Tópicos de Análise de Algoritmos

Busca local: redes neurais de Hopfield e corte máximo

Secs 12.3 e 12.4 do KT

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

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

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

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

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

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

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

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 $\sum_{e \in \delta(u)\,\text{boa}} |w_e| \geq \sum_{e \in \delta(u)\,\text{ruim}} |w_e|$.

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 $\sum_{e \in \delta(u)\, \text{boa}} |w_e| \geq \sum_{e \in \delta(u)\, \text{ruim}} |w_e|$.

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 $\sum_{e \in \delta(u)\, \text{boa}} |w_e| \geq \sum_{e \in \delta(u)\, \text{ruim}} |w_e|$.

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 $\sum_{e \in \delta(u)\, \text{boa}} |w_e| \geq \sum_{e \in \delta(u)\, \text{ruim}} |w_e|$.

Configurações estáveis: todos os nós estão satisfeitos.

Sempre existe uma configuração estável?

Aresta e é boa sse $w_e\, s_u\, s_v < 0$.

Nó u está satisfeito se $\sum_{e \in \delta(u)\, \text{boa}} |w_e| \geq \sum_{e \in \delta(u)\, \text{ruim}} |w_e|$.

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.

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 $W = \sum_e |w_e|$.

Sempre existe uma configuração estável?

Aresta e é boa sse $w_e \, s_u \, s_v < 0$.

Nó u está satisfeito se $\sum_{e \in \delta(u) \text{ boa}} |w_e| \geq \sum_{e \in \delta(u) \text{ ruim}} |w_e|$.

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.

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 $W = \sum_e |w_e|$.

Sempre existe uma configuração estável?

Aresta e é boa sse $w_e \, s_u \, s_v < 0$.

Nó u está satisfeito se $\sum_{e \in \delta(u) \text{ boa}} |w_e| \geq \sum_{e \in \delta(u) \text{ ruim}} |w_e|$.

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.

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 $W = \sum_e |w_e|$.

Sempre existe uma configuração estável?

Aresta e é boa sse $w_e\, s_u\, s_v < 0$.

Nó u está satisfeito se $\sum_{e \in \delta(u)\, \text{boa}} |w_e| \geq \sum_{e \in \delta(u)\, \text{ruim}} |w_e|$.

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.

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 $W = \sum_e |w_e|$.

Sempre existe uma configuração estável?

Aresta e é boa sse $w_e\, s_u\, s_v < 0$.

Nó u está satisfeito se $\sum_{e \in \delta(u)\, \text{boa}} |w_e| \geq \sum_{e \in \delta(u)\, \text{ruim}} |w_e|$.

Configurações estáveis: todos os nós estão satisfeitos.

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

Algoritmo que busca configuração estável:
comece de uma configuração qualquer
enquanto existe nó insatisfeito, troque seu sinal

Sempre existe uma configuração estável?

Aresta e é boa sse $w_e\, s_u\, s_v < 0$.

Nó u está satisfeito se $\sum_{e \in \delta(u)\, \text{boa}} |w_e| \geq \sum_{e \in \delta(u)\, \text{ruim}} |w_e|$.

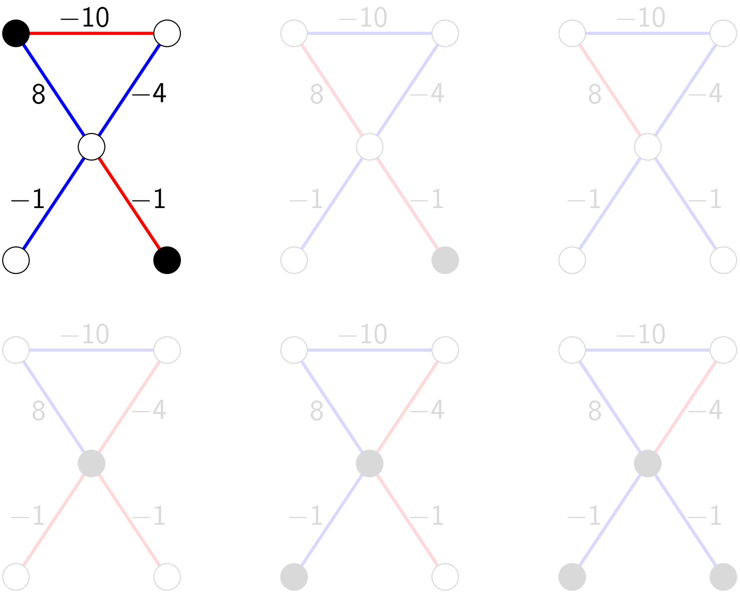
Configurações estáveis: todos os nós estão satisfeitos.

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

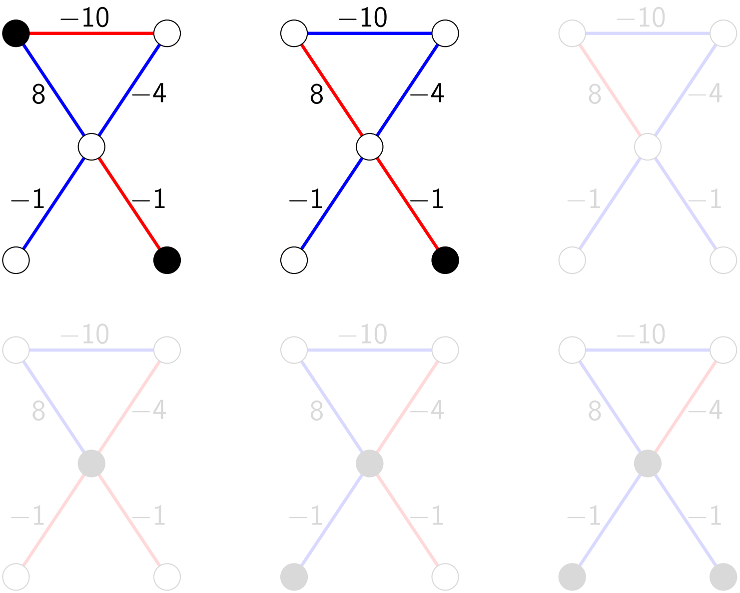
Algoritmo que busca configuração estável:

comece de uma configuração qualquer
enquanto existe nó insatisfeito, troque seu sinal

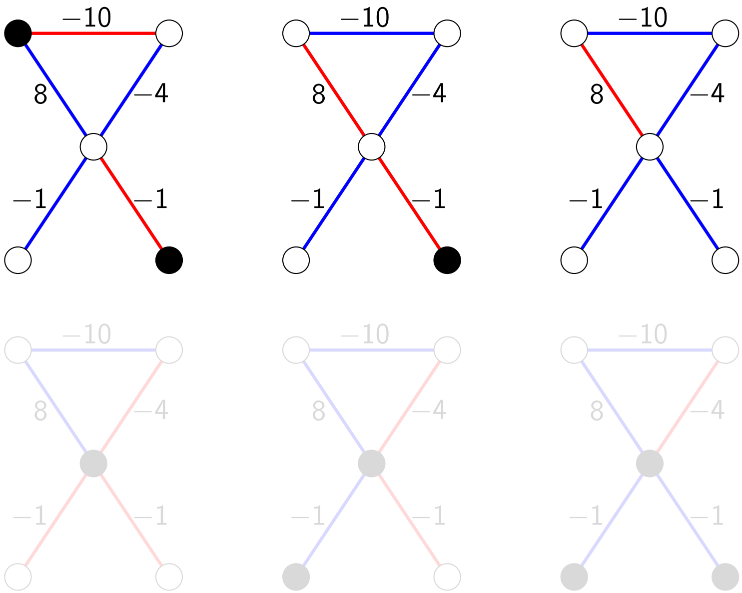
Algoritmo de busca local por configuração estável



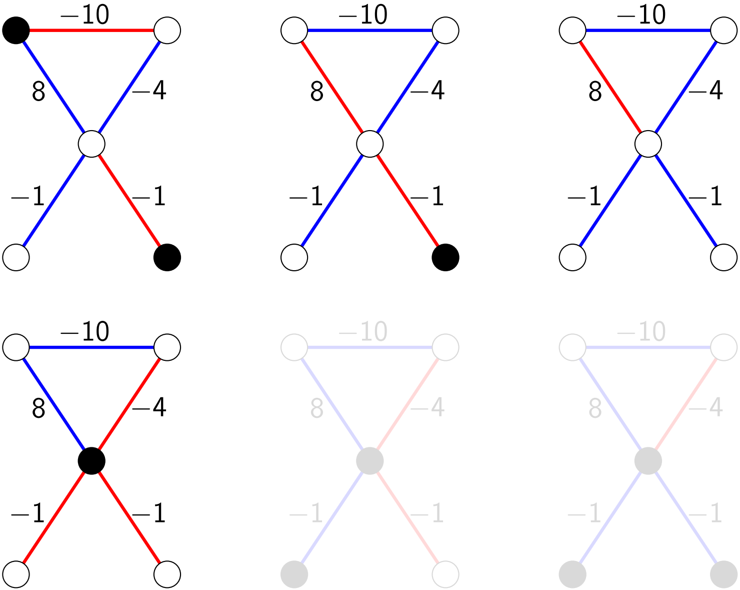
Algoritmo de busca local por configuração estável



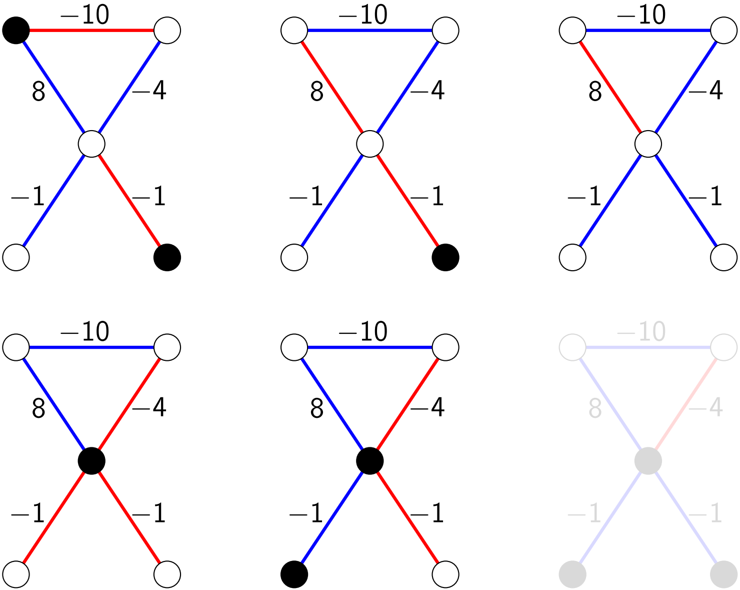
Algoritmo de busca local por configuração estável



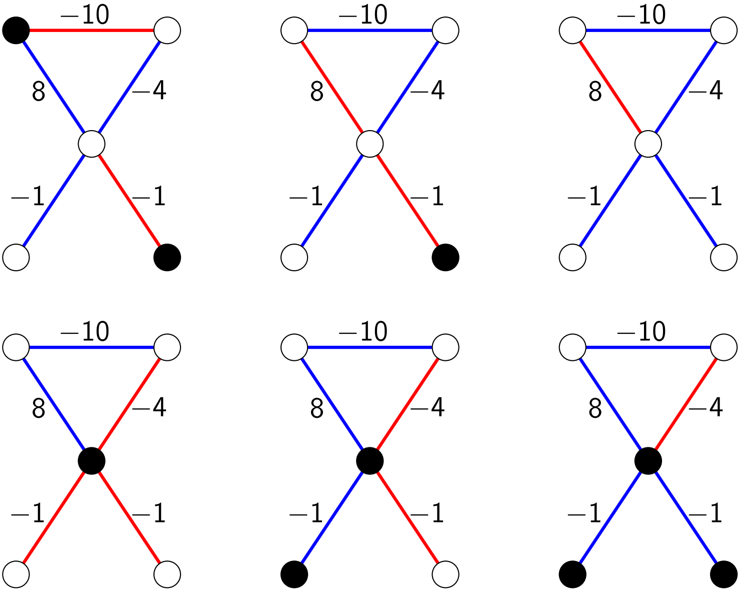
Algoritmo de busca local por configuração estável



Algoritmo de busca local por configuração estável



Algoritmo de busca local por configuração estável



O algoritmo

Comece de uma configuração qualquer
Enquanto existe um nó insatisfeito
  troque o seu sinal

O algoritmo termina?

Se termina, certamente termina numa configuração estável.

Vamos mostrar que o algoritmo acima sempre termina,
e que faz $O(n + W)$ iterações, onde $W = \sum_e |w_e|$.

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 $W = \sum_e |w_e|$.

O algoritmo

Comece de uma configuração qualquer
Enquanto existe um nó insatisfeito
  troque o seu sinal

O algoritmo termina?

Se termina, certamente termina numa configuração estável.

Vamos mostrar que o algoritmo acima sempre termina,
e que faz $O(n + W)$ iterações, onde $W = \sum_e |w_e|$.

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 $W = \sum_e |w_e|$.

O algoritmo

Comece de uma configuração qualquer
Enquanto existe um nó insatisfeito
  troque o seu sinal

O algoritmo termina?

Se termina, certamente termina numa configuração estável.

Vamos mostrar que o algoritmo acima sempre termina,
e que faz $O(n + W)$ iterações, onde $W = \sum_e |w_e|$.

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 $W = \sum_e |w_e|$.

O algoritmo

Comece de uma configuração qualquer
Enquanto existe um nó insatisfeito
  troque o seu sinal

O algoritmo termina?

Se termina, certamente termina numa configuração estável.

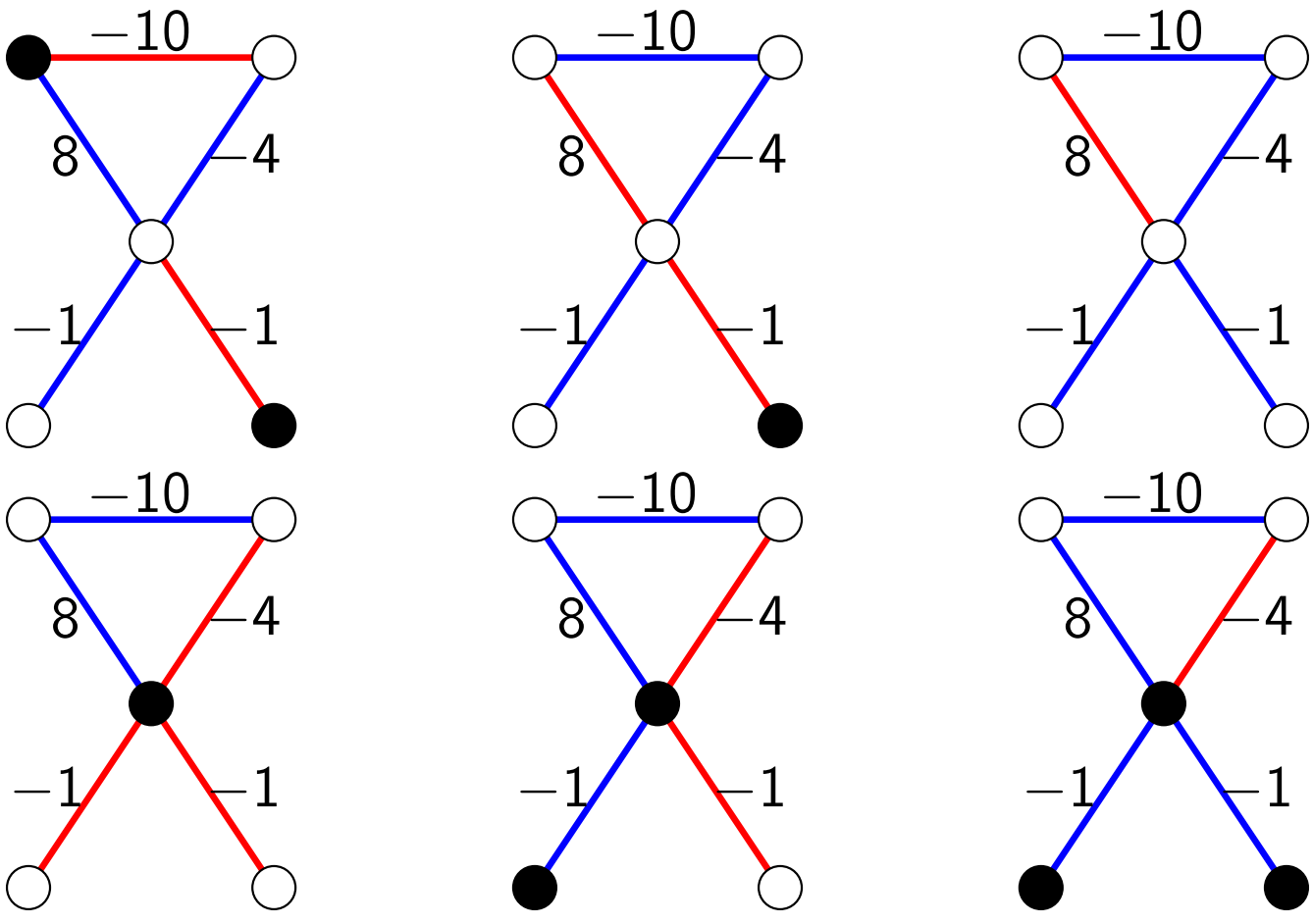
Vamos mostrar que o algoritmo acima sempre termina,
e que faz $O(n + W)$ iterações, onde $W = \sum_e |w_e|$.

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 $W = \sum_e |w_e|$.

Quantas iterações o algoritmo pode fazer?

É verdade que

- o número de nós insatisfeitos diminui a cada iteração?



Quantas iterações o algoritmo pode fazer?

É verdade que

- o número de nós insatisfeitos diminui a cada iteração?
- cada nó troca de sinal no máximo uma vez?

Como medir o "progresso" do algoritmo?

Considere a soma do peso absoluto das arestas boas:

$$\sum_{e \text{ boa}} |w_e|$$

O que acontece com esta soma a cada iteração?

Quantas iterações o algoritmo pode fazer?

É verdade que

- o número de nós insatisfeitos diminui a cada iteração?
- cada nó troca de sinal no máximo uma vez?

Como medir o "progresso" do algoritmo?

Considere a soma do peso absoluto das arestas boas:

$$\sum_{e \text{ boa}} |w_e|$$

O que acontece com esta soma a cada iteração?

Quantas iterações o algoritmo pode fazer?

É verdade que

- o número de nós insatisfeitos diminui a cada iteração?
- cada nó troca de sinal no máximo uma vez?

Como medir o "progresso" do algoritmo?

Considere a soma do peso absoluto das arestas boas:

$$\sum_{e \text{ boa}} |w_e|$$

O que acontece com esta soma a cada iteração?

Análise do algoritmo

Como medir o "progresso" do algoritmo?

O que acontece com a soma do peso absoluto das arestas boas:

$$\sum_{e \text{ boa}} |w_e|$$

O que acontece com esta soma a cada iteração?

Quando o sinal de um nó troca,
as arestas em torno deste nó mudam de boas para ruins, e vice-versa.

Nó u está insatisfeito se $\sum_{e \in \delta(u) \text{ boa}} |w_e| < \sum_{e \in \delta(u) \text{ ruim}} |w_e|$.

Portanto a soma acima aumenta a cada iteração!

Análise do algoritmo

Como medir o "progresso" do algoritmo?

O que acontece com a soma do peso absoluto das arestas boas:

$$\sum_{e \text{ boa}} |w_e|$$

O que acontece com esta soma a cada iteração?

Quando o sinal de um nó troca,
as arestas em torno deste nó mudam de boas para ruins, e vice-versa.

Nó u está insatisfeito se $\sum_{e \in \delta(u) \text{ boa}} |w_e| < \sum_{e \in \delta(u) \text{ ruim}} |w_e|$.

Portanto a soma acima aumenta a cada iteração!

Análise do algoritmo

Como medir o "progresso" do algoritmo?

O que acontece com a soma do peso absoluto das arestas boas:

$$\sum_{e \text{ boa}} |w_e|$$

O que acontece com esta soma a cada iteração?

Quando o sinal de um nó troca,
as arestas em torno deste nó mudam de boas para ruins, e vice-versa.

Nó u está insatisfeito se $\sum_{e \in \delta(u) \text{ boa}} |w_e| < \sum_{e \in \delta(u) \text{ ruim}} |w_e|$.

Portanto a soma acima aumenta a cada iteração!

Análise do algoritmo

A soma do peso absoluto das arestas boas

$$\sum_{e \text{ boa}} |w_e|$$

é sempre maior ou igual a zero.

Ela começa de um valor qualquer maior ou igual a zero
e aumenta a cada iteração.

Como os pesos w_e são inteiros,
ela aumenta de pelo menos um a cada iteração.

A soma é no máximo $W = \sum_e |w_e|$.

Logo o algoritmo faz no máximo W iterações (pseudo-polinomial)
e termina com uma configuração estável.

Análise do algoritmo

A soma do peso absoluto das arestas boas

$$\sum_{e \text{ boa}} |w_e|$$

é sempre maior ou igual a zero.

Ela começa de um valor qualquer maior ou igual a zero
e aumenta a cada iteração.

Como os pesos w_e são inteiros,
ela aumenta de pelo menos um a cada iteração.

A soma é no máximo $W = \sum_e |w_e|$.

Logo o algoritmo faz no máximo W iterações (pseudo-polinomial)
e termina com uma configuração estável.

Análise do algoritmo

A soma do peso absoluto das arestas boas

$$\sum_{e \text{ boa}} |w_e|$$

é sempre maior ou igual a zero.

Ela começa de um valor qualquer maior ou igual a zero
e aumenta a cada iteração.

Como os pesos w_e são inteiros,
ela aumenta de pelo menos um a cada iteração.

A soma é no máximo $W = \sum_e |w_e|$.

Logo o algoritmo faz no máximo W iterações (pseudo-polinomial)
e termina com uma configuração estável.

Análise do algoritmo

A soma do peso absoluto das arestas boas

$$\sum_{e \text{ boa}} |w_e|$$

é sempre maior ou igual a zero.

Ela começa de um valor qualquer maior ou igual a zero
e aumenta a cada iteração.

Como os pesos w_e são inteiros,
ela aumenta de pelo menos um a cada iteração.

A soma é no máximo $W = \sum_e |w_e|$.

Logo o algoritmo faz no máximo W iterações (pseudo-polinomial)
e termina com uma configuração estável.

Problema do corte máximo

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

Um corte é uma partição $\{A, B\}$ de V com $A \neq \emptyset$ e $B \neq \emptyset$.

O peso do corte $\{A, B\}$ é

$$w(A, B) := \sum_{e=xy : x \in A, y \in B} w_e .$$

Dado um grafo $G = (V, E)$ com um peso w_e inteiro positivo em cada aresta $e \in E$, determinar um corte de peso máximo.

Esse problema é NP-difícil e é conhecido como MAXCUT.

Problema do corte máximo

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

Um corte é uma partição $\{A, B\}$ de V com $A \neq \emptyset$ e $B \neq \emptyset$.

O peso do corte $\{A, B\}$ é

$$w(A, B) := \sum_{e=xy : x \in A, y \in B} w_e.$$

Dado um grafo $G = (V, E)$ com um peso w_e inteiro positivo em cada aresta $e \in E$, determinar um corte de peso máximo.

Esse problema é NP-difícil e é conhecido como MAXCUT.

Problema do corte máximo

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

Um corte é uma partição $\{A, B\}$ de V com $A \neq \emptyset$ e $B \neq \emptyset$.

O peso do corte $\{A, B\}$ é

$$w(A, B) := \sum_{e=xy : x \in A, y \in B} w_e.$$

Dado um grafo $G = (V, E)$ com um peso w_e inteiro positivo em cada aresta $e \in E$, determinar um corte de peso máximo.

Esse problema é NP-difícil e é conhecido como MAXCUT.

Algoritmo inspirado no anterior

Comece de um corte $\{A, B\}$ qualquer
Enquanto existe um vértice insatisfeito
  troque-o de conjunto

Vértice insatisfeito:

Se $v \in A$ e $\sum_{u \in N(v) \cap A} w_{uv} > \sum_{u \in N(v) \cap B} w_{uv}$,
  então v prefere mudar para B.

Se $v \in B$ e $\sum_{u \in N(v) \cap B} w_{uv} > \sum_{u \in N(v) \cap A} w_{uv}$,
  então v prefere mudar para A.

O algoritmo termina?

Sim! O peso do corte está sempre aumentando! Então termina!

Podemos dizer algo sobre o peso do corte em que termina?

Algoritmo inspirado no anterior

Comece de um corte $\{A, B\}$ qualquer
Enquanto existe um vértice insatisfeito
  troque-o de conjunto

Vértice insatisfeito:

Se $v \in A$ e $\sum_{u \in N(v) \cap A} w_{uv} > \sum_{u \in N(v) \cap B} w_{uv}$,

  então v prefere mudar para B.

Se $v \in B$ e $\sum_{u \in N(v) \cap B} w_{uv} > \sum_{u \in N(v) \cap A} w_{uv}$,

  então v prefere mudar para A.

O algoritmo termina?

Sim! O peso do corte está sempre aumentando! Então termina!

Podemos dizer algo sobre o peso do corte em que termina?

Algoritmo inspirado no anterior

Comece de um corte $\{A, B\}$ qualquer
Enquanto existe um vértice insatisfeito
  troque-o de conjunto

Vértice insatisfeito:

Se $v \in A$ e $\sum_{u \in N(v) \cap A} w_{uv} > \sum_{u \in N(v) \cap B} w_{uv}$,

  então v prefere mudar para B.

Se $v \in B$ e $\sum_{u \in N(v) \cap B} w_{uv} > \sum_{u \in N(v) \cap A} w_{uv}$,

  então v prefere mudar para A.

O algoritmo termina?

Sim! O peso do corte está sempre aumentando! Então termina!

Podemos dizer algo sobre o peso do corte em que termina?

Algoritmo inspirado no anterior

Comece de um corte $\{A, B\}$ qualquer
Enquanto existe um vértice insatisfeito
  troque-o de conjunto

Vértice insatisfeito:

Se $v \in A$ e $\sum_{u \in N(v) \cap A} w_{uv} > \sum_{u \in N(v) \cap B} w_{uv}$,

  então v prefere mudar para B.

Se $v \in B$ e $\sum_{u \in N(v) \cap B} w_{uv} > \sum_{u \in N(v) \cap A} w_{uv}$,

  então v prefere mudar para A.

O algoritmo termina?

Sim! O peso do corte está sempre aumentando! Então termina!

Podemos dizer algo sobre o peso do corte em que termina?

Algoritmo inspirado no anterior

Comece de um corte $\{A, B\}$ qualquer
Enquanto existe um vértice insatisfeito
  troque-o de conjunto

Vértice insatisfeito:

Se $v \in A$ e $\sum_{u \in N(v) \cap A} w_{uv} > \sum_{u \in N(v) \cap B} w_{uv}$,

  então v prefere mudar para B.

Se $v \in B$ e $\sum_{u \in N(v) \cap B} w_{uv} > \sum_{u \in N(v) \cap A} w_{uv}$,

  então v prefere mudar para A.

O algoritmo termina?

Sim! O peso do corte está sempre aumentando! Então termina!

Podemos dizer algo sobre o peso do corte em que termina?

Qualidade do corte produzido

O peso do corte produzido é

$$\sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} = \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv}.$$

Termina com todo vértice satisfeito:

- Se $v \in A$, então $\sum_{u \in N(v) \cap B} w_{uv} \geq \sum_{u \in N(v) \cap A} w_{uv}$.
- Se $v \in B$, então $\sum_{u \in N(v) \cap A} w_{uv} \geq \sum_{u \in N(v) \cap B} w_{uv}$.

Em outras palavras,

- se $v \in A$, então $\sum_{u \in N(v) \cap B} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ e
- se $v \in B$, então $\sum_{u \in N(v) \cap A} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$.

Qualidade do corte produzido

O peso do corte produzido é

$$\sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} = \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv}.$$

Termina com todo vértice satisfeito:

- Se $v \in A$, então $\sum_{u \in N(v) \cap B} w_{uv} \geq \sum_{u \in N(v) \cap A} w_{uv}$.
- Se $v \in B$, então $\sum_{u \in N(v) \cap A} w_{uv} \geq \sum_{u \in N(v) \cap B} w_{uv}$.

Em outras palavras,

- se $v \in A$, então $\sum_{u \in N(v) \cap B} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ e
- se $v \in B$, então $\sum_{u \in N(v) \cap A} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$.

Qualidade do corte produzido

O peso do corte produzido é

$$\sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} = \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv}.$$

Termina com todo vértice satisfeito:

- Se $v \in A$, então $\sum_{u \in N(v) \cap B} w_{uv} \geq \sum_{u \in N(v) \cap A} w_{uv}$.
- Se $v \in B$, então $\sum_{u \in N(v) \cap A} w_{uv} \geq \sum_{u \in N(v) \cap B} w_{uv}$.

Em outras palavras,

- se $v \in A$, então $\sum_{u \in N(v) \cap B} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ e
- se $v \in B$, então $\sum_{u \in N(v) \cap A} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$.

Qualidade do corte produzido

O peso do corte produzido é

$$\sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} = \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv}.$$

Para todo vértice v, $\sum_{u \in N(v) \cap B} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ se $v \in A$ e

$$\sum_{u \in N(v) \cap A} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv} \text{ se } v \in B.$$

Então

$$\begin{aligned} W = \sum_e w_e &= \frac{1}{2} \sum_{v \in V} \sum_{u \in N(v)} w_{uv} \\ &= \frac{1}{2} \left(\sum_{v \in A} \sum_{u \in N(v)} w_{uv} + \sum_{v \in B} \sum_{u \in N(v)} w_{uv} \right) \\ &\leq \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} + \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv} \\ &= 2 \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv}. \qquad \triangleright\ 2 \times \text{valor do corte produzido} \end{aligned}$$

Como um corte máximo tem peso no máximo W, o algoritmo produz um corte que tem pelo menos metade do peso de um corte máximo.

Qualidade do corte produzido

O peso do corte produzido é

$$\sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} = \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv}.$$

Para todo vértice v, $\sum_{u \in N(v) \cap B} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ se $v \in A$ e

$\sum_{u \in N(v) \cap A} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ se $v \in B$.

Então

$$\begin{aligned} W = \sum_e w_e &= \frac{1}{2} \sum_{v \in V} \sum_{u \in N(v)} w_{uv} \\ &= \frac{1}{2} \left(\sum_{v \in A} \sum_{u \in N(v)} w_{uv} + \sum_{v \in B} \sum_{u \in N(v)} w_{uv} \right) \\ &\leq \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} + \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv} \\ &= 2 \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv}. \qquad \triangleright\ 2 \times \text{valor do corte produzido} \end{aligned}$$

Como um corte máximo tem peso no máximo W, o algoritmo produz um corte que tem pelo menos metade do peso de um corte máximo.

Qualidade do corte produzido

O peso do corte produzido é

$$\sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} = \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv}.$$

Para todo vértice v, $\sum_{u \in N(v) \cap B} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ se $v \in A$ e

$$\sum_{u \in N(v) \cap A} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv} \text{ se } v \in B.$$

Então

$$\begin{aligned} W = \sum_e w_e &= \frac{1}{2} \sum_{v \in V} \sum_{u \in N(v)} w_{uv} \\ &= \frac{1}{2} \left(\sum_{v \in A} \sum_{u \in N(v)} w_{uv} + \sum_{v \in B} \sum_{u \in N(v)} w_{uv} \right) \\ &\leq \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} + \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv} \\ &= 2 \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv}. \qquad \triangleright\ 2 \times \text{valor do corte produzido} \end{aligned}$$

Como um corte máximo tem peso no máximo W, o algoritmo produz um corte que tem pelo menos metade do peso de um corte máximo.

Qualidade do corte produzido

O peso do corte produzido é

$$\sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} = \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv}.$$

Para todo vértice v, $\sum_{u \in N(v) \cap B} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ se $v \in A$ e

$$\sum_{u \in N(v) \cap A} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv} \text{ se } v \in B.$$

Então

$$\begin{aligned} W = \sum_e w_e &= \frac{1}{2} \sum_{v \in V} \sum_{u \in N(v)} w_{uv} \\ &= \frac{1}{2} \left(\sum_{v \in A} \sum_{u \in N(v)} w_{uv} + \sum_{v \in B} \sum_{u \in N(v)} w_{uv} \right) \\ &\leq \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} + \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv} \\ &= 2 \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv}. \qquad \triangleright\ 2 \times \text{valor do corte produzido} \end{aligned}$$

Como um corte máximo tem peso no máximo W, o algoritmo produz um corte que tem pelo menos metade do peso de um corte máximo.

Qualidade do corte produzido

O peso do corte produzido é

$$\sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} = \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv}.$$

Para todo vértice v, $\sum_{u \in N(v) \cap B} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ se $v \in A$ e

$$\sum_{u \in N(v) \cap A} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv} \text{ se } v \in B.$$

Então

$$\begin{aligned} W = \sum_e w_e &= \frac{1}{2} \sum_{v \in V} \sum_{u \in N(v)} w_{uv} \\ &= \frac{1}{2} \left(\sum_{v \in A} \sum_{u \in N(v)} w_{uv} + \sum_{v \in B} \sum_{u \in N(v)} w_{uv} \right) \\ &\leq \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} + \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv} \\ &= 2 \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv}. \qquad \triangleright\ 2 \times \text{valor do corte produzido} \end{aligned}$$

Como um corte máximo tem peso no máximo W, o algoritmo produz um corte que tem pelo menos metade do peso de um corte máximo.

Qualidade do corte produzido

O peso do corte produzido é

$$\sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} = \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv}.$$

Para todo vértice v, $\sum_{u \in N(v) \cap B} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ se $v \in A$ e

$$\sum_{u \in N(v) \cap A} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv} \text{ se } v \in B.$$

Então

$$\begin{aligned} W = \sum_e w_e &= \frac{1}{2} \sum_{v \in V} \sum_{u \in N(v)} w_{uv} \\ &= \frac{1}{2} \left(\sum_{v \in A} \sum_{u \in N(v)} w_{uv} + \sum_{v \in B} \sum_{u \in N(v)} w_{uv} \right) \\ &\leq \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} + \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv} \\ &= 2 \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv}. \qquad \triangleright\ 2 \times \text{valor do corte produzido} \end{aligned}$$

Como um corte máximo tem peso no máximo W, o algoritmo produz um corte que tem pelo menos metade do peso de um corte máximo.

Qualidade do corte produzido

O peso do corte produzido é

$$\sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} = \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv}.$$

Para todo vértice v, $\sum_{u \in N(v) \cap B} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ se $v \in A$ e

$\sum_{u \in N(v) \cap A} w_{uv} \geq \frac{1}{2} \sum_{u \in N(v)} w_{uv}$ se $v \in B$.

Então

$$\begin{aligned} W = \sum_e w_e &= \frac{1}{2} \sum_{v \in V} \sum_{u \in N(v)} w_{uv} \\ &= \frac{1}{2} \left(\sum_{v \in A} \sum_{u \in N(v)} w_{uv} + \sum_{v \in B} \sum_{u \in N(v)} w_{uv} \right) \\ &\leq \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv} + \sum_{v \in B} \sum_{u \in N(v) \cap A} w_{uv} \\ &= 2 \sum_{v \in A} \sum_{u \in N(v) \cap B} w_{uv}. \qquad \triangleright\ 2 \times \text{valor do corte produzido} \end{aligned}$$

Como um corte máximo tem peso no máximo W, o algoritmo produz um corte que tem pelo menos metade do peso de um corte máximo.

2-aproximação?

O algoritmo é então uma 2-aproximação?

Não... porque ele é pseudo-polinomial, e não polinomial.
O número de iterações é $O(W)$.

Altere o algoritmo levemente. Para um $\epsilon > 0$,

se $v \in A$ e $\sum_{u \in N(v) \cap A} w_{uv} \geq \sum_{u \in N(v) \cap B} w_{uv} + \frac{\epsilon}{n} w(A, B)$,

então v muda para B;

se $v \in B$ e $\sum_{u \in N(v) \cap B} w_{uv} \geq \sum_{u \in N(v) \cap A} w_{uv} + \frac{\epsilon}{n} w(A, B)$,

então v muda para A.

Exercício 1: Quantas iterações no máximo o algoritmo faz agora?

Exercício 2: Mostre que essa versão é uma $(2 + \frac{\epsilon}{2})$-aproximação.

2-aproximação?

O algoritmo é então uma 2-aproximação?

Não... porque ele é pseudo-polinomial, e não polinomial.
O número de iterações é $\mathrm{O}(W)$.

Altere o algoritmo levemente. Para um $\epsilon > 0$,

se $v \in A$ e $\sum_{u \in N(v) \cap A} w_{uv} \geq \sum_{u \in N(v) \cap B} w_{uv} + \frac{\epsilon}{n} w(A, B)$,

então v muda para B;

se $v \in B$ e $\sum_{u \in N(v) \cap B} w_{uv} \geq \sum_{u \in N(v) \cap A} w_{uv} + \frac{\epsilon}{n} w(A, B)$,

então v muda para A.

Exercício 1: Quantas iterações no máximo o algoritmo faz agora?

Exercício 2: Mostre que essa versão é uma $(2 + \frac{\epsilon}{2})$-aproximação.

2-aproximação?

O algoritmo é então uma 2-aproximação?

Não... porque ele é pseudo-polinomial, e não polinomial.
O número de iterações é $\mathrm{O}(W)$.

Altere o algoritmo levemente. Para um $\epsilon > 0$,

se $v \in A$ e $\sum_{u \in N(v) \cap A} w_{uv} \geq \sum_{u \in N(v) \cap B} w_{uv} + \frac{\epsilon}{n} w(A, B)$,

então v muda para B;

se $v \in B$ e $\sum_{u \in N(v) \cap B} w_{uv} \geq \sum_{u \in N(v) \cap A} w_{uv} + \frac{\epsilon}{n} w(A, B)$,

então v muda para A.

Exercício 1: Quantas iterações no máximo o algoritmo faz agora?

Exercício 2: Mostre que essa versão é uma $(2 + \frac{\epsilon}{2})$-aproximação.

2-aproximação?

O algoritmo é então uma 2-aproximação?

Não... porque ele é pseudo-polinomial, e não polinomial.
O número de iterações é $\mathrm{O}(W)$.

Altere o algoritmo levemente. Para um $\epsilon > 0$,

se $v \in A$ e $\sum_{u \in N(v) \cap A} w_{uv} \geq \sum_{u \in N(v) \cap B} w_{uv} + \frac{\epsilon}{n} w(A, B)$,

então v muda para B;

se $v \in B$ e $\sum_{u \in N(v) \cap B} w_{uv} \geq \sum_{u \in N(v) \cap A} w_{uv} + \frac{\epsilon}{n} w(A, B)$,

então v muda para A.

Exercício 1: Quantas iterações no máximo o algoritmo faz agora?

Exercício 2: Mostre que essa versão é uma $(2 + \frac{\epsilon}{2})$-aproximação.

2-aproximação?

O algoritmo é então uma 2-aproximação?

Não... porque ele é pseudo-polinomial, e não polinomial.
O número de iterações é $\mathrm{O}(W)$.

Altere o algoritmo levemente. Para um $\epsilon > 0$,

se $v \in A$ e $\sum_{u \in N(v) \cap A} w_{uv} \geq \sum_{u \in N(v) \cap B} w_{uv} + \frac{\epsilon}{n} w(A, B)$,

então v muda para B;

se $v \in B$ e $\sum_{u \in N(v) \cap B} w_{uv} \geq \sum_{u \in N(v) \cap A} w_{uv} + \frac{\epsilon}{n} w(A, B)$,

então v muda para A.

Exercício 1: Quantas iterações no máximo o algoritmo faz agora?

Exercício 2: Mostre que essa versão é uma $(2 + \frac{\epsilon}{2})$-aproximação.

Exercício 1: número de iterações

Quantas iterações no máximo a nova versão do algoritmo faz?

A cada iteração,
o peso do corte aumenta por um fator de pelo menos $1 + \frac{\epsilon}{n}$.

O peso dobra após cada $\frac{2n}{\epsilon}$ iterações, pois $(1 + \frac{\epsilon}{n})^{\frac{2n}{\epsilon}} \geq 2$:

$$(\frac{1}{1+\frac{\epsilon}{n}})^{\frac{2n}{\epsilon}} = (1 - \frac{\frac{\epsilon}{n}}{1+\frac{\epsilon}{n}})^{\frac{2n}{\epsilon}} \leq e^{\frac{-2}{1+\frac{\epsilon}{n}}} = (\frac{1}{e})^{\frac{2}{1+\frac{\epsilon}{n}}} < \frac{1}{e} < \frac{1}{2}.$$

Assim o algoritmo não pode fazer mais que $\frac{2n}{\epsilon} \lg W$ iterações,

ou seja, ele consume tempo polinomial em n e $\lg W$.

Exercício 1: número de iterações

Quantas iterações no máximo a nova versão do algoritmo faz?

A cada iteração,
o peso do corte aumenta por um fator de pelo menos $1 + \frac{\epsilon}{n}$.

O peso dobra após cada $\frac{2n}{\epsilon}$ iterações, pois $(1 + \frac{\epsilon}{n})^{\frac{2n}{\epsilon}} \geq 2$:

$$\left(\frac{1}{1+\frac{\epsilon}{n}}\right)^{\frac{2n}{\epsilon}} = \left(1 - \frac{\frac{\epsilon}{n}}{1+\frac{\epsilon}{n}}\right)^{\frac{2n}{\epsilon}} \leq e^{\frac{-2}{1+\frac{\epsilon}{n}}} = \left(\frac{1}{e}\right)^{\frac{2}{1+\frac{\epsilon}{n}}} < \frac{1}{e} < \frac{1}{2}.$$

Assim o algoritmo não pode fazer mais que $\frac{2n}{\epsilon} \lg W$ iterações,

ou seja, ele consume tempo polinomial em n e $\lg W$.

Exercício 1: número de iterações

Quantas iterações no máximo a nova versão do algoritmo faz?

A cada iteração,
o peso do corte aumenta por um fator de pelo menos $1 + \frac{\epsilon}{n}$.

O peso dobra após cada $\frac{2n}{\epsilon}$ iterações, pois $(1 + \frac{\epsilon}{n})^{\frac{2n}{\epsilon}} \geq 2$:

$$\left(\frac{1}{1+\frac{\epsilon}{n}}\right)^{\frac{2n}{\epsilon}} = \left(1 - \frac{\frac{\epsilon}{n}}{1+\frac{\epsilon}{n}}\right)^{\frac{2n}{\epsilon}} \leq e^{\frac{-2}{1+\frac{\epsilon}{n}}} = \left(\frac{1}{e}\right)^{\frac{2}{1+\frac{\epsilon}{n}}} < \frac{1}{e} < \frac{1}{2}.$$

Assim o algoritmo não pode fazer mais que $\frac{2n}{\epsilon} \lg W$ iterações,

ou seja, ele consume tempo polinomial em n e $\lg W$.

Exercício 2: razão de aproximação

O peso do corte produzido é

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

Para todo $v \in A$, $\sum_{u\in N(v)\cap A} w_{uv} < \sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, ou equivalentemente, $\sum_{u\in N(v)} w_{uv} < 2\sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, e

$$\sum_{v\in A}\sum_{u\in N(v)} w_{uv} < 2w(A,B) + \frac{|A|\epsilon}{n} w(A,B) = (2 + \frac{|A|\epsilon}{n})\, w(A,B).$$

Similarly,

$$\sum_{v\in B}\sum_{u\in N(v)} w_{uv} < (2 + \tfrac{|B|\epsilon}{n})\, w(A,B).$$

Therefore, $2W < (4+\epsilon)\, w(A,B)$,
which implies that $(2 + \frac{\epsilon}{2})\, w(A,B) > W \geq \text{opt}$.

Exercício 2: razão de aproximação

O peso do corte produzido é

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

Para todo $v \in A$, $\sum_{u\in N(v)\cap A} w_{uv} < \sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, ou equivalentemente, $\sum_{u\in N(v)} w_{uv} < 2\sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, e

$$\sum_{v\in A}\sum_{u\in N(v)} w_{uv} < 2w(A,B) + \frac{|A|\epsilon}{n} w(A,B) = (2 + \frac{|A|\epsilon}{n})\, w(A,B).$$

Similarly,

$$\sum_{v\in B}\sum_{u\in N(v)} w_{uv} < (2 + \tfrac{|B|\epsilon}{n})\, w(A,B).$$

Therefore, $2W < (4+\epsilon)\, w(A,B)$,
which implies that $(2+\frac{\epsilon}{2})\, w(A,B) > W \geq \text{opt}$.

Exercício 2: razão de aproximação

O peso do corte produzido é

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

Para todo $v \in A$, $\sum_{u\in N(v)\cap A} w_{uv} < \sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, ou equivalentemente, $\sum_{u\in N(v)} w_{uv} < 2\sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, e

$$\sum_{v\in A}\sum_{u\in N(v)} w_{uv} < 2w(A,B) + \frac{|A|\epsilon}{n} w(A,B) = (2 + \frac{|A|\epsilon}{n})\, w(A,B).$$

Similarly,

$$\sum_{v\in B}\sum_{u\in N(v)} w_{uv} < (2 + \tfrac{|B|\epsilon}{n})\, w(A,B).$$

Therefore, $2W < (4+\epsilon)\, w(A,B)$,
which implies that $(2 + \frac{\epsilon}{2})\, w(A,B) > W \geq \text{opt}$.

Exercício 2: razão de aproximação

O peso do corte produzido é

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

Para todo $v \in A$, $\sum_{u\in N(v)\cap A} w_{uv} < \sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, ou equivalentemente, $\sum_{u\in N(v)} w_{uv} < 2\sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, e

$$\sum_{v\in A}\sum_{u\in N(v)} w_{uv} < 2w(A,B) + \frac{|A|\epsilon}{n} w(A,B) = (2 + \frac{|A|\epsilon}{n})\, w(A,B).$$

Similarly,

$$\sum_{v\in B}\sum_{u\in N(v)} w_{uv} < (2 + \tfrac{|B|\epsilon}{n})\, w(A,B).$$

Therefore, $2W < (4+\epsilon)\, w(A,B)$,
which implies that $(2+\frac{\epsilon}{2})\, w(A,B) > W \geq \text{opt}$.

Exercício 2: razão de aproximação

O peso do corte produzido é

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

Para todo $v \in A$, $\sum_{u\in N(v)\cap A} w_{uv} < \sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, ou equivalentemente, $\sum_{u\in N(v)} w_{uv} < 2\sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, e

$$\sum_{v\in A}\sum_{u\in N(v)} w_{uv} < 2w(A,B) + \frac{|A|\epsilon}{n} w(A,B) = (2 + \frac{|A|\epsilon}{n})\, w(A,B).$$

Similarly,

$$\sum_{v\in B}\sum_{u\in N(v)} w_{uv} < (2 + \tfrac{|B|\epsilon}{n})\, w(A,B).$$

Therefore, $2W < (4+\epsilon)\, w(A,B)$,
which implies that $(2+\frac{\epsilon}{2})\, w(A,B) > W \geq \text{opt}$.

Exercício 2: razão de aproximação

O peso do corte produzido é

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

Para todo $v \in A$, $\sum_{u\in N(v)\cap A} w_{uv} < \sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, ou equivalentemente, $\sum_{u\in N(v)} w_{uv} < 2\sum_{u\in N(v)\cap B} w_{uv} + \frac{\epsilon}{n} w(A,B)$, e

$$\sum_{v\in A}\sum_{u\in N(v)} w_{uv} < 2w(A,B) + \frac{|A|\epsilon}{n} w(A,B) = (2 + \frac{|A|\epsilon}{n})\, w(A,B).$$

Similarly,

$$\sum_{v\in B}\sum_{u\in N(v)} w_{uv} < (2 + \tfrac{|B|\epsilon}{n})\, w(A,B).$$

Therefore, $2W < (4+\epsilon)\, w(A,B)$,
which implies that $(2+\frac{\epsilon}{2})\, w(A,B) > W \geq \mathrm{opt}$.