, adicionamos a  
noção de **vizinhança** entre as soluções.

# Busca local

Problema de otimização combinatória:

conjunto de **instâncias**

para cada instância  $I$ ,

um conjunto  $\text{Sol}(I)$  (não vazio) de soluções (viáveis) e

uma função  $\text{val}(S)$  para cada solução  $S$  em  $\text{Sol}(I)$

**Objetivo:** Dada uma instância  $I$  viável,  
encontrar solução  $S$  em  $\text{Sol}(I)$  tq  $\text{val}(S)$  é mínimo/máximo.

Tipicamente  $\text{Sol}(I)$  tem tamanho exponencial.

Em uma busca local, adicionamos a  
noção de **vizinhança** entre as soluções.

Relações que vizinhança **simétricas**.

# Busca local

## Algoritmo:

Começa com uma solução arbitrária  $S$ .

Iterativamente muda para uma solução  $S'$  vizinha a  $S$ .

# Busca local

## Algoritmo:

Começa com uma solução arbitrária  $S$ .

Iterativamente muda para uma solução  $S'$  vizinha a  $S$ .

Durante o processo, guarda a melhor solução visitada.

# Busca local

## Algoritmo:

Começa com uma solução arbitrária  $S$ .

Iterativamente muda para uma solução  $S'$  vizinha a  $S$ .

Durante o processo, guarda a melhor solução visitada.

## Pontos críticos:

- ▶ como definir a vizinhança de uma solução

# Busca local

## Algoritmo:

Começa com uma solução arbitrária  $S$ .

Iterativamente muda para uma solução  $S'$  vizinha a  $S$ .

Durante o processo, guarda a melhor solução visitada.

## Pontos críticos:

- ▶ como definir a vizinhança de uma solução
- ▶ como escolher  $S'$  na vizinhança de  $S$

# Busca local

## Algoritmo:

Começa com uma solução arbitrária  $S$ .

Iterativamente muda para uma solução  $S'$  vizinha a  $S$ .

Durante o processo, guarda a melhor solução visitada.

## Pontos críticos:

- ▶ como definir a vizinhança de uma solução
- ▶ como escolher  $S'$  na vizinhança de  $S$

Grafo cujo conjunto de vértices é  $\text{Sol}(I)$  e adjacência é dada pela relação de vizinhança entre as soluções.

Algoritmo de busca local é um passeio neste grafo, que tenta alcançar uma boa solução.

## Cobertura por vértices

Grafo  $G = (V, E)$

$S \subseteq V$  é **cobertura por vértices** se

toda aresta  $e$  em  $E$  tem pelo menos uma ponta em  $S$ .

# Cobertura por vértices

Grafo  $G = (V, E)$

$S \subseteq V$  é **cobertura por vértices** se  
toda aresta  $e$  em  $E$  tem pelo menos uma ponta em  $S$ .

**Problema:** Dado  $G$ ,  
encontrar cobertura por vértices de tamanho mínimo.

custo  $c(S) = |S|$

# Cobertura por vértices

Grafo  $G = (V, E)$

$S \subseteq V$  é **cobertura por vértices** se  
toda aresta  $e$  em  $E$  tem pelo menos uma ponta em  $S$ .

**Problema:** Dado  $G$ ,  
encontrar cobertura por vértices de tamanho mínimo.

custo  $c(S) = |S|$

**Relação de vizinhança:**

$S \sim S'$  se  $S'$  é obtido de  $S$   
adicionando-se ou removendo-se um vértice.

# Cobertura por vértices

Grafo  $G = (V, E)$

$S \subseteq V$  é **cobertura por vértices** se  
toda aresta  $e$  em  $E$  tem pelo menos uma ponta em  $S$ .

**Problema:** Dado  $G$ ,  
encontrar cobertura por vértices de tamanho mínimo.

custo  $c(S) = |S|$

**Relação de vizinhança:**

$S \sim S'$  se  $S'$  é obtido de  $S$   
adicionando-se ou removendo-se um vértice.

Cada solução  $S$  tem  $n$  vizinhos, onde  $n = |V|$ .

# Algoritmo de busca local

## Gradiente descendente

**BUSCA-LOCAL** ( $n, E$ )     $\triangleright G = (V, E)$  onde  $V = [n]$

- 1     $S \leftarrow [n]$
- 2    **enquanto** existe vizinho  $S'$  de  $S$   
          tq  $|S'| < |S|$  e  $S'$  é cobertura **faça**
- 3         $S \leftarrow S'$
- 4    **devolva**  $S$

# Algoritmo de busca local

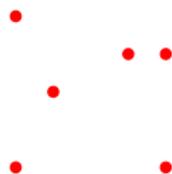
## Gradiente descendente

**BUSCA-LOCAL** ( $n, E$ )     $\triangleright G = (V, E)$  onde  $V = [n]$

- 1     $S \leftarrow [n]$
- 2    **enquanto** existe vizinho  $S'$  de  $S$   
      tq  $|S'| < |S|$  e  $S'$  é cobertura **faça**
- 3         $S \leftarrow S'$
- 4    **devolva**  $S$

**Exemplo 1:**  $G = (V, \emptyset)$

$S$  decresce um vértice de cada vez até  $S = \emptyset$ , que é ótimo.



# Algoritmo de busca local

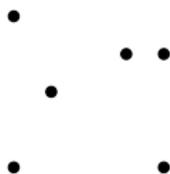
## Gradiente descendente

**BUSCA-LOCAL** ( $n, E$ )     $\triangleright G = (V, E)$  onde  $V = [n]$

- 1     $S \leftarrow [n]$
- 2    **enquanto** existe vizinho  $S'$  de  $S$   
          tq  $|S'| < |S|$  e  $S'$  é cobertura **faça**
- 3         $S \leftarrow S'$
- 4    **devolva**  $S$

**Exemplo 1:**  $G = (V, \emptyset)$

$S$  decresce um vértice de cada vez até  $S = \emptyset$ , que é ótimo.



# Algoritmo de busca local

## Gradiente descendente

**BUSCA-LOCAL** ( $n, E$ )     $\triangleright G = (V, E)$  onde  $V = [n]$

- 1     $S \leftarrow [n]$
- 2    **enquanto** existe vizinho  $S'$  de  $S$   
          tq  $|S'| < |S|$  e  $S'$  é cobertura **faça**
- 3         $S \leftarrow S'$
- 4    **devolva**  $S$

**Exemplo 2:**  $G$  é estrela de centro  $v$



# Algoritmo de busca local

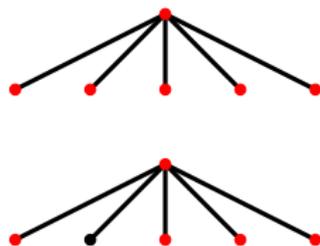
## Gradiente descendente

**BUSCA-LOCAL** ( $n, E$ )     $\triangleright G = (V, E)$  onde  $V = [n]$

- 1     $S \leftarrow [n]$
- 2    **enquanto** existe vizinho  $S'$  de  $S$   
          tq  $|S'| < |S|$  e  $S'$  é cobertura **faça**
- 3         $S \leftarrow S'$
- 4    **devolva**  $S$

**Exemplo 2:**  $G$  é estrela de centro  $v$

Se, ao final da primeira iteração,  $S \neq S \setminus \{v\}$ ,  
o algoritmo termina com  $S = \{v\}$ , que é ótimo.



# Algoritmo de busca local

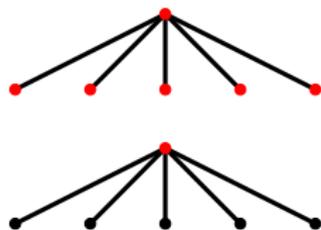
## Gradiente descendente

**BUSCA-LOCAL** ( $n, E$ )     $\triangleright G = (V, E)$  onde  $V = [n]$

- 1     $S \leftarrow [n]$
- 2    **enquanto** existe vizinho  $S'$  de  $S$   
          tq  $|S'| < |S|$  e  $S'$  é cobertura **faça**
- 3         $S \leftarrow S'$
- 4    **devolva**  $S$

**Exemplo 2:**  $G$  é estrela de centro  $v$

Se, ao final da primeira iteração,  $S \neq S \setminus \{v\}$ ,  
o algoritmo termina com  $S = \{v\}$ , que é ótimo.



# Algoritmo de busca local

## Gradiente descendente

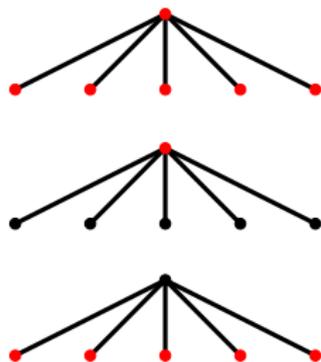
**BUSCA-LOCAL** ( $n, E$ )     $\triangleright G = (V, E)$  onde  $V = [n]$

- 1     $S \leftarrow [n]$
- 2    **enquanto** existe vizinho  $S'$  de  $S$   
          tq  $|S'| < |S|$  e  $S'$  é cobertura **faça**
- 3         $S \leftarrow S'$
- 4    **devolva**  $S$

**Exemplo 2:**  $G$  é estrela de centro  $v$

Se, ao final da primeira iteração,  $S \neq S \setminus \{v\}$ ,  
o algoritmo termina com  $S = \{v\}$ , que é ótimo.

Se, ao final da primeira iteração,  $S = S \setminus \{v\}$ ,  
o algoritmo termina com esse ótimo local.



# Algoritmo de busca local

## Gradiente descendente

**BUSCA-LOCAL** ( $n, E$ )     $\triangleright G = (V, E)$  onde  $V = [n]$

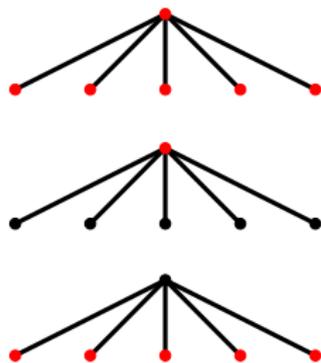
- 1     $S \leftarrow [n]$
- 2    **enquanto** existe vizinho  $S'$  de  $S$   
          tq  $|S'| < |S|$  e  $S'$  é cobertura **faça**
- 3         $S \leftarrow S'$
- 4    **devolva**  $S$

**Exemplo 2:**  $G$  é estrela de centro  $v$

Se, ao final da primeira iteração,  $S \neq S \setminus \{v\}$ ,  
o algoritmo termina com  $S = \{v\}$ , que é ótimo.

Se, ao final da primeira iteração,  $S = S \setminus \{v\}$ ,  
o algoritmo termina com esse ótimo local.

O algoritmo pode se dar arbitrariamente mal.



# Algoritmo de busca local

## Gradiente descendente

**BUSCA-LOCAL** ( $n, E$ )     $\triangleright G = (V, E)$  onde  $V = [n]$

- 1     $S \leftarrow [n]$
- 2    **enquanto** existe vizinho  $S'$  de  $S$   
          tq  $|S'| < |S|$  e  $S'$  é cobertura **faça**
- 3         $S \leftarrow S'$
- 4    **devolva**  $S$

**Exemplo 3:**  $G$  é um caminho com número ímpar de vértices



# Algoritmo de busca local

## Gradiente descendente

**BUSCA-LOCAL** ( $n, E$ )     $\triangleright G = (V, E)$  onde  $V = [n]$

- 1     $S \leftarrow [n]$
- 2    **enquanto** existe vizinho  $S'$  de  $S$   
          tq  $|S'| < |S|$  e  $S'$  é cobertura **faça**
- 3         $S \leftarrow S'$
- 4    **devolva**  $S$

**Exemplo 3:**  $G$  é um caminho com número ímpar de vértices



**Solução ótima:** conjunto de vértices pares

# Algoritmo de busca local

## Gradiente descendente

**BUSCA-LOCAL** ( $n, E$ )     $\triangleright G = (V, E)$  onde  $V = [n]$

- 1     $S \leftarrow [n]$
- 2    **enquanto** existe vizinho  $S'$  de  $S$   
          tq  $|S'| < |S|$  e  $S'$  é cobertura **faça**
- 3         $S \leftarrow S'$
- 4    **devolva**  $S$

**Exemplo 3:**  $G$  é um caminho com número ímpar de vértices



**Solução ótima:** conjunto de vértices pares

Muitos mínimos locais.

# Algoritmo Metropolis

Metropolis, Rosenbluth, Teller, e Teller (1953)

Simulação de sistema físico  
de acordo com mecânica estatística.

# Algoritmo Metropolis

Metropolis, Rosenbluth, Teller, e Teller (1953)

Simulação de sistema físico  
de acordo com mecânica estatística.

Função de Gibbs-Boltzmann:  $e^{-E/(kT)}$ ,  
onde  $E$  é a energia,  $T$  é a temperatura e  $k$  é uma constante.

# Algoritmo Metropolis

Metropolis, Rosenbluth, Teller, e Teller (1953)

Simulação de sistema físico  
de acordo com mecânica estatística.

Função de Gibbs-Boltzmann:  $e^{-E/(kT)}$ ,  
onde  $E$  é a energia,  $T$  é a temperatura e  $k$  é uma constante.

Modelo: o sistema está num estado de energia  $E$   
com probabilidade dada pela função de Gibbs-Boltzmann.

# Algoritmo Metropolis

Metropolis, Rosenbluth, Teller, e Teller (1953)

Simulação de sistema físico  
de acordo com mecânica estatística.

Função de Gibbs-Boltzmann:  $e^{-E/(kT)}$ ,  
onde  $E$  é a energia,  $T$  é a temperatura e  $k$  é uma constante.

**Modelo:** o sistema está num estado de energia  $E$   
com probabilidade dada pela função de Gibbs-Boltzmann.

Para  $T$  fixo, a função decresce com  $E$ .

(Maior a chance de estar em um estado de menor energia.)

# Algoritmo Metropolis

Metropolis, Rosenbluth, Teller, e Teller (1953)

Simulação de sistema físico  
de acordo com mecânica estatística.

Função de Gibbs-Boltzmann:  $e^{-E/(kT)}$ ,  
onde  $E$  é a energia,  $T$  é a temperatura e  $k$  é uma constante.

Modelo: o sistema está num estado de energia  $E$   
com probabilidade dada pela função de Gibbs-Boltzmann.

Para  $T$  fixo, a função decresce com  $E$ .

(Maior a chance de estar em um estado de menor energia.)

Para  $T$  pequeno,  
estado de menor energia é mais provável que estado de energia alta.

# Algoritmo Metropolis

## Metropolis, Rosenbluth, Teller, e Teller (1953)

Simulação de sistema físico  
de acordo com mecânica estatística.

**Função de Gibbs-Boltzmann:**  $e^{-E/(kT)}$ ,  
onde  $E$  é a energia,  $T$  é a temperatura e  $k$  é uma constante.

**Modelo:** o sistema está num estado de energia  $E$   
com probabilidade dada pela função de Gibbs-Boltzmann.

Para  $T$  fixo, a função decresce com  $E$ .

(Maior a chance de estar em um estado de menor energia.)

Para  $T$  pequeno,  
estado de menor energia é mais provável que estado de energia alta.

Para  $T$  grande, a diferença entre estas probabilidades é bem pequena.

# Algoritmo Metropolis

**Objetivo:** chegar a um estado de energia mínima.

# Algoritmo Metropolis

**Objetivo:** chegar a um estado de energia mínima.

**Algoritmo Metropolis:**      ▷ para um dado  $T$

Comece em um estado  $S$ .

A cada iteração, gere uma perturbação pequena  $S'$  de  $S$ .

Se  $E(S') \leq E(S)$ , mude para  $S'$ .

Senão seja  $\Delta(E) = E(S') - E(S)$ .

Mude para  $S'$  com probabilidade  $e^{-\Delta(E)/(kT)}$ .

# Algoritmo Metropolis

**Objetivo:** chegar a um estado de energia mínima.

**Algoritmo Metropolis:**  $\triangleright$  para um dado  $T$

Comece em um estado  $S$ .

A cada iteração, gere uma perturbação pequena  $S'$  de  $S$ .

Se  $E(S') \leq E(S)$ , mude para  $S'$ .

Senão seja  $\Delta(E) = E(S') - E(S)$ .

Mude para  $S'$  com probabilidade  $e^{-\Delta(E)/(kT)}$ .

Seja  $Z = \sum_S e^{-E(S)/(kT)}$ .

Para um estado  $S$ , seja  $f_S(t)$  a fração dos  $t$  primeiros passos da simulação em que o algoritmo passa em  $S$ .

$\lim_{t \rightarrow \infty} f_S(t) = \frac{1}{Z} \cdot e^{-E(S)/(kT)}$  com probabilidade indo para 1

# Algoritmo Metropolis

**Objetivo:** chegar a um estado de energia mínima.

**Algoritmo Metropolis:**  $\triangleright$  para um dado  $T$

Comece em um estado  $S$ .

A cada iteração, gere uma perturbação pequena  $S'$  de  $S$ .

Se  $E(S') \leq E(S)$ , mude para  $S'$ .

Senão seja  $\Delta(E) = E(S') - E(S)$ .

Mude para  $S'$  com probabilidade  $e^{-\Delta(E)/(kT)}$ .

Seja  $Z = \sum_S e^{-E(S)/(kT)}$ .

Para um estado  $S$ , seja  $f_S(t)$  a fração dos  $t$  primeiros passos da simulação em que o algoritmo passa em  $S$ .

$\lim_{t \rightarrow \infty} f_S(t) = \frac{1}{Z} \cdot e^{-E(S)/(kT)}$  com probabilidade indo para 1

**Fica em  $S$  o tempo esperado.**

# Metropolis para cobertura por vértices

Exemplo 1:  $G$  é estrela de centro  $v$

Se, ao final da primeira iteração,  $S = S \setminus \{v\} \dots$



# Metropolis para cobertura por vértices

Exemplo 1:  $G$  é estrela de centro  $v$

Se, ao final da primeira iteração,  $S = S \setminus \{v\}$ ...



com probabilidade positiva coloca  $v$  de volta no  $S$   
e, mais adiante, encontra a cobertura mínima!

# Metropolis para cobertura por vértices

Exemplo 1:  $G$  é estrela de centro  $v$

Se, ao final da primeira iteração,  $S = S \setminus \{v\}$ ...



com probabilidade positiva coloca  $v$  de volta no  $S$   
e, mais adiante, encontra a cobertura mínima!

Exemplo 2:  $G = (V, \emptyset)$

Quando  $S$  está grande,  
tem boas chances do  $S$  diminuir.



# Metropolis para cobertura por vértices

Exemplo 1:  $G$  é estrela de centro  $v$

Se, ao final da primeira iteração,  $S = S \setminus \{v\}$ ...



com probabilidade positiva coloca  $v$  de volta no  $S$   
e, mais adiante, encontra a cobertura mínima!

Exemplo 2:  $G = (V, \emptyset)$

Quando  $S$  está grande,  
tem boas chances do  $S$  diminuir.



Mas quando  $S$  fica pequeno,  
tem boas chances de voltar a crescer  
e é difícil chegar em  $S = \emptyset$ .



# Algoritmo Metropolis

**Algoritmo Metropolis:**      ▷ para um dado  $T$

Comece em um estado  $S$ .

A cada iteração, gere uma perturbação pequena  $S'$  de  $S$ .

Se  $E(S') \leq E(S)$ , mude para  $S'$ .

Senão seja  $\Delta(E) = E(S') - E(S)$ .

Mude para  $S'$  com probabilidade  $e^{-\Delta(E)/(kT)}$ .

Mas e a temperatura  $T$ ?

Qual é o papel e o efeito da temperatura?

# Simulated Annealing

De volta ao sistema físico.

## Annealing:

processo de cristalização em que o material é resfriado lentamente, para atingir o estado de energia mínima.

# Simulated Annealing

De volta ao sistema físico.

## Annealing:

processo de cristalização em que o material é resfriado lentamente, para atingir o estado de energia mínima.

Material em altas temperaturas não cristaliza.

# Simulated Annealing

De volta ao sistema físico.

## Annealing:

processo de cristalização em que o material é resfriado lentamente, para atingir o estado de energia mínima.

Material em altas temperaturas não cristaliza.

Se esfria rapidamente, cristaliza com muitas imperfeições.

# Simulated Annealing

De volta ao sistema físico.

## Annealing:

processo de cristalização em que o material é resfriado lentamente, para atingir o estado de energia mínima.

Material em altas temperaturas não cristaliza.

Se esfria rapidamente, cristaliza com muitas imperfeições.

**Simulated annealing** imita este processo, ajustando as probabilidades de mudar para um estado de energia maior de acordo com um parâmetro  $T$  que diminui com o tempo.

# Simulated Annealing

De volta ao sistema físico.

## Annealing:

processo de cristalização em que o material é resfriado lentamente, para atingir o estado de energia mínima.

Material em altas temperaturas não cristaliza.

Se esfria rapidamente, cristaliza com muitas imperfeições.

**Simulated annealing** imita este processo, ajustando as probabilidades de mudar para um estado de energia maior de acordo com um parâmetro  $T$  que diminui com o tempo.

Em geral, não tem garantia de qualidade.

# Simulated Annealing

De volta ao sistema físico.

## Annealing:

processo de cristalização em que o material é resfriado lentamente, para atingir o estado de energia mínima.

Material em altas temperaturas não cristaliza.

Se esfria rapidamente, cristaliza com muitas imperfeições.

**Simulated annealing** imita este processo, ajustando as probabilidades de mudar para um estado de energia maior de acordo com um parâmetro  $T$  que diminui com o tempo.

Em geral, não tem garantia de qualidade.

Em que velocidade diminuir o  $T$ ?

# Redes neurais de Hopfield

Às vezes buscamos um ótimo local, não necessariamente global.

# Redes neurais de Hopfield

Às vezes buscamos um ótimo local, não necessariamente global.

Rede neural de Hopfield:

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

# Redes neurais de Hopfield

Às vezes buscamos um ótimo local, não necessariamente global.

Rede neural de Hopfield:

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

Configuração da rede: atribuição de um sinal  $+$  ou  $-$  para cada nó

# Redes neurais de Hopfield

Às vezes buscamos um ótimo local, não necessariamente global.

**Rede neural de Hopfield:**

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

**Configuração da rede:** atribuição de um sinal  $+$  ou  $-$  para cada nó

O sinal que um nó escolhe é influenciado pelo sinal de seus vizinhos.

# Redes neurais de Hopfield

Às vezes buscamos um ótimo local, não necessariamente global.

**Rede neural de Hopfield:**

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

**Configuração da rede:** atribuição de um sinal  $+$  ou  $-$  para cada nó

O sinal que um nó escolhe é influenciado pelo sinal de seus vizinhos.

Cada aresta  $e$  representa um requisito:

se  $e$  tem peso positivo, os nós querem sinais opostos

se  $e$  tem peso negativo, os nós querem ter o mesmo sinal

# Redes neurais de Hopfield

Às vezes buscamos um ótimo local, não necessariamente global.

**Rede neural de Hopfield:**

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

**Configuração da rede:** atribuição de um sinal  $+$  ou  $-$  para cada nó

O sinal que um nó escolhe é influenciado pelo sinal de seus vizinhos.

Cada aresta  $e$  representa um requisito:

se  $e$  tem peso positivo, os nós querem sinais opostos

se  $e$  tem peso negativo, os nós querem ter o mesmo sinal

$|w_e|$  representa a intensidade da dependência entre os extremos de  $e$ .

# Redes neurais de Hopfield

Às vezes buscamos um ótimo local, não necessariamente global.

**Rede neural de Hopfield:**

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

**Configuração da rede:** atribuição de um sinal  $+$  ou  $-$  para cada nó

O sinal que um nó escolhe é influenciado pelo sinal de seus vizinhos.

Cada aresta  $e$  representa um requisito:

se  $e$  tem peso positivo, os nós querem sinais opostos

se  $e$  tem peso negativo, os nós querem ter o mesmo sinal

$|w_e|$  representa a intensidade da dependência entre os extremos de  $e$ .

Pode não existir configuração que satisfaça todos os requisitos.

Pense num triângulo com as três arestas com peso 1.

# Redes neurais de Hopfield

Rede neural de Hopfield:

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

Configuração da rede: atribuição de um sinal  $+$  ou  $-$  para cada nó

# Redes neurais de Hopfield

Rede neural de Hopfield:

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

Configuração da rede: atribuição de um sinal  $+$  ou  $-$  para cada nó

Aresta  $e = uv$  é boa se  $u$  e  $v$  respeitam seu requisito:

se  $w_e < 0$  então  $s_u = s_v$  e se  $w_e > 0$  então  $s_u \neq s_v$

Senão  $e$  é ruim.

# Redes neurais de Hopfield

Rede neural de Hopfield:

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

Configuração da rede: atribuição de um sinal  $+$  ou  $-$  para cada nó

Aresta  $e = uv$  é boa se  $u$  e  $v$  respeitam seu requisito:

se  $w_e < 0$  então  $s_u = s_v$  e se  $w_e > 0$  então  $s_u \neq s_v$

Senão  $e$  é ruim.

Em outras palavras,  $e$  é boa sse  $w_e s_u s_v < 0$ .

# Redes neurais de Hopfield

Rede neural de Hopfield:

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

Configuração da rede: atribuição de um sinal  $+$  ou  $-$  para cada nó

Aresta  $e = uv$  é boa se  $u$  e  $v$  respeitam seu requisito:

se  $w_e < 0$  então  $s_u = s_v$  e se  $w_e > 0$  então  $s_u \neq s_v$

Senão  $e$  é ruim.

Em outras palavras,  $e$  é boa sse  $w_e s_u s_v < 0$ .

Nó  $u$  está satisfeito se  $\sum_e \text{boa} |w_e| > \sum_e \text{ruim} |w_e|$

# Redes neurais de Hopfield

Rede neural de Hopfield:

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

Configuração da rede: atribuição de um sinal  $+$  ou  $-$  para cada nó

Aresta  $e = uv$  é boa se  $u$  e  $v$  respeitam seu requisito:

se  $w_e < 0$  então  $s_u = s_v$  e se  $w_e > 0$  então  $s_u \neq s_v$

Senão  $e$  é ruim.

Em outras palavras,  $e$  é boa sse  $w_e s_u s_v < 0$ .

Nó  $u$  está satisfeito se  $\sum_e \text{boa} |w_e| > \sum_e \text{ruim} |w_e|$

Configurações estáveis: todos os nós estão satisfeitos.

# Redes neurais de Hopfield

Rede neural de Hopfield:

grafo  $G = (V, E)$  com peso inteiro  $w_e$  em cada aresta  $e \in E$

Configuração da rede: atribuição de um sinal  $+$  ou  $-$  para cada nó

Aresta  $e = uv$  é boa se  $u$  e  $v$  respeitam seu requisito:

se  $w_e < 0$  então  $s_u = s_v$  e se  $w_e > 0$  então  $s_u \neq s_v$

Senão  $e$  é ruim.

Em outras palavras,  $e$  é boa sse  $w_e s_u s_v < 0$ .

Nó  $u$  está satisfeito se  $\sum_e \text{boa} |w_e| > \sum_e \text{ruim} |w_e|$

Configurações estáveis: todos os nós estão satisfeitos.

Sempre existe uma configuração estável?

# Configurações estáveis

**Configurações estáveis:** todos os nós estão satisfeitos.

Por que o nome 'estável'?

# Configurações estáveis

**Configurações estáveis:** todos os nós estão satisfeitos.

Por que o nome 'estável'?

Se um nó está insatisfeito, ele pode mudar seu sinal, e ficará satisfeito!

# Configurações estáveis

**Configurações estáveis:** todos os nós estão satisfeitos.

Por que o nome 'estável'?

Se um nó está insatisfeito, ele pode mudar seu sinal, e ficará satisfeito!

Então numa configuração estável, nenhum nó quer mudar seu sinal.

# Configurações estáveis

**Configurações estáveis:** todos os nós estão satisfeitos.

Por que o nome 'estável'?

Se um nó está insatisfeito, ele pode mudar seu sinal, e ficará satisfeito!

Então numa configuração estável, nenhum nó quer mudar seu sinal.

Note que, se  $u$  troca seu sinal, então arestas boas incidentes a  $u$  ficam ruins e arestas ruins tornam-se boas.

# Configurações estáveis

**Configurações estáveis:** todos os nós estão satisfeitos.

Por que o nome 'estável'?

Se um nó está insatisfeito, ele pode mudar seu sinal, e ficará satisfeito!

Então numa configuração estável, nenhum nó quer mudar seu sinal.

Note que, se  $u$  que troca seu sinal, então arestas boas incidentes a  $u$  ficam ruins e arestas ruins tornam-se boas.

**Teorema:** Toda rede neural de Hopfield com  $n$  nós tem uma configuração estável, e tal configuração pode ser encontrada em tempo polinomial em  $n$  e  $W = \sum_e |w_e|$ .

## Aula que vem: dois exemplos

- ▶ **Redes neurais de Hopfield:**

Prova do teorema anterior usando busca local.

- ▶ **Problema do corte máximo:**

Um algoritmo de aproximação que usa busca local.