Heurisztikák

Hajnal Péter

2020. tavasz

A kimerítő keresés

A kimerítő keresés

• A kombinatorikus optimalizálási problémák esetén általában könnyű algoritmusunk egy részét véges sok döntés hozatalaként megfogalmzni.

A kimerítő keresés

• A kombinatorikus optimalizálási problémák esetén általában könnyű algoritmusunk egy részét véges sok döntés hozatalaként megfogalmzni.

• Ezen döntési részen könnyű egy „kimerítő keresési” algoritmust felírni.

A kimerítő keresés

• A kombinatorikus optimalizálási problémák esetén általában könnyű algoritmusunk egy részét véges sok döntés hozatalaként megfogalmzni.

• Ezen döntési részen könnyű egy „kimerítő keresési” algoritmust felírni. Nézzük végig a lehetséges döntési kimeneteleket.

A kimerítő keresés

• A kombinatorikus optimalizálási problémák esetén általában könnyű algoritmusunk egy részét véges sok döntés hozatalaként megfogalmzni.

• Ezen döntési részen könnyű egy „kimerítő keresési” algoritmust felírni. Nézzük végig a lehetséges döntési kimeneteleket.

• Ezek azonban általában legalább exponenciális függésűek az input méretétől, így viszonylag kis inputok esetén is használhatatlanok.

A kimerítő keresés

• A kombinatorikus optimalizálási problémák esetén általában könnyű algoritmusunk egy részét véges sok döntés hozatalaként megfogalmzni.

• Ezen döntési részen könnyű egy „kimerítő keresési” algoritmust felírni. Nézzük végig a lehetséges döntési kimeneteleket.

• Ezek azonban általában legalább exponenciális függésűek az input méretétől, így viszonylag kis inputok esetén is használhatatlanok.

• Sok probléma $\mathcal{NP}$-teljes így ez a nehézség várhatóan szükségszerű.

A kimerítő keresés

• A kombinatorikus optimalizálási problémák esetén általában könnyű algoritmusunk egy részét véges sok döntés hozatalaként megfogalmzni.

• Ezen döntési részen könnyű egy „kimerítő keresési” algoritmust felírni. Nézzük végig a lehetséges döntési kimeneteleket.

• Ezek azonban általában legalább exponenciális függésűek az input méretétől, így viszonylag kis inputok esetén is használhatatlanok.

• Sok probléma $\mathcal{NP}$-teljes így ez a nehézség várhatóan szükségszerű. Gyakorlatban azonban szükséges lehet ilyen problémák megoldása is.

A kimerítő keresés

• A kombinatorikus optimalizálási problémák esetén általában könnyű algoritmusunk egy részét véges sok döntés hozatalaként megfogalmzni.

• Ezen döntési részen könnyű egy „kimerítő keresési” algoritmust felírni. Nézzük végig a lehetséges döntési kimeneteleket.

• Ezek azonban általában legalább exponenciális függésűek az input méretétől, így viszonylag kis inputok esetén is használhatatlanok.

• Sok probléma $\mathcal{NP}$-teljes így ez a nehézség várhatóan szükségszerű. Gyakorlatban azonban szükséges lehet ilyen problémák megoldása is.

• Ekkor heurisztikákkal próbáljuk meg csökkenteni az esetek teljes átnézését.

A Branch-and-Bound séma

A Branch-and-Bound séma

- Az esetek burjánzását gyakran egy fa kettős-águhajtásos növekedésével (**branch**) lehet leírni.

A Branch-and-Bound séma

• Az esetek burjánzását gyakran egy fa kettős-águhajtásos növekedésével (**branch**) lehet leírni. A teljes fa leírása túl költséges.

A Branch-and-Bound séma

- Az esetek burjánzását gyakran egy fa kettős-águhajtásos növekedésével (**branch**) lehet leírni. A teljes fa leírása túl költséges.
- Az egyes irányú növekedésénél a célfüggvény értékét becsüljük (**bound**).

A Branch-and-Bound séma

- Az esetek burjánzását gyakran egy fa kettős-águhajtásos növekedésével (**branch**) lehet leírni. A teljes fa leírása túl költséges.
- Az egyes irányú növekedésénél a célfüggvény értékét becsüljük (**bound**).
- Ezek a becslések néha lehetőséget adnak, hogy kizárhassuk, hogy a keresett optimális hely az aktzális hely „alatt” legyen.

A Branch-and-Bound séma

- Az esetek burjánzását gyakran egy fa kettős-águhajtásos növekedésével (**branch**) lehet leírni. A teljes fa leírása túl költséges.
- Az egyes irányú növekédésénél a célfüggvény értékét becsüljük (**bound**).
- Ezek a becslések néha lehetőséget adnak, hogy kizárhassuk, hogy a keresett optimális hely az aktzális hely „alatt" legyen.
- Így bizonyos irányban ne növeljük a fát.

A Branch-and-Bound séma

- Az esetek burjánzását gyakran egy fa kettős-ághajtásos növekedésével (**branch**) lehet leírni. A teljes fa leírása túl költséges.
- Az egyes irányú növekédésénél a célfüggvény értékét becsüljük (**bound**).
- Ezek a becslések néha lehetőséget adnak, hogy kizárhassuk, hogy a keresett optimális hely az aktzális hely „alatt" legyen.
- Így bizonyos irányban ne növeljük a fát. A teljes fához képest nagy részeket lenyessünk róla.

A Branch-and-Bound séma

- Az esetek burjánzását gyakran egy fa kettős-ághajtásos növekedésével (**branch**) lehet leírni. A teljes fa leírása túl költséges.
- Az egyes irányú növekédésénél a célfüggvény értékét becsüljük (**bound**).
- Ezek a becslések néha lehetőséget adnak, hogy kizárhassuk, hogy a keresett optimális hely az aktzális hely „alatt" legyen.
- Így bizonyos irányban ne növeljük a fát. A teljes fához képest nagy részeket lenyessünk róla.
- Jó heurisztikák speciális esetben jelentős gyorsulást eredményezhetnek.

Feltétel nélküli optimalizálás

Feltétel nélküli optimalizálás

A minta kérdésünk nagyon egyszerű:

Minimalizáljuk	$c(x)$-t

ahol $c(x) : \mathbb{R}^n \to \mathbb{R}$.

Feltétel nélküli optimalizálás

A minta kérdésünk nagyon egyszerű:

Minimalizáljuk	$c(x)$-t

ahol $c(x) : \mathbb{R}^n \to \mathbb{R}$.

• Feladatunk p^* optimális érték, x^* optimális hely meghatározása, közelítése.

Feltétel nélküli optimalizálás

A minta kérdésünk nagyon egyszerű:

Minimalizáljuk	$c(x)$-t

ahol $c(x) : \mathbb{R}^n \to \mathbb{R}$.

- Feladatunk p^* optimális érték, x^* optimális hely meghatározása, közelítése.
- Ha c-ről semmi tudásunk nincs, akkor helyzetünk reménytelen.

Feltétel nélküli optimalizálás

A minta kérdésünk nagyon egyszerű:

Minimalizáljuk	$c(x)$-t

ahol $c(x) : \mathbb{R}^n \to \mathbb{R}$.

- Feladatunk p^* optimális érték, x^* optimális hely meghatározása, közelítése.
- Ha c-ről semmi tudásunk nincs, akkor helyzetünk reménytelen.

Definíció: Tégla

$$[a_1, b_1] \times \ldots \times [a_n, b_n] \subset \mathbb{R}^n$$

alakú ponthalmazokat — ahol n a dimenzió és $-\infty < a_i < b_i < \infty \quad i = 1, \ldots, n$ — téglának nevezzük.

Szép függvényektől elvárt tulajdonságok

Szép függvényektől elvárt tulajdonságok

1) Adott egy $T_0 = [a_1, b_1] \times \ldots \times [a_n, b_n]$ tégla, amelyre $x^* \in T_0$,

Szép függvényektől elvárt tulajdonságok

1) Adott egy $T_0 = [a_1, b_1] \times \ldots \times [a_n, b_n]$ tégla, amelyre $x^* \in T_0$,
2) T tégla esetén van a_T, f_T kiszámolható alsó és felső becslés $\min_{x \in T} c(x)$-re. Továbbá ezek a becslések teljesítik:

Szép függvényektől elvárt tulajdonságok

1) Adott egy $T_0 = [a_1, b_1] \times \ldots \times [a_n, b_n]$ tégla, amelyre $x^* \in T_0$,
2) T tégla esetén van a_T, f_T kiszámolható alsó és felső becslés $\min_{x \in T} c(x)$-re. Továbbá ezek a becslések teljesítik:

   (A) Ha $T = T_1 \overset{\circ}{\cup} T_2$, azaz alkalmas i-re és v-re

$$T_1 = [a_1, b_1] \times \ldots \times [a_{i-1}, b_{i-1}] \times [a_i, v] \times [a_{i+1}, b_{i+1}] \times \ldots \times [a_n, b_n],$$

$$T_2 = [a_1, b_1] \times \ldots \times [a_{i-1}, b_{i-1}] \times [v, b_i] \times [a_{i+1}, b_{i+1}] \times \ldots \times [a_n, b_n].$$

   akkor $a_T \leq a_{T_1}, a_{T_2}$ és $f_{T_1}, f_{T_2} \leq f_T$ teljesül.

Szép függvényektől elvárt tulajdonságok

1) Adott egy $T_0 = [a_1, b_1] \times \ldots \times [a_n, b_n]$ tégla, amelyre $x^* \in T_0$,
2) T tégla esetén van a_T, f_T kiszámolható alsó és felső becslés $\min_{x \in T} c(x)$-re. Továbbá ezek a becslések teljesítik:
   (A) Ha $T = T_1 \overset{\circ}{\cup} T_2$, azaz alkalmas i-re és v-re

$$T_1 = [a_1, b_1] \times \ldots \times [a_{i-1}, b_{i-1}] \times [a_i, v] \times [a_{i+1}, b_{i+1}] \times \ldots \times [a_n, b_n],$$

$$T_2 = [a_1, b_1] \times \ldots \times [a_{i-1}, b_{i-1}] \times [v, b_i] \times [a_{i+1}, b_{i+1}] \times \ldots \times [a_n, b_n].$$

   akkor $a_T \leq a_{T_1}, a_{T_2}$ és $f_{T_1}, f_{T_2} \leq f_T$ teljesül.
   (B) Továbbá minden $\varepsilon > 0$ esetén van olyan $\delta > 0$, hogy tetszőleges T téglára, ha $\forall_i : b_i - a_i \leq \delta$, akkor $0 \leq f_T - a_T \leq \varepsilon$, azaz ha egy tégla mérete egy ponthoz közelít, akkor a fölső és alsó becslése közeli lesz.

Naív algoritmus

Naív algoritmus

- **Cél:** Szép $c(x)$ függvényt szeretnénk minimalizálni.

Pontosabban célunk olyan x keresése, melyre $c(x) \leq p^* + \varepsilon$.

Naív algoritmus

- **Cél:** Szép $c(x)$ függvényt szeretnénk minimalizálni.

Pontosabban célunk olyan x keresése, melyre $c(x) \leq p^* + \varepsilon$.

Naív algoritmus

Naív algoritmus

- **Cél:** Szép $c(x)$ függvényt szeretnénk minimalizálni.

Pontosabban célunk olyan x keresése, melyre $c(x) \leq p^* + \varepsilon$.

Naív algoritmus

(T) // A teljes fa felépítése (BRANCH)

Naív algoritmus

- **Cél:** Szép $c(x)$ függvényt szeretnénk minimalizálni.

Pontosabban célunk olyan x keresése, melyre $c(x) \leq p^* + \varepsilon$.

Naív algoritmus

(T) // A teljes fa felépítése (BRANCH)
Vegyünk egy T_0 téglát és daraboljuk fel úgy, hogy a keletkező kis téglák élei kisebbek legyenek, mint δ.

Naív algoritmus

- **Cél:** Szép $c(x)$ függvényt szeretnénk minimalizálni.

Pontosabban célunk olyan x keresése, melyre $c(x) \leq p^* + \varepsilon$.

Naív algoritmus

(T) // A teljes fa felépítése (BRANCH)
Vegyünk egy T_0 téglát és daraboljuk fel úgy, hogy a keletkező kis téglák élei kisebbek legyenek, mint δ.

(M) // Minden kis tégla vizsgálata

Naív algoritmus

- **Cél:** Szép $c(x)$ függvényt szeretnénk minimalizálni.

Pontosabban célunk olyan x keresése, melyre $c(x) \leq p^* + \varepsilon$.

Naív algoritmus

(T) // A teljes fa felépítése (BRANCH)
Vegyünk egy T_0 téglát és daraboljuk fel úgy, hogy a keletkező kis téglák élei kisebbek legyenek, mint δ.

(M) // Minden kis tégla vizsgálata
Így a hozzájuk tartozó alsó és felső becslés különbsége legfeljebb ε lesz.

Naív algoritmus

- **Cél:** Szép $c(x)$ függvényt szeretnénk minimalizálni.

Pontosabban célunk olyan x keresése, melyre $c(x) \leq p^* + \varepsilon$.

Naív algoritmus

(T) // A teljes fa felépítése (BRANCH)
Vegyünk egy T_0 téglát és daraboljuk fel úgy, hogy a keletkező kis téglák élei kisebbek legyenek, mint δ.

(M) // Minden kis tégla vizsgálata
Így a hozzájuk tartozó alsó és felső becslés különbsége legfeljebb ε lesz.

(Opt) // Optimalizáslás

Naív algoritmus

- **Cél:** Szép $c(x)$ függvényt szeretnénk minimalizálni.

Pontosabban célunk olyan x keresése, melyre $c(x) \leq p^* + \varepsilon$.

Naív algoritmus

(T) // A teljes fa felépítése (BRANCH)
Vegyünk egy T_0 téglát és daraboljuk fel úgy, hogy a keletkező kis téglák élei kisebbek legyenek, mint δ.

(M) // Minden kis tégla vizsgálata
Így a hozzájuk tartozó alsó és felső becslés különbsége legfeljebb ε lesz.

(Opt) // Optimalizáslás
Ezek után számoljuk ki mindegyik kis téglához tartozó alsó becslést,

Naív algoritmus

- **Cél:** Szép $c(x)$ függvényt szeretnénk minimalizálni.

Pontosabban célunk olyan x keresése, melyre $c(x) \leq p^* + \varepsilon$.

Naív algoritmus

(T) // A teljes fa felépítése (BRANCH)
Vegyünk egy T_0 téglát és daraboljuk fel úgy, hogy a keletkező kis téglák élei kisebbek legyenek, mint δ.

(M) // Minden kis tégla vizsgálata
Így a hozzájuk tartozó alsó és felső becslés különbsége legfeljebb ε lesz.

(Opt) // Optimalizáslás
Ezek után számoljuk ki mindegyik kis téglához tartozó alsó becslést,

(Out) // Output

Naív algoritmus

- **Cél:** Szép $c(x)$ függvényt szeretnénk minimalizálni.

Pontosabban célunk olyan x keresése, melyre $c(x) \leq p^* + \varepsilon$.

Naív algoritmus

(T) // A teljes fa felépítése (BRANCH)
Vegyünk egy T_0 téglát és daraboljuk fel úgy, hogy a keletkező kis téglák élei kisebbek legyenek, mint δ.

(M) // Minden kis tégla vizsgálata
Így a hozzájuk tartozó alsó és felső becslés különbsége legfeljebb ε lesz.

(Opt) // Optimalizáslás
Ezek után számoljuk ki mindegyik kis téglához tartozó alsó becslést,

(Out) // Output
A legkisebb alsó becsléssel rendelkező tégla minden pontja jó output lesz.

A javítás filozófiája: Baranch-and-Bound

A javítás filozófiája: Baranch-and-Bound

- A naív algoritmus (Opt) lépése általában exponenciális sok téglára optimalizál.

A javítás filozófiája: Baranch-and-Bound

• A naív algoritmus (Opt) lépése általában exponenciális sok téglára optimalizál. Így alkalmazása nehézségekbe ütközhet a gyakorlatban.

A javítás filozófiája: Baranch-and-Bound

- A naív algoritmus (Opt) lépése általában exponenciális sok téglára optimalizál. Így alkalmazása nehézségekbe ütközhet a gyakorlatban.

- Az alábbiakban egy javított változatot mutatunk be.

A javítás filozófiája: Baranch-and-Bound

• A naív algoritmus (Opt) lépése általában exponenciális sok téglára optimalizál. Így alkalmazása nehézségekbe ütközhet a gyakorlatban.

• Az alábbiakban egy javított változatot mutatunk be. Ahelyett, hogy minden téglát párhuzamosan, „gondolkodás nélkül” továbbvágnánk, csak azokat daraboljuk, maik a legígéretesebbek.

A javítás filozófiája: Baranch-and-Bound

• A naív algoritmus (Opt) lépése általában exponenciális sok téglára optimalizál. Így alkalmazása nehézségekbe ütközhet a gyakorlatban.

• Az alábbiakban egy javított változatot mutatunk be. Ahelyett, hogy minden téglát párhuzamosan, „gondolkodás nélkül" továbbvágnánk, csak azokat daraboljuk, maik a legígéretesebbek.

• A fenti algoritusból hiányzott a BOUND lépés. A vágások sorozata teljes volt, a végső kis kockák analízisénél becsültünk csak.

A javítás filozófiája: Baranch-and-Bound

• A naív algoritmus (Opt) lépése általában exponenciális sok téglára optimalizál. Így alkalmazása nehézségekbe ütközhet a gyakorlatban.

• Az alábbiakban egy javított változatot mutatunk be. Ahelyett, hogy minden téglát párhuzamosan, „gondolkodás nélkül" továbbvágnánk, csak azokat daraboljuk, maik a legígéretesebbek.

• A fenti algoritusból hiányzott a BOUND lépés. A vágások sorozata teljes volt, a végső kis kockák analízisénél becsültünk csak.

• Új eljárásunk során lesz egy $\mathcal{T}$ egy téglarendszerünk, melyre teljesül, hogy elemeik belső pontjai diszjunktak, továbbá $\cup\mathcal{T} = T_0$.

A javítás filozófiája: Baranch-and-Bound

• A naív algoritmus (Opt) lépése általában exponenciális sok téglára optimalizál. Így alkalmazása nehézségekbe ütközhet a gyakorlatban.

• Az alábbiakban egy javított változatot mutatunk be. Ahelyett, hogy minden téglát párhuzamosan, „gondolkodás nélkül" továbbvágnánk, csak azokat daraboljuk, maik a legígéretesebbek.

• A fenti algoritusból hiányzott a BOUND lépés. A vágások sorozata teljes volt, a végső kis kockák analízisénél becsültünk csak.

• Új eljárásunk során lesz egy $\mathcal{T}$ egy téglarendszerünk, melyre teljesül, hogy elemeik belső pontjai diszjunktak, továbbá $\cup\mathcal{T} = T_0$.

• Egy lépésben csak egy téglát darabolunk tovább.

Branch-and-Bound algoritmus

Branch-and-Bound algoritmus

Branch-and-Bound algoritmus

Branch-and-Bound algoritmus

(0) Kezdetben $\mathcal{T} = \{T_0\}$.

Branch-and-Bound algoritmus

Branch-and-Bound algoritmus

(0) Kezdetben $\mathcal{T} = \{T_0\}$.

(1) Kiválasztunk egy „ígéretes” $T \in \mathcal{T}$-t.

Branch-and-Bound algoritmus

Branch-and-Bound algoritmus

(0) Kezdetben $\mathcal{T} = \{T_0\}$.

(1) Kiválasztunk egy „ígéretes” $T \in \mathcal{T}$-t.

(2) Kiválasztunk egy i dimenziót és egy ebbe az irányba mutató élét T-nek.

Branch-and-Bound algoritmus

Branch-and-Bound algoritmus

(0) Kezdetben $\mathcal{T} = \{T_0\}$.

(1) Kiválasztunk egy „ígéretes” $T \in \mathcal{T}$-t.

(2) Kiválasztunk egy i dimenziót és egy ebbe az irányba mutató élét T-nek.

(3) T-t eszerint az él szerint felezzük meg (T' és T'' a két féltégla, $\mathcal{T} \leftarrow \mathcal{T} - \{T\} \cup \{T', T''\}$).

Branch-and-Bound algoritmus

Branch-and-Bound algoritmus

(0) Kezdetben $\mathcal{T} = \{T_0\}$.

(1) Kiválasztunk egy „ígéretes” $T \in \mathcal{T}$-t.

(2) Kiválasztunk egy i dimenziót és egy ebbe az irányba mutató élét T-nek.

(3) T-t eszerint az él szerint felezzük meg (T' és T'' a két féltégla, $\mathcal{T} \leftarrow \mathcal{T} - \{T\} \cup \{T', T''\}$).

(4) Az új $\mathcal{T}$ tégla-rendszerhez számoljuk ki $a_{\mathcal{T}} := \min_{T \in \mathcal{T}} a_T$ és $\Delta = f_T - a_T$, ahol T az a tégla, amelyik alsó becslése $a_{\mathcal{T}}$.

Branch-and-Bound algoritmus

Branch-and-Bound algoritmus

(0) Kezdetben $\mathcal{T} = \{T_0\}$.

(1) Kiválasztunk egy „ígéretes” $T \in \mathcal{T}$-t.

(2) Kiválasztunk egy i dimenziót és egy ebbe az irányba mutató élét T-nek.

(3) T-t eszerint az él szerint felezzük meg (T' és T'' a két féltégla, $\mathcal{T} \leftarrow \mathcal{T} - \{T\} \cup \{T', T''\}$).

(4) Az új $\mathcal{T}$ tégla-rendszerhez számoljuk ki $a_{\mathcal{T}} := \min_{T \in \mathcal{T}} a_T$ és $\Delta = f_T - a_T$, ahol T az a tégla, amelyik alsó becslése $a_{\mathcal{T}}$.

AMÍG $\Delta > \varepsilon$, ADDIG ismételjük (1)-(4) lépéseket.

Branch-and-Bound algoritmus

Branch-and-Bound algoritmus

(0) Kezdetben $\mathcal{T} = \{T_0\}$.

(1) Kiválasztunk egy „ígéretes” $T \in \mathcal{T}$-t.

(2) Kiválasztunk egy i dimenziót és egy ebbe az irányba mutató élét T-nek.

(3) T-t eszerint az él szerint felezzük meg (T' és T'' a két féltégla, $\mathcal{T} \leftarrow \mathcal{T} - \{T\} \cup \{T', T''\}$).

(4) Az új $\mathcal{T}$ tégla-rendszerhez számoljuk ki $a_{\mathcal{T}} := \min_{T \in \mathcal{T}} a_T$ és $\Delta = f_T - a_T$, ahol T az a tégla, amelyik alsó becslése $a_{\mathcal{T}}$.

AMÍG $\Delta > \varepsilon$, ADDIG ismételjük (1)-(4) lépéseket.

(STOP) Ha $f_{\mathcal{T}} - a_{\mathcal{T}} \leq \varepsilon$.

Branch-and-Bound algoritmus

Branch-and-Bound algoritmus

(0) Kezdetben $\mathcal{T} = \{T_0\}$.

(1) Kiválasztunk egy „ígéretes” $T \in \mathcal{T}$-t.

(2) Kiválasztunk egy i dimenziót és egy ebbe az irányba mutató élét T-nek.

(3) T-t eszerint az él szerint felezzük meg (T' és T'' a két féltégla, $\mathcal{T} \leftarrow \mathcal{T} - \{T\} \cup \{T', T''\}$).

(4) Az új $\mathcal{T}$ tégla-rendszerhez számoljuk ki $a_{\mathcal{T}} := \min_{T \in \mathcal{T}} a_T$ és $\Delta = f_T - a_T$, ahol T az a tégla, amelyik alsó becslése $a_{\mathcal{T}}$.

AMÍG $\Delta > \varepsilon$, ADDIG ismételjük (1)-(4) lépéseket.

(STOP) Ha $f_{\mathcal{T}} - a_{\mathcal{T}} \leq \varepsilon$.

(OUTPUT): $a_{\mathcal{T}}, f_{\mathcal{T}}$, melyre $a_{\mathcal{T}} \leq p^* \leq f_{\mathcal{T}}$. Továbbá legyen $T : a_T = a_{\mathcal{T}}$ és output $x \in T$.

Pontosítás (1)

Pontosítás (1)

(1) T legyen az a tégla, amelyre a_T a lehető legkisebb.

Pontosítás (1)

(1) T legyen az a tégla, amelyre a_T a lehető legkisebb.

- Ezen heurisztika előnyét a következő megjegyzés foglalja össze.

Megjegyzés

Ha $T' \in \mathcal{T}$ tetszőleges olyan tégla, hogy $f_T \leq a_{T'}$

Pontosítás (1)

(1) T legyen az a tégla, amelyre a_T a lehető legkisebb.

- Ezen heurisztika előnyét a következő megjegyzés foglalja össze.

Megjegyzés

Ha $T' \in \mathcal{T}$ tetszőleges olyan tégla, hogy $f_T \leq a_{T'}$ (ezt $[a_T, f_T] \leq [a_{T'}, f_{T'}]$-ként jelöljük),

Pontosítás (1)

(1) T legyen az a tégla, amelyre a_T a lehető legkisebb.

- Ezen heurisztika előnyét a következő megjegyzés foglalja össze.

Megjegyzés

Ha $T' \in \mathcal{T}$ tetszőleges olyan tégla, hogy $f_T \leq a_{T'}$ (ezt $[a_T, f_T] \leq [a_{T'}, f_{T'}]$-ként jelöljük), akkor T'-t sosem osztjuk tovább.

Pontosítás (1)

(1) T legyen az a tégla, amelyre a_T a lehető legkisebb.

- Ezen heurisztika előnyét a következő megjegyzés foglalja össze.

Megjegyzés

Ha $T' \in \mathcal{T}$ tetszőleges olyan tégla, hogy $f_T \leq a_{T'}$ (ezt $[a_T, f_T] \leq [a_{T'}, f_{T'}]$-ként jelöljük), akkor T'-t sosem osztjuk tovább.

- Valóban. Mindig lesz olyan tégla a rendszerünkben, ami T része.

Pontosítás (1)

(1) T legyen az a tégla, amelyre a_T a lehető legkisebb.

- Ezen heurisztika előnyét a következő megjegyzés foglalja össze.

Megjegyzés

Ha $T' \in \mathcal{T}$ tetszőleges olyan tégla, hogy $f_T \leq a_{T'}$ (ezt $[a_T, f_T] \leq [a_{T'}, f_{T'}]$-ként jelöljük), akkor T'-t sosem osztjuk tovább.

- Valóban. Mindig lesz olyan tégla a rendszerünkben, ami T része. Ennek alsó becslése olyan, hogy „nem engedi" T' kiválasztását.

Pontosítás (2)

Pontosítás (2)

(2) Legyen i a T tégla leghosszabb oldalának a dimenziója.

Pontosítás (2)

(2) Legyen i a T tégla leghosszabb oldalának a dimenziója.

• Ezen heurisztika kihasználásához további definíciókra lesz szükségünk.

Pontosítás (2)

(2) Legyen i a T tégla leghosszabb oldalának a dimenziója.

• Ezen heurisztika kihasználásához további definíciókra lesz szükségünk.

Definíció

Legyen $T = [a_1 b_1] \times \ldots \times [a_n, b_n]$ egy tégla.

Pontosítás (2)

(2) Legyen i a T tégla leghosszabb oldalának a dimenziója.

• Ezen heurisztika kihasználásához további definíciókra lesz szükségünk.

Definíció

Legyen $T = [a_1 b_1] \times \ldots \times [a_n, b_n]$ egy tégla.

(i) $\text{vol}\,(T) = \Pi_{i=1}^{n}(b_i - a_i)$, a T tégla térfogata.

Pontosítás (2)

(2) Legyen i a T tégla leghosszabb oldalának a dimenziója.

• Ezen heurisztika kihasználásához további definíciókra lesz szükségünk.

Definíció

Legyen $T = [a_1 b_1] \times \ldots \times [a_n, b_n]$ egy tégla.

(i) $\text{vol}\,(T) = \Pi_{i=1}^{n}(b_i - a_i)$, a T tégla térfogata.

(ii) $|T| = \max_{i=1,\ldots,n}(b_i - a_i)$, a T tégla mérete.

Pontosítás (2)

(2) Legyen i a T tégla leghosszabb oldalának a dimenziója.

• Ezen heurisztika kihasználásához további definíciókra lesz szükségünk.

Definíció

Legyen $T = [a_1 b_1] \times \ldots \times [a_n, b_n]$ egy tégla.

(i) $\text{vol}\,(T) = \Pi_{i=1}^{n}(b_i - a_i)$, a T tégla térfogata.

(ii) $|T| = \max_{i=1,\ldots,n}(b_i - a_i)$, a T tégla mérete.

(iii) $\text{torz}(T) = \frac{\max_{i-1,\ldots,n}(b_i - a_i)}{\min_{i=1,\ldots,n}(b_i - a_i)}$ a T tégla torzulása.

Pontosítás (2)

(2) Legyen i a T tégla leghosszabb oldalának a dimenziója.

• Ezen heurisztika kihasználásához további definíciókra lesz szükségünk.

Definíció

Legyen $T = [a_1 b_1] \times \ldots \times [a_n, b_n]$ egy tégla.

(i) $\text{vol}\,(T) = \Pi_{i=1}^{n}(b_i - a_i)$, a T tégla térfogata.

(ii) $|T| = \max_{i=1,\ldots,n}(b_i - a_i)$, a T tégla mérete.

(iii) $\text{torz}(T) = \frac{\max_{i-1,\ldots,n}(b_i - a_i)}{\min_{i=1,\ldots,n}(b_i - a_i)}$ a T tégla torzulása.

• $\text{Torz}(T) \geq 1$ és egyenlőség pontosan akkor áll fent, ha az n dimenziós téglánk egy kocka.

Pontosítás (2)

(2) Legyen i a T tégla leghosszabb oldalának a dimenziója.

• Ezen heurisztika kihasználásához további definíciókra lesz szükségünk.

Definíció

Legyen $T = [a_1 b_1] \times \ldots \times [a_n, b_n]$ egy tégla.

(i) $\text{vol}\,(T) = \Pi_{i=1}^{n}(b_i - a_i)$, a T tégla térfogata.

(ii) $|T| = \max_{i=1,\ldots,n}(b_i - a_i)$, a T tégla mérete.

(iii) $\text{torz}(T) = \frac{\max_{i-1,\ldots,n}(b_i - a_i)}{\min_{i=1,\ldots,n}(b_i - a_i)}$ a T tégla torzulása.

• $\text{Torz}(T) \geq 1$ és egyenlőség pontosan akkor áll fent, ha az n dimenziós téglánk egy kocka.

• A leghosszabb oldal felezésének előnye, hogy eljárásunk során sosem lesz túlságosan „torz” téglánk.

Észrevétel és bizonyítása

Észrevétel és bizonyítása

Észrevétel

Az algoritmus folyamán fellépő mindegyik T téglára:

$$\text{torz}\,(T) \leq \max\{2, \text{torz}(T_0)\},$$

ahol T_0 a kiinduló tégla.

Észrevétel és bizonyítása

Észrevétel

Az algoritmus folyamán fellépő mindegyik T téglára:

$$\text{torz}\,(T) \leq \max\{2, \text{torz}(T_0)\},$$

ahol T_0 a kiinduló tégla.

- Vegyünk egy tetszőleges T téglát, amelyre teljesül az észrevétel.

Észrevétel és bizonyítása

Észrevétel

Az algoritmus folyamán fellépő mindegyik T téglára:

$$\text{torz}\,(T) \leq \max\{2, \text{torz}(T_0)\},$$

ahol T_0 a kiinduló tégla.

• Vegyünk egy tetszőleges T téglát, amelyre teljesül az észrevétel. Ennek leghosszabb oldalt felezzük el.

Észrevétel és bizonyítása

Észrevétel

Az algoritmus folyamán fellépő mindegyik T téglára:

$$\text{torz}\,(T) \leq \max\{2, \text{torz}(T_0)\},$$

ahol T_0 a kiinduló tégla.

• Vegyünk egy tetszőleges T téglát, amelyre teljesül az észrevétel. Ennek leghosszabb oldalt felezzük el.

• **Állítás:** Ekkor két olyan téglát kapunk, melyekre igaz az észrevételben szereplő állítás.

Észrevétel és bizonyítása

Észrevétel

Az algoritmus folyamán fellépő mindegyik T téglára:

$$\text{torz}\,(T) \leq \max\{2, \text{torz}(T_0)\},$$

ahol T_0 a kiinduló tégla.

• Vegyünk egy tetszőleges T téglát, amelyre teljesül az észrevétel. Ennek leghosszabb oldalt felezzük el.

• **Állítás:** Ekkor két olyan téglát kapunk, melyekre igaz az észrevételben szereplő állítás.

• Az állításból teljes indukcióval adódik az Észrevétel.

Észrevétel és bizonyítása

Észrevétel

Az algoritmus folyamán fellépő mindegyik T téglára:

$$\text{torz}\,(T) \leq \max\{2, \text{torz}(T_0)\},$$

ahol T_0 a kiinduló tégla.

- Vegyünk egy tetszőleges T téglát, amelyre teljesül az észrevétel. Ennek leghosszabb oldalt felezzük el.
- **Állítás:** Ekkor két olyan téglát kapunk, melyekre igaz az észrevételben szereplő állítás.
- Az állításból teljes indukcióval adódik az Észrevétel.
- Ennek igazolására tekintsük az alábbi két esetet.

Az eset analízis

Az eset analízis

1. eset: *A kiinduló T téglára torz $(T) > 2$.*

Az eset analízis

1. eset: *A kiinduló T téglára torz $(T) > 2$.*

Ekkor az eredeti leghosszabb él feleződik, az új tégla leghosszabb éle legfeljebb az eredeti leghosszabb élével azonos nagyságú.

Az eset analízis

1. eset: *A kiinduló T téglára torz $(T) > 2$.*

Ekkor az eredeti leghosszabb él feleződik, az új tégla leghosszabb éle legfeljebb az eredeti leghosszabb élével azonos nagyságú.

Az eredeti legrövidebb éle marad az új tégla legrövidebb éle. Így torz (új T) $\leq$ torz (T)

Az eset analízis

1. eset: *A kiinduló T téglára torz $(T) > 2$.*

Ekkor az eredeti leghosszabb él feleződik, az új tégla leghosszabb éle legfeljebb az eredeti leghosszabb élével azonos nagyságú.

Az eredeti legrövidebb éle marad az új tégla legrövidebb éle. Így torz (új T) $\leq$ torz (T)

2. eset: *Tegyük fel, hogy torz $(T) \leq 2$.*

Az eset analízis

1. eset: *A kiinduló T téglára torz $(T) > 2$.*

Ekkor az eredeti leghosszabb él feleződik, az új tégla leghosszabb éle legfeljebb az eredeti leghosszabb élével azonos nagyságú.

Az eredeti legrövidebb éle marad az új tégla legrövidebb éle. Így torz (új T) $\leq$ torz (T)

2. eset: *Tegyük fel, hogy torz $(T) \leq 2$.*

Ekkor a felezés után az eredeti tégla leghosszabb éle lesz az új tégla legrövidebb éle, csak feleakkora méretben.

Az eset analízis

1. eset: *A kiinduló T téglára torz $(T) > 2$.*

Ekkor az eredeti leghosszabb él feleződik, az új tégla leghosszabb éle legfeljebb az eredeti leghosszabb élével azonos nagyságú.

Az eredeti legrövidebb éle marad az új tégla legrövidebb éle. Így torz (új T) $\leq$ torz (T)

2. eset: *Tegyük fel, hogy torz $(T) \leq 2$.*

Ekkor a felezés után az eredeti tégla leghosszabb éle lesz az új tégla legrövidebb éle, csak feleakkora méretben.

A leghossszabb él hossza nem nőhet, így a torzítottságra teljesül, hogy az új T-re torz $(T) \leq 2$ a kiindulási feltétel miatt.

Észrevétel

Észrevétel

Észrevétel

T tetszőleges téglatest esetén

$$|T| \leq \sqrt[n]{(\text{torz } T)^{n-1} \cdot \text{vol } T}.$$

Észrevétel

Észrevétel

T tetszőleges téglatest esetén

$$|T| \leq \sqrt[n]{(\text{torz } T)^{n-1} \cdot \text{vol } T}.$$

$$\text{vol } T = \prod_i (b_i - a_i) \geq \max_i (b_i - a_i) \left(\min_i (b_i - a_i) \right)^{n-1} =$$

$$= \frac{(\max_i (b_i - a_i))^n}{\left(\frac{\max_i (b_i - a_i)}{\min_i (b_i - a_i)} \right)^{n-1}} = \frac{|T|^n}{(\text{torz } T)^{n-1}}.$$

Összegzés

Összegzés

A fentiekből azt kapjuk, hogy:

Összegzés

A fentiekből azt kapjuk, hogy:

Észrevétel

A fenti algoritmus véges felezés után leáll, ha az (1)-es és (2)-es pontot a heurisztika szerint hajtjuk végre.

Összegzés

A fentiekből azt kapjuk, hogy:

Észrevétel

A fenti algoritmus véges felezés után leáll, ha az (1)-es és (2)-es pontot a heurisztika szerint hajtjuk végre.

- Alkalmazzuk az algoritmust addig, amíg N tégla nem keletkezik.

Összegzés

A fentiekből azt kapjuk, hogy:

Észrevétel

A fenti algoritmus véges felezés után leáll, ha az (1)-es és (2)-es pontot a heurisztika szerint hajtjuk végre.

- Alkalmazzuk az algoritmust addig, amíg N tégla nem keletkezik.
- Legyen T a legkisebb térfogatú tégla, amit az algoritmus kialakított (így vol $T \leq$ vol T_0/N).

Összegzés

A fentiekből azt kapjuk, hogy:

Észrevétel

A fenti algoritmus véges felezés után leáll, ha az (1)-es és (2)-es pontot a heurisztika szerint hajtjuk végre.

- Alkalmazzuk az algoritmust addig, amíg N tégla nem keletkezik.
- Legyen T a legkisebb térfogatú tégla, amit az algoritmus kialakított (így vol $T \leq$ vol T_0/N).
- Ekkor felhasználva észrevételeinket azt kapjuk, hogy N választható úgy, hogy $|T| \leq \frac{\delta}{2}$ legyen.

Összegzés

A fentiekből azt kapjuk, hogy:

Észrevétel

A fenti algoritmus véges felezés után leáll, ha az (1)-es és (2)-es pontot a heurisztika szerint hajtjuk végre.

- Alkalmazzuk az algoritmust addig, amíg N tégla nem keletkezik.
- Legyen T a legkisebb térfogatú tégla, amit az algoritmus kialakított (így vol $T \leq$ vol T_0/N).
- Ekkor felhasználva észrevételeinket azt kapjuk, hogy N választható úgy, hogy $|T| \leq \frac{\delta}{2}$ legyen.
- Ez csak úgy lehet, ha T-t olyan T^- felezésével kaptuk, amelyre $|T^-| \leq \delta$ teljesült.

Összegzés

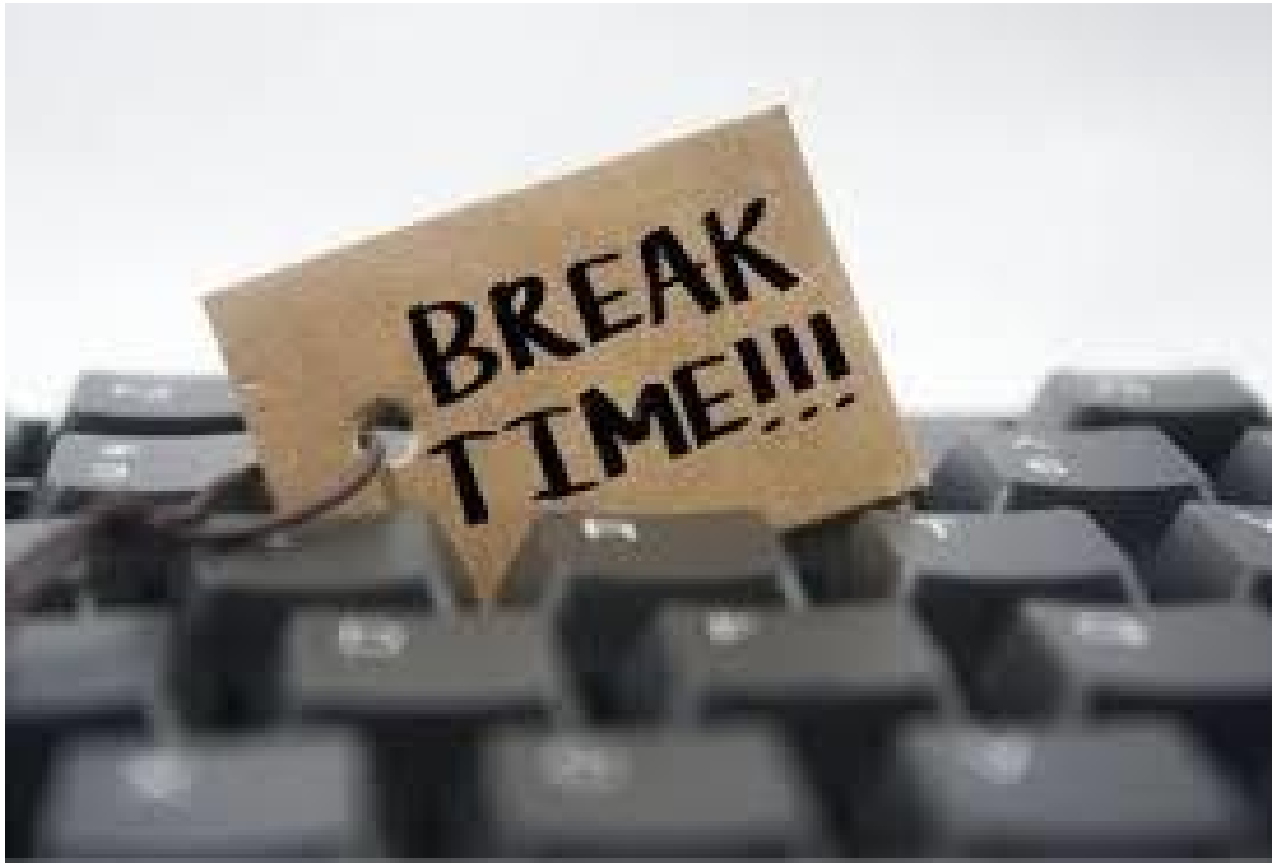
A fentiekből azt kapjuk, hogy:

Észrevétel

A fenti algoritmus véges felezés után leáll, ha az (1)-es és (2)-es pontot a heurisztika szerint hajtjuk végre.

- Alkalmazzuk az algoritmust addig, amíg N tégla nem keletkezik.
- Legyen T a legkisebb térfogatú tégla, amit az algoritmus kialakított (így vol $T \leq$ vol T_0/N).
- Ekkor felhasználva észrevételeinket azt kapjuk, hogy N választható úgy, hogy $|T| \leq \frac{\delta}{2}$ legyen.
- Ez csak úgy lehet, ha T-t olyan T^- felezésével kaptuk, amelyre $|T^-| \leq \delta$ teljesült.
- Ekkor azonban az algoritmusnak le kellett volna állnia (lásd 3) feltétel c szépségére).

Szünet



Az alapkérdés

Az alapkérdés

Minimalizáljuk	$c(x, d)$-t
Feltéve, hogy	$f_i(x, d) \leq 0$, ha $i = 1, 2, \ldots, k$

ahol $x \in \mathbb{R}^n, d \in \{0, 1\}^\nu$ és c, f_i konvex függvények.

Az alapkérdés

Minimalizáljuk	$c(x, d)$-t
Feltéve, hogy	$f_i(x, d) \leq 0$, ha $i = 1, 2, \ldots, k$

ahol $x \in \mathbb{R}^n$, $d \in \{0, 1\}^\nu$ és c, f_i konvex függvények.

• Ha a $d \in \{0, 1\}^\nu$ feltétel nem szerepelne, akkor egy könnyen kezelhető problémánk lenne. $p^* = ?$

Az alapkérdés

Minimalizáljuk	$c(x, d)$-t
Feltéve, hogy	$f_i(x, d) \leq 0$, ha $i = 1, 2, \ldots, k$

ahol $x \in \mathbb{R}^n$, $d \in \{0, 1\}^\nu$ és c, f_i konvex függvények.

- Ha a $d \in \{0, 1\}^\nu$ feltétel nem szerepelne, akkor egy könnyen kezelhető problémánk lenne. $p^* = ?$
- Azonban a feltétel nehézséget okoz, a naív megoldás:

Az alapkérdés

Minimalizáljuk	$c(x, d)$-t
Feltéve, hogy	$f_i(x, d) \leq 0$, ha $i = 1, 2, \ldots, k$

ahol $x \in \mathbb{R}^n$, $d \in \{0, 1\}^\nu$ és c, f_i konvex függvények.

- Ha a $d \in \{0, 1\}^\nu$ feltétel nem szerepelne, akkor egy könnyen kezelhető problémánk lenne. $p^* = ?$
- Azonban a feltétel nehézséget okoz, a naív megoldás:

Naív algoritmus

Az alapkérdés

Minimalizáljuk	$c(x, d)$-t
Feltéve, hogy	$f_i(x, d) \leq 0$, ha $i = 1, 2, \ldots, k$

ahol $x \in \mathbb{R}^n$, $d \in \{0, 1\}^\nu$ és c, f_i konvex függvények.

- Ha a $d \in \{0, 1\}^\nu$ feltétel nem szerepelne, akkor egy könnyen kezelhető problémánk lenne. $p^* = ?$
- Azonban a feltétel nehézséget okoz, a naív megoldás:

Naív algoritmus

(1) Fixáljuk le d-t az összes lehetséges módon.

Az alapkérdés

Minimalizáljuk	$c(x, d)$-t
Feltéve, hogy	$f_i(x, d) \leq 0$, ha $i = 1, 2, \ldots, k$

ahol $x \in \mathbb{R}^n$, $d \in \{0, 1\}^\nu$ és c, f_i konvex függvények.

- Ha a $d \in \{0, 1\}^\nu$ feltétel nem szerepelne, akkor egy könnyen kezelhető problémánk lenne. $p^* = ?$
- Azonban a feltétel nehézséget okoz, a naív megoldás:

Naív algoritmus

(1) Fixáljuk le d-t az összes lehetséges módon.

(2) Az így kapott 2^ν darab problémát kezeljük.

Az alapkérdés

Minimalizáljuk	$c(x, d)$-t
Feltéve, hogy	$f_i(x, d) \leq 0$, ha $i = 1, 2, \ldots, k$

ahol $x \in \mathbb{R}^n$, $d \in \{0, 1\}^\nu$ és c, f_i konvex függvények.

- Ha a $d \in \{0, 1\}^\nu$ feltétel nem szerepelne, akkor egy könnyen kezelhető problémánk lenne. $p^* = ?$
- Azonban a feltétel nehézséget okoz, a naív megoldás:

Naív algoritmus

(1) Fixáljuk le d-t az összes lehetséges módon.

(2) Az így kapott 2^ν darab problémát kezeljük.

(3) A legjobb kapott érték az optimum.

Az alapkérdés

Minimalizáljuk	$c(x, d)$-t
Feltéve, hogy	$f_i(x, d) \leq 0$, ha $i = 1, 2, \ldots, k$

ahol $x \in \mathbb{R}^n$, $d \in \{0, 1\}^\nu$ és c, f_i konvex függvények.

- Ha a $d \in \{0, 1\}^\nu$ feltétel nem szerepelne, akkor egy könnyen kezelhető problémánk lenne. $p^* = ?$
- Azonban a feltétel nehézséget okoz, a naív megoldás:

Naív algoritmus

(1) Fixáljuk le d-t az összes lehetséges módon.

(2) Az így kapott 2^ν darab problémát kezeljük.

(3) A legjobb kapott érték az optimum.

Kicsinek tekinthető ν érték esetén is 2^ν túl nagy lesz ahhoz, hogy ez hatékony kezelés legyen.

Becslések

Becslések

- A relaxációs módszerrel alsó és felső becslés adható az optimális értékre.

Becslések

- A relaxációs módszerrel alsó és felső becslés adható az optimális értékre.
- $d \in \{0,1\}^\nu$ feltételt relaxáljuk $0 \preceq d \preceq 1$ feltétellel $(0, d, 1 \in \mathbb{R}^\nu)$.

Becslések

- A relaxációs módszerrel alsó és felső becslés adható az optimális értékre.
- $d \in \{0,1\}^\nu$ feltételt relaxáljuk $0 \preceq d \preceq 1$ feltétellel $(0, d, 1 \in \mathbb{R}^\nu)$.
- A kapott folytonos konvex $p^*_{\mathbb{R}}$ optimális érték alúlról becsüli az eredeti p^* optimumot.

Becslések

- A relaxációs módszerrel alsó és felső becslés adható az optimális értékre.
- $d \in \{0,1\}^\nu$ feltételt relaxáljuk $0 \preceq d \preceq 1$ feltétellel $(0, d, 1 \in \mathbb{R}^\nu)$.
- A kapott folytonos konvex $p^*_{\mathbb{R}}$ optimális érték alúlról becsüli az eredeti p^* optimumot.
- Egy felső becslést is kapunk, ha egy „jó" lehetséges megoldáson kiértékeljük a c célfüggvényt.

Becslések

• A relaxációs módszerrel alsó és felső becslés adható az optimális értékre.

• $d \in \{0,1\}^\nu$ feltételt relaxáljuk $0 \preceq d \preceq 1$ feltétellel $(0, d, 1 \in \mathbb{R}^\nu)$.

• A kapott folytonos konvex $p^*_{\mathbb{R}}$ optimális érték alúlról becsüli az eredeti p^* optimumot.

• Egy felső becslést is kapunk, ha egy „jó" lehetséges megoldáson kiértékeljük a c célfüggvényt.

• Egy ilyen lehetséges megoldást kapunk, ha a folytonos feladat $x^*_{\mathbb{R}}$ optimális helyében a d ($[0,1]$-beli) komponenseket egészre (azaz $\{0,1\}$-be) kerekítjük.

Becslések

- A relaxációs módszerrel alsó és felső becslés adható az optimális értékre.
- $d \in \{0,1\}^\nu$ feltételt relaxáljuk $0 \preceq d \preceq 1$ feltétellel $(0, d, 1 \in \mathbb{R}^\nu)$.
- A kapott folytonos konvex $p^*_{\mathbb{R}}$ optimális érték alúlról becsüli az eredeti p^* optimumot.
- Egy felső becslést is kapunk, ha egy „jó" lehetséges megoldáson kiértékeljük a c célfüggvényt.
- Egy ilyen lehetséges megoldást kapunk, ha a folytonos feladat $x^*_{\mathbb{R}}$ optimális helyében a d ($[0,1]$-beli) komponenseket egészre (azaz $\{0,1\}$-be) kerekítjük.
- Ezzel az F feladat optimális étékére egy a_F alsó és egy f_F felsó becslést adhatunk.

Szétágazások

Szétágazások

• Vegyük az eredeti feladatot. Tekintsük ezt mint egy problémasereget tartalmazó gyökeres fa gyökere/egy levél ℓ csúcsa.

Szétágazások

- Vegyük az eredeti feladatot. Tekintsük ezt mint egy problémasereget tartalmazó gyökeres fa gyökere/egy levél ℓ csúcsa.
- Válasszunk ki egy $i \in \{1, \ldots, k\}$ komponenst, amely „szabad” .

Szétágazások

- Vegyük az eredeti feladatot. Tekintsük ezt mint egy problémasereget tartalmazó gyökeres fa gyökere/egy levél ℓ csúcsa.
- Válasszunk ki egy $i \in \{1, \ldots, k\}$ komponenst, amely „szabad” .
- Ekkor $d_i = 0$ vagy $d_i = 1$ döntéssel, egy-egy részfeladatot kapunk.

Szétágazások

• Vegyük az eredeti feladatot. Tekintsük ezt mint egy problémasereget tartalmazó gyökeres fa gyökere/egy levél ℓ csúcsa.

• Válasszunk ki egy $i \in \{1, \ldots, k\}$ komponenst, amely „szabad” .

• Ekkor $d_i = 0$ vagy $d_i = 1$ döntéssel, egy-egy részfeladatot kapunk.

• Vegyük a $d_i = 0$ részfeladatot, melynek optimális értéke p_0^*. Ez csupán egy diszkrét változóval kevesebbet tartalmaz, mint az eredeti, kezelhetetlen feladat. Ennek ellenére a fenti alsó/felső becslés technika alkalmazható rá.

Szétágazások

• Vegyük az eredeti feladatot. Tekintsük ezt mint egy problémasereget tartalmazó gyökeres fa gyökere/egy levél ℓ csúcsa.

• Válasszunk ki egy $i \in \{1, \ldots, k\}$ komponenst, amely „szabad” .

• Ekkor $d_i = 0$ vagy $d_i = 1$ döntéssel, egy-egy részfeladatot kapunk.

• Vegyük a $d_i = 0$ részfeladatot, melynek optimális értéke p_0^*. Ez csupán egy diszkrét változóval kevesebbet tartalmaz, mint az eredeti, kezelhetetlen feladat. Ennek ellenére a fenti alsó/felső becslés technika alkalmazható rá.

• Vegyük a $d_i = 1$ részfeladatot, kezeljük ugyanígy.

Szétágazások

• Vegyük az eredeti feladatot. Tekintsük ezt mint egy problémasereget tartalmazó gyökeres fa gyökere/egy levél ℓ csúcsa.

• Válasszunk ki egy $i \in \{1, \ldots, k\}$ komponenst, amely „szabad” .

• Ekkor $d_i = 0$ vagy $d_i = 1$ döntéssel, egy-egy részfeladatot kapunk.

• Vegyük a $d_i = 0$ részfeladatot, melynek optimális értéke p_0^*. Ez csupán egy diszkrét változóval kevesebbet tartalmaz, mint az eredeti, kezelhetetlen feladat. Ennek ellenére a fenti alsó/felső becslés technika alkalmazható rá.

• Vegyük a $d_i = 1$ részfeladatot, kezeljük ugyanígy.

• A két új feladat az előző fához adható: Az ℓ levélnek keletkezik ℓ_0 és ℓ_1 leszármazottja.

Szétágazások

• Vegyük az eredeti feladatot. Tekintsük ezt mint egy problémasereget tartalmazó gyökeres fa gyökere/egy levél ℓ csúcsa.

• Válasszunk ki egy $i \in \{1, \ldots, k\}$ komponenst, amely „szabad” .

• Ekkor $d_i = 0$ vagy $d_i = 1$ döntéssel, egy-egy részfeladatot kapunk.

• Vegyük a $d_i = 0$ részfeladatot, melynek optimális értéke p_0^*. Ez csupán egy diszkrét változóval kevesebbet tartalmaz, mint az eredeti, kezelhetetlen feladat. Ennek ellenére a fenti alsó/felső becslés technika alkalmazható rá.

• Vegyük a $d_i = 1$ részfeladatot, kezeljük ugyanígy.

• A két új feladat az előző fához adható: Az ℓ levélnek keletkezik ℓ_0 és ℓ_1 leszármazottja.

• Az eredeti feladatok és a két részfeladat egy gyökeres fában ábrázolható.

A Branch-and-Bound algoritmus

A Branch-and-Bound algoritmus

A Branch-and-Bound algoritmus

A Branch-and-Bound algoritmus

(0) Legyen T egy 1-pontú gyökeres fa, amely egyetlen csúcsa (egyben levele) a kiinduló probléma.

A Branch-and-Bound algoritmus

A Branch-and-Bound algoritmus

(0) Legyen T egy 1-pontú gyökeres fa, amely egyetlen csúcsa (egyben levele) a kiinduló probléma.
Számoljuk ki az erre a problémára vonatkozó a, f alsó/felső becslés.

AMÍG $f - a > \varepsilon$

(1) Egy ℓ levél/probléma kiválasztása T-ből.

(2) A kiválasztott ℓ levél/probléma egy eddig nem fixált d komponensének kiválasztása. Jelöljük ezt a komponenst i-vel.

A Branch-and-Bound algoritmus (folytatás)

A Branch-and-Bound algoritmus (folytatás)

A Branch-and-Bound algoritmus(folytatás)

(3a) Legyen ℓ_0 az a probléma, amit ℓ-ből kapunk $d_i = 0$ választással.

(3b$_0$) A maradék komponenseket relaxáljuk, és ebből számoljuk ki az a_0, f_0 alsó és felső becslést a fenti módon.

(3b$_1$) Legyen ℓ_1 az a probléma, amit ℓ-ből kapunk $d_i = 1$ választással. A maradék komponenseket relaxáljuk, és ebből számoljuk ki az a_1, f_1 alsó és felső becslést a fenti módon.

(4) Legyen T az a fa, amit az ℓ-ből történő ℓ_0 és ℓ_1 leveleket behozó ághajtással kapunk. Legyen $a = \min\{a, a_0, a_1\}$ és $f = \min\{f, f_0, f_1\}$.

A részletek tisztázása

A részletek tisztázása

- (1)-ben a legkisebb alsó becslésű levelet választjuk.

A részletek tisztázása

- (1)-ben a legkisebb alsó becslésű levelet választjuk.

- A (2)-es lépésnél a levél relaxált problémájának optimális helye alapján azt a komponenst választunk, amelyik komponens a legközelebb van a $\frac{1}{2}$-hez.

A részletek tisztázása

- (1)-ben a legkisebb alsó becslésű levelet választjuk.

- A (2)-es lépésnél a levél relaxált problémájának optimális helye alapján azt a komponenst választunk, amelyik komponens a legközelebb van a $\frac{1}{2}$-hez.

- Itt is nyilvánvaló, hogy néhány részfeladatnál felléphet, hogy nem kell tovább „felezni”.

A részletek tisztázása

- (1)-ben a legkisebb alsó becslésű levelet választjuk.

- A (2)-es lépésnél a levél relaxált problémájának optimális helye alapján azt a komponenst választunk, amelyik komponens a legközelebb van a $\frac{1}{2}$-hez.

- Itt is nyilvánvaló, hogy néhány részfeladatnál felléphet, hogy nem kell tovább „felezni”.

- Fánk nem növekedhet tovább mint a teljes ν mélységű bináris fa.

Egy új alapkérdés

Egy új alapkérdés

Minimalizáljuk	$\lvert\{i : x_i \neq 0\}\rvert$-t
Feltéve, hogy	$Ax \preceq b$,

ahol $x \in \mathbb{R}^n, A \in \mathbb{R}^{k \times n}, b \in \mathbb{R}^k$.

Egy új alapkérdés

Minimalizáljuk	$\lvert\{i : x_i \neq 0\}\rvert$-t
Feltéve, hogy	$Ax \preceq b$,

ahol $x \in \mathbb{R}^n, A \in \mathbb{R}^{k \times n}, b \in \mathbb{R}^k$.

• Azaz lineáris egyenlőtlenségrendszert akarunk megoldani, ahol a megoldásnak a lehető legkevesebb nemnulla kompononse van.

Egy új alapkérdés

Minimalizáljuk	$\lvert\{i : x_i \neq 0\}\rvert$-t
Feltéve, hogy	$Ax \preceq b$,

ahol $x \in \mathbb{R}^n, A \in \mathbb{R}^{k \times n}, b \in \mathbb{R}^k$.

• Azaz lineáris egyenlőtlenségrendszert akarunk megoldani, ahol a megoldásnak a lehető legkevesebb nemnulla kompononse van.

• Feladatunkat visszavezetjük a vegyes konvex-egész típusú feladatra.

Segéd LP feladatok

Segéd LP feladatok

A redukcióhoz először oldjuk meg az alábbi $2n$ darab LP feladatot $(i = 1, \ldots, n)$

Segéd LP feladatok

A redukcióhoz először oldjuk meg az alábbi $2n$ darab LP feladatot $(i = 1, \ldots, n)$

Minimalizáljuk	x_i-t
Feltéve, hogy	$Ax \preceq b$,

Segéd LP feladatok

A redukcióhoz először oldjuk meg az alábbi $2n$ darab LP feladatot $(i = 1, \ldots, n)$

Minimalizáljuk	x_i-t
Feltéve, hogy	$Ax \preceq b$,

és

Maximalizáljuk	x_i-t
Feltéve, hogy	$Ax \preceq b$.

Segéd LP feladatok

A redukcióhoz először oldjuk meg az alábbi $2n$ darab LP feladatot $(i = 1, \ldots, n)$

Minimalizáljuk	x_i-t
Feltéve, hogy	$Ax \preceq b$,

és

Maximalizáljuk	x_i-t
Feltéve, hogy	$Ax \preceq b$.

Legyen az első n db LP feladat optimuma m_i,

Segéd LP feladatok

A redukcióhoz először oldjuk meg az alábbi $2n$ darab LP feladatot $(i = 1, \ldots, n)$

Minimalizáljuk	x_i-t
Feltéve, hogy	$Ax \preceq b$,

és

Maximalizáljuk	x_i-t
Feltéve, hogy	$Ax \preceq b$.

Legyen az első n db LP feladat optimuma m_i, a második n db-é pedig M_i.

Az új alapkérdés mint MIP

Az új alapkérdés mint MIP

- Ekkor az $Ax \preceq b$ feltételből következik, hogy $m_i \leq x_i \leq M_i$.

Az új alapkérdés mint MIP

- Ekkor az $Ax \preceq b$ feltételből következik, hogy $m_i \leq x_i \leq M_i$.
- Feltételeinkhez adjuk hozzá az $m_i y_i \leq x_i \leq M_i y_i$ feltételeket, ahol $y_i \in \{0, 1\}$, új Boole-változók.

Az új alapkérdés mint MIP

- Ekkor az $Ax \preceq b$ feltételből következik, hogy $m_i \leq x_i \leq M_i$.
- Feltételeinkhez adjuk hozzá az $m_i y_i \leq x_i \leq M_i y_i$ feltételeket, ahol $y_i \in \{0, 1\}$, új Boole-változók.
- $y_i = 1$ érték esetén a feltétel semmitmondó, míg $y_i = 0$ esetén $x_i = 0$-t kényszerít.

Az új alapkérdés mint MIP

- Ekkor az $Ax \preceq b$ feltételből következik, hogy $m_i \leq x_i \leq M_i$.
- Feltételeinkhez adjuk hozzá az $m_i y_i \leq x_i \leq M_i y_i$ feltételeket, ahol $y_i \in \{0, 1\}$, új Boole-változók.
- $y_i = 1$ érték esetén a feltétel semmitmondó, míg $y_i = 0$ esetén $x_i = 0$-t kényszerít.
- Így célunk minél több $y_i = 0$, azaz $\sum_{i=1}^{n} y_i$ minimalizálása.

Az új alapkérdés mint MIP

- Ekkor az $Ax \preceq b$ feltételből következik, hogy $m_i \leq x_i \leq M_i$.
- Feltételeinkhez adjuk hozzá az $m_i y_i \leq x_i \leq M_i y_i$ feltételeket, ahol $y_i \in \{0, 1\}$, új Boole-változók.
- $y_i = 1$ érték esetén a feltétel semmitmondó, míg $y_i = 0$ esetén $x_i = 0$-t kényszerít.
- Így célunk minél több $y_i = 0$, azaz $\sum_{i=1}^{n} y_i$ minimalizálása.
- Tehát az eredetivel ekvivalens vegyes IP probléma:

Az új alapkérdés mint MIP

- Ekkor az $Ax \preceq b$ feltételből következik, hogy $m_i \leq x_i \leq M_i$.
- Feltételeinkhez adjuk hozzá az $m_i y_i \leq x_i \leq M_i y_i$ feltételeket, ahol $y_i \in \{0, 1\}$, új Boole-változók.
- $y_i = 1$ érték esetén a feltétel semmitmondó, míg $y_i = 0$ esetén $x_i = 0$-t kényszerít.
- Így célunk minél több $y_i = 0$, azaz $\sum_{i=1}^{n} y_i$ minimalizálása.
- Tehát az eredetivel ekvivalens vegyes IP probléma:

Minimalizáljuk	$\sum_{i=1}^{n} y_i$-t	
Feltéve, hogy	$Ax \preceq b$	
	$m_i y_i \leq x_i \leq M_i y_i,$	$i = 1, 2, \ldots, k$

ahol $x \in \mathbb{R}^n, y = (y_i)_{i=1}^n \in \{0, 1\}^n,\ A \in \mathbb{R}^{k \times n}, b \in \mathbb{R}^k$.

Az új alapkérdés mint MIP

- Ekkor az $Ax \preceq b$ feltételből következik, hogy $m_i \leq x_i \leq M_i$.
- Feltételeinkhez adjuk hozzá az $m_i y_i \leq x_i \leq M_i y_i$ feltételeket, ahol $y_i \in \{0,1\}$, új Boole-változók.
- $y_i = 1$ érték esetén a feltétel semmitmondó, míg $y_i = 0$ esetén $x_i = 0$-t kényszerít.
- Így célunk minél több $y_i = 0$, azaz $\sum_{i=1}^{n} y_i$ minimalizálása.
- Tehát az eredetivel ekvivalens vegyes IP probléma:

Minimalizáljuk	$\sum_{i=1}^{n} y_i$-t	
Feltéve, hogy	$Ax \preceq b$	
	$m_i y_i \leq x_i \leq M_i y_i,$	$i = 1, 2, \ldots, k$

ahol $x \in \mathbb{R}^n, y = (y_i)_{i=1}^n \in \{0,1\}^n,\ A \in \mathbb{R}^{k \times n}, b \in \mathbb{R}^k$.

- A korábbi módszerünk alkalmazható.

Szünet

Alapkérdés: IP

Alapkérdés: IP

- Emlékeztető: Mit tudunk a lineáris egyenletrendszerek megoldásának „logikájáról”.

Alapkérdés: IP

- Emlékeztető: Mit tudunk a lineáris egyenletrendszerek megoldásának „logikájáról”.
- Már általános iskolában megtanultuk, hogy

Alapkérdés: IP

- Emlékeztető: Mit tudunk a lineáris egyenletrendszerek megoldásának „logikájáról”.
- Már általános iskolában megtanultuk, hogy

(i) Egyenlőtlenségek összeadhatók (feltesszük, hogy megállapodunk, mindig a kisebb egyenlőoldal szerepel a bal oldalon).

Alapkérdés: IP

- Emlékeztető: Mit tudunk a lineáris egyenletrendszerek megoldásának „logikájáról”.
- Már általános iskolában megtanultuk, hogy

(i) Egyenlőtlenségek összeadhatók (feltesszük, hogy megállapodunk, mindig a kisebb egyenlőoldal szerepel a bal oldalon).

(ii) Egyenlőtlenségek nemnegatív számmal szorozhatók (a 0-val való szorzás a — nyilván igaz — $0 \leq 0$ egyenlőtlenséghez vezet).

Alapkérdés: IP

• Emlékeztető: Mit tudunk a lineáris egyenletrendszerek megoldásának „logikájáról”.

• Már általános iskolában megtanultuk, hogy

(i) Egyenlőtlenségek összeadhatók (feltesszük, hogy megállapodunk, mindig a kisebb egyenlőoldal szerepel a bal oldalon).

(ii) Egyenlőtlenségek nemnegatív számmal szorozhatók (a 0-val való szorzás a — nyilván igaz — $0 \leq 0$ egyenlőtlenséghez vezet).

• Az így „levezetett” egyenlőtlenségek a kiinduló feltételek következményei.

Alapkérdés: IP

• Emlékeztető: Mit tudunk a lineáris egyenletrendszerek megoldásának „logikájáról”.

• Már általános iskolában megtanultuk, hogy

(i) Egyenlőtlenségek összeadhatók (feltesszük, hogy megállapodunk, mindig a kisebb egyenlőoldal szerepel a bal oldalon).

(ii) Egyenlőtlenségek nemnegatív számmal szorozhatók (a 0-val való szorzás a — nyilván igaz — $0 \leq 0$ egyenlőtlenséghez vezet).

• Az így „levezetett” egyenlőtlenségek a kiinduló feltételek következményei. A feltételrendszerünkhöz hozzáadva nem változik a lehetséges megoldások halmaza.

Alapkérdés: IP

• Emlékeztető: Mit tudunk a lineáris egyenletrendszerek megoldásának „logikájáról”.

• Már általános iskolában megtanultuk, hogy

(i) Egyenlőtlenségek összeadhatók (feltesszük, hogy megállapodunk, mindig a kisebb egyenlőoldal szerepel a bal oldalon).

(ii) Egyenlőtlenségek nemnegatív számmal szorozhatók (a 0-val való szorzás a — nyilván igaz — $0 \leq 0$ egyenlőtlenséghez vezet).

• Az így „levezetett” egyenlőtlenségek a kiinduló feltételek következményei. A feltételrendszerünkhöz hozzáadva nem változik a lehetséges megoldások halmaza.

• Ha ilyen módon nyerünk új egyenlőtlenséget, akkor azt mondjuk, hogy L-következtetést hajtottunk végre.

Alapkérdés: IP

• Emlékeztető: Mit tudunk a lineáris egyenletrendszerek megoldásának „logikájáról”.

• Már általános iskolában megtanultuk, hogy

(i) Egyenlőtlenségek összeadhatók (feltesszük, hogy megállapodunk, mindig a kisebb egyenlőoldal szerepel a bal oldalon).

(ii) Egyenlőtlenségek nemnegatív számmal szorozhatók (a 0-val való szorzás a — nyilván igaz — $0 \leq 0$ egyenlőtlenséghez vezet).

• Az így „levezetett” egyenlőtlenségek a kiinduló feltételek következményei. A feltételrendszerünkhöz hozzáadva nem változik a lehetséges megoldások halmaza.

• Ha ilyen módon nyerünk új egyenlőtlenséget, akkor azt mondjuk, hogy L-következtetést hajtottunk végre. (L$\equiv$ lineáris, valós.)

Egészértékűség

Egészértékűség

- A fenti logika persze egyenlőtlenségrendszerek egész megoldásainak keresése közben is alkalmazható.

Egészértékűség

• A fenti logika persze egyenlőtlenségrendszerek egész megoldásainak keresése közben is alkalmazható.

• Egyenlőtlenségeink nemnegatív együtthatós lineáris kombinációit véve feltételeink egy következményét kapjuk.

Egészértékűség

- A fenti logika persze egyenlőtlenségrendszerek egész megoldásainak keresése közben is alkalmazható.

- Egyenlőtlenségeink nemnegatív együtthatós lineáris kombinációit véve feltételeink egy következményét kapjuk.

- Speciálisan nem vesztünk lehetséges valós megoldást.

Egészértékűség

• A fenti logika persze egyenlőtlenségrendszerek egész megoldásainak keresése közben is alkalmazható.

• Egyenlőtlenségeink nemnegatív együtthatós lineáris kombinációit véve feltételeink egy következményét kapjuk.

• Speciálisan nem vesztünk lehetséges valós megoldást.

• Ha azonban a lehetséges megoldások az egészek, akkor arra törekszünk, hogy egész megoldásokat ne veszítsünk el.

Egészértékűség

• A fenti logika persze egyenlőtlenségrendszerek egész megoldásainak keresése közben is alkalmazható.

• Egyenlőtlenségeink nemnegatív együtthatós lineáris kombinációit véve feltételeink egy következményét kapjuk.

• Speciálisan nem vesztünk lehetséges valós megoldást.

• Ha azonban a lehetséges megoldások az egészek, akkor arra törekszünk, hogy egész megoldásokat ne veszítsünk el.

• Új logikai következtetést is tehetünk.

A kiterjesztett logika

A kiterjesztett logika

Vizsgáljuk a következő egészértékű programozási feladatot:

Minimalizáljuk	$a^T x$-t
Feltéve, hogy	$Ax \preceq b$
	$x \succeq 0,\ x \in \mathbb{Z}^n$

A kiterjesztett logika

Vizsgáljuk a következő egészértékű programozási feladatot:

Minimalizáljuk	$a^T x$-t
Feltéve, hogy	$Ax \preceq b$
	$x \succeq 0$, $x \in \mathbb{Z}^n$

Egy új következtetés

Legyen $\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \leq \beta$ egyenlőtlenség a feltételek egy L-következménye.

A kiterjesztett logika

Vizsgáljuk a következő egészértékű programozási feladatot:

Minimalizáljuk	$a^T x$-t
Feltéve, hogy	$Ax \preceq b$
	$x \succeq 0,\ x \in \mathbb{Z}^n$

Egy új következtetés

Legyen $\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \leq \beta$ egyenlőtlenség a feltételek egy L-következménye. Azaz az összes lehetséges valós megoldás teljesítse.

A kiterjesztett logika

Vizsgáljuk a következő egészértékű programozási feladatot:

Minimalizáljuk	$a^T x$-t
Feltéve, hogy	$Ax \preceq b$
	$x \succeq 0,\ x \in \mathbb{Z}^n$

Egy új következtetés

Legyen $\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \leq \beta$ egyenlőtlenség a feltételek egy L-következménye. Azaz az összes lehetséges valós megoldás teljesítse.

Ekkor

$$\lfloor \alpha_1 \rfloor x_1 + \lfloor \alpha_2 \rfloor x_2 + \ldots + \lfloor \alpha_n \rfloor x_n \leq \lfloor \beta \rfloor$$

egyenlőtlenség is a feltételek következménye.

A kiterjesztett logika

Vizsgáljuk a következő egészértékű programozási feladatot:

Minimalizáljuk	$a^T x$-t
Feltéve, hogy	$Ax \preceq b$
	$x \succeq 0,\ x \in \mathbb{Z}^n$

Egy új következtetés

Legyen $\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \leq \beta$ egyenlőtlenség a feltételek egy L-következménye. Azaz az összes lehetséges valós megoldás teljesítse.

Ekkor

$$\lfloor \alpha_1 \rfloor x_1 + \lfloor \alpha_2 \rfloor x_2 + \ldots + \lfloor \alpha_n \rfloor x_n \leq \lfloor \beta \rfloor$$

egyenlőtlenség is a feltételek következménye. A feltételeinkhez való hozzáadásával nem veszítünk lehetséges (egész) megoldást.

Bizonyítás

Bizonyítás

- Valóban: $\lfloor \alpha \rfloor \leq \alpha$.

Bizonyítás

- Valóban: $\lfloor \alpha \rfloor \leq \alpha$.

- Így nemnegatív x-re $\lfloor \alpha \rfloor x \leq \alpha x$.

Bizonyítás

- Valóban: $\lfloor \alpha \rfloor \leq \alpha$.

- Így nemnegatív x-re $\lfloor \alpha \rfloor x \leq \alpha x$.

- Általában

$$\lfloor \alpha_1 \rfloor x_1 + \lfloor \alpha_2 \rfloor x_2 + \ldots + \lfloor \alpha_n \rfloor x_n \leq \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \leq \beta.$$

Bizonyítás

- Valóban: $\lfloor \alpha \rfloor \leq \alpha$.

- Így nemnegatív x-re $\lfloor \alpha \rfloor x \leq \alpha x$.

- Általában

$$\lfloor \alpha_1 \rfloor x_1 + \lfloor \alpha_2 \rfloor x_2 + \ldots + \lfloor \alpha_n \rfloor x_n \leq \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \leq \beta.$$

- Ha ráadásul az x_i-k egészek, akkor a bal oldal egész. Így a β felső becslés javítható $\lfloor \beta \rfloor$ értékre.

Bizonyítás

- Valóban: $\lfloor \alpha \rfloor \leq \alpha$.

- Így nemnegatív x-re $\lfloor \alpha \rfloor x \leq \alpha x$.

- Általában

$$\lfloor \alpha_1 \rfloor x_1 + \lfloor \alpha_2 \rfloor x_2 + \ldots + \lfloor \alpha_n \rfloor x_n \leq \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \leq \beta.$$

- Ha ráadásul az x_i-k egészek, akkor a bal oldal egész. Így a β felső becslés javítható $\lfloor \beta \rfloor$ értékre.

- Kapjuk a bizonyítandót.

Példa (ALGEBRA)

Példa (ALGEBRA)

- A fenti egyszerű érvelésnek egyszerű geometriai szemléltetése van.

Példa (ALGEBRA)

- A fenti egyszerű érvelésnek egyszerű geometriai szemléltetése van.

- Vizsgáljuk a következő feladatot

Minimalizáljuk	$-2x - 5y$-t
Feltéve, hogy	$10x + 3y \leq 45$
	$4x + 20y \leq 65$
	$x, y \in \mathbb{N}$

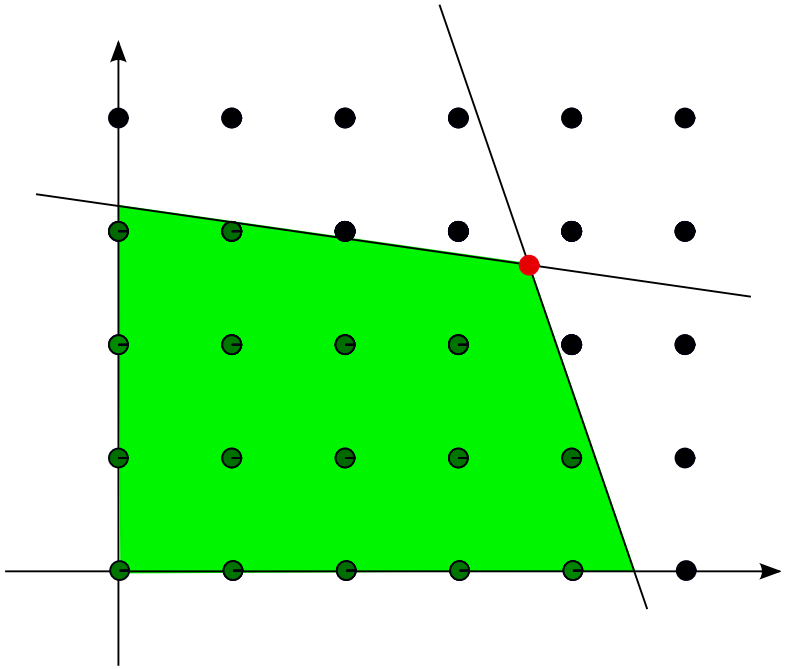
Példa (GEOMETRIA)

Példa (GEOMETRIA)

Az ábra a lehetséges megoldásokat mutatja.

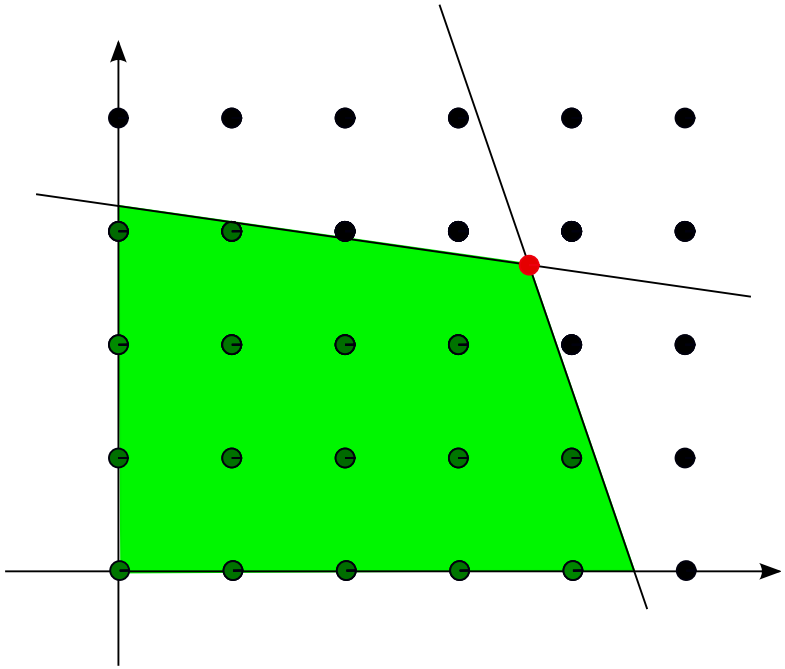
Példa (GEOMETRIA)

Az ábra a lehetséges megoldásokat mutatja.



Példa (GEOMETRIA)

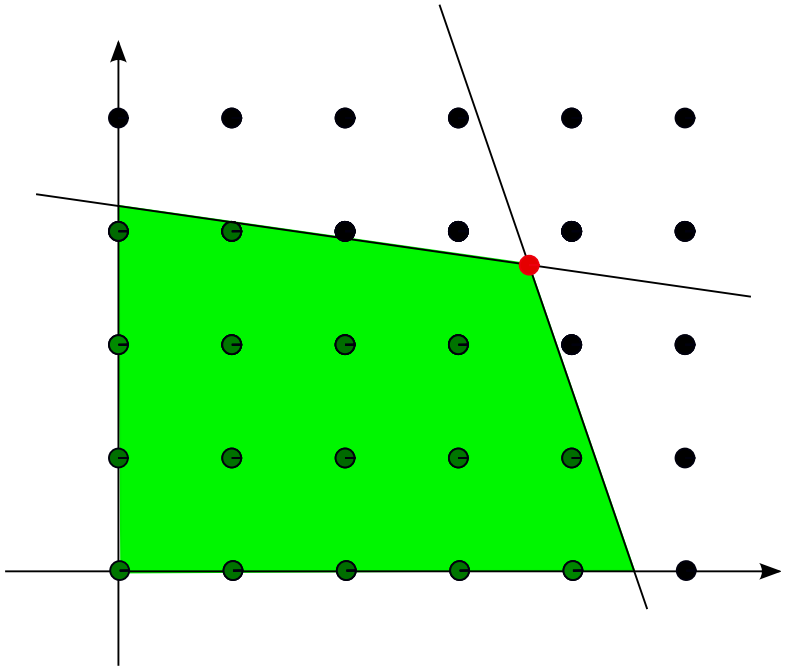
Az ábra a lehetséges megoldásokat mutatja.



A zöld tartomány az LP relaxáció politópja.

Példa (GEOMETRIA)

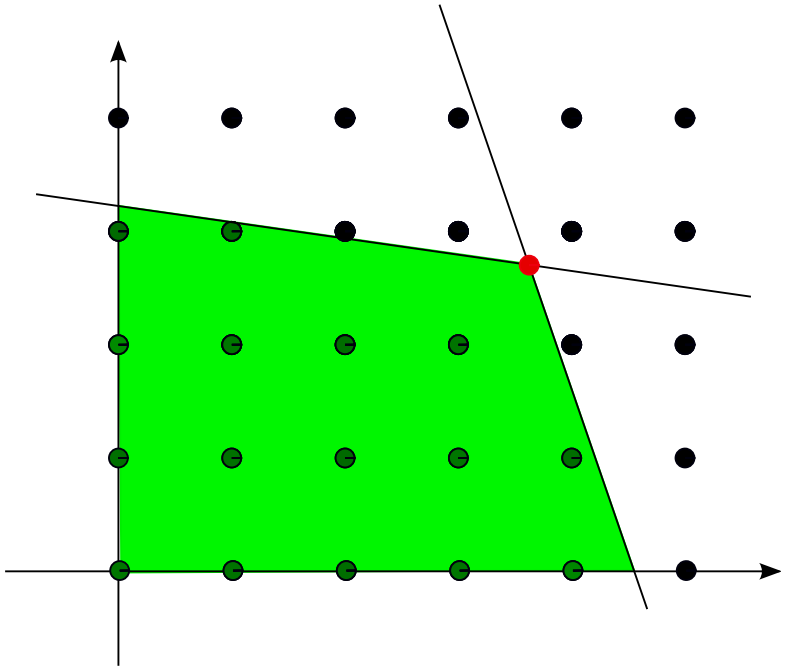
Az ábra a lehetséges megoldásokat mutatja.



A zöld tartomány az LP relaxáció politópja. A sötét zöld diszkrét ponthalmaz az egészértékű feladat lehetséges megoldásainak véges halmaza.

Példa (GEOMETRIA)

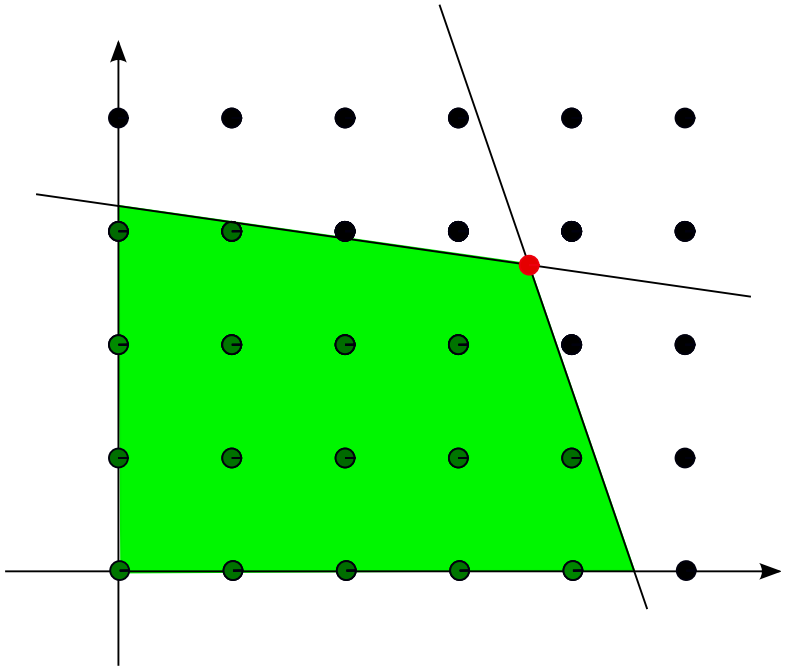
Az ábra a lehetséges megoldásokat mutatja.



A zöld tartomány az LP relaxáció politópja. A sötét zöld diszkrét ponthalmaz az egészértékű feladat lehetséges megoldásainak véges halmaza. Az LP feladat optimuma a piros pont:
$\left(\frac{15}{4}, \frac{5}{2}\right) = (3{,}75, 2{,}5)$,

Példa (GEOMETRIA)

Az ábra a lehetséges megoldásokat mutatja.



A zöld tartomány az LP relaxáció politópja. A sötét zöld diszkrét ponthalmaz az egészértékű feladat lehetséges megoldásainak véges halmaza. Az LP feladat optimuma a piros pont: $\left(\frac{15}{4}, \frac{5}{2}\right) = (3{,}75, 2{,}5)$, az egészértékű problémának nem lehetséges megoldása.

Példa (LOGIKA)

Példa (LOGIKA)

Felírünk néhány L-következtetést.

Példa (LOGIKA)

Felírünk néhány L-következtetést.

- Az első egyenlőtlenség $1/10$-del szorozva:

$$x + 0{,}3y \leq 4{,}5.$$

Példa (LOGIKA)

Felírünk néhány L-következtetést.

- Az első egyenlőtlenség $1/10$-del szorozva:

$$x + 0{,}3y \leq 4{,}5.$$

- A második egyenlőtlenség $1/4$-nel szorozva.

$$x + 5y \leq 16{,}25.$$

Példa (LOGIKA)

Felírünk néhány L-következtetést.

- Az első egyenlőtlenség $1/10$-del szorozva:

$$x + 0{,}3y \leq 4{,}5.$$

- A második egyenlőtlenség $1/4$-nel szorozva.

$$x + 5y \leq 16{,}25.$$

- Az első egyenlőtlenség kétszereséhez a másodikat adva, majd 24-gyel osztva kapjuk, hogy

$$x + 1\frac{1}{12}y \leq 6\frac{11}{24}.$$

Példa (LOGIKA)

Felírünk néhány L-következtetést.

- Az első egyenlőtlenség $1/10$-del szorozva:

$$x + 0{,}3y \leq 4{,}5.$$

- A második egyenlőtlenség $1/4$-nel szorozva.

$$x + 5y \leq 16{,}25.$$

- Az első egyenlőtlenség kétszereséhez a másodikat adva, majd 24-gyel osztva kapjuk, hogy

$$x + 1\frac{1}{12}y \leq 6\frac{11}{24}.$$

(GEOMETRIA: Mindhárom következtetés megoldáshalmaza egy olyan félsík, amely határoló egyenes áthalad a piros ponton (miért?).)

Példa (LOGIKA II)

Példa (LOGIKA II)

- Alkalmazzuk mindhárom L-következményre az I-logikát.

Példa (LOGIKA II)

- Alkalmazzuk mindhárom L-következményre az I-logikát.
- Az alábbi három egyenlőtlenséget kapjuk:

$$x \leq 4,$$

$$x + 5y \leq 16,$$

$$x + y \leq 6.$$

Példa (LOGIKA II)

- Alkalmazzuk mindhárom L-következményre az I-logikát.
- Az alábbi három egyenlőtlenséget kapjuk:

$$x \leq 4,$$

$$x + 5y \leq 16,$$

$$x + y \leq 6.$$

- A megoldáshalmazból „nem esett ki” egész koordinátájú pont.

Példa (GEOMETRIA II)

Példa (GEOMETRIA II)

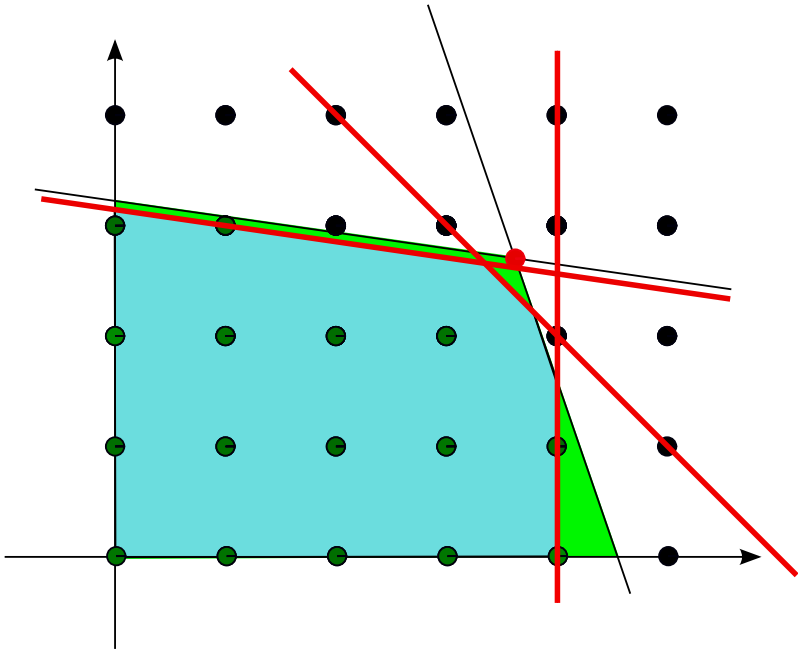
A fenti I-következmények által leírt félsíkok határolóegyeneseit piros színnel jelöltük.

Példa (GEOMETRIA II)

A fenti I-következmények által leírt félsíkok határolóegyeneseit piros színnel jelöltük. A következmények hozzáadása után kapott egyenlőtlenségrendszer relaxált megoldás halmaza a világos kék halmaz.

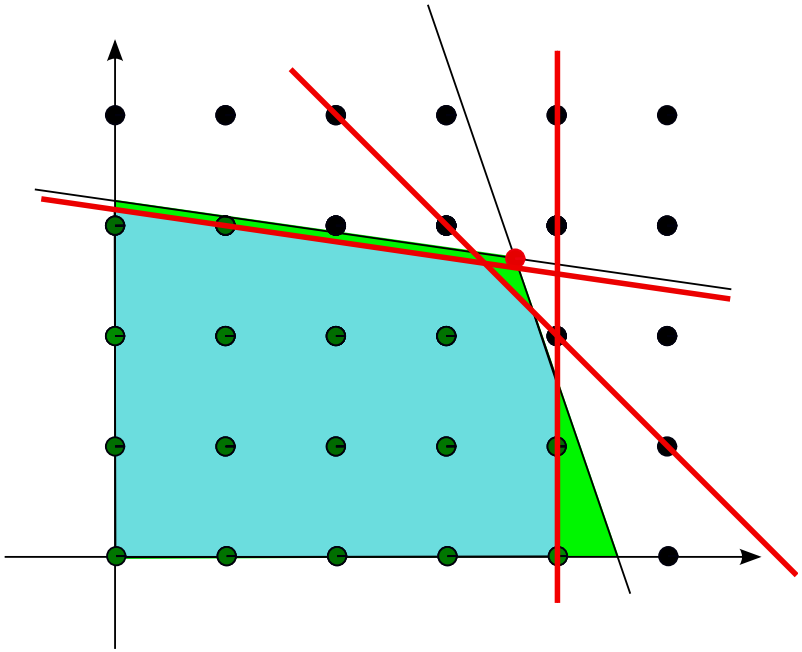
Példa (GEOMETRIA II)

A fenti I-következmények által leírt félsíkok határolóegyeneseit piros színnel jelöltük. A következmények hozzáadása után kapott egyenlőtlenségrendszer relaxált megoldás halmaza a világos kék halmaz.



Példa (GEOMETRIA II)

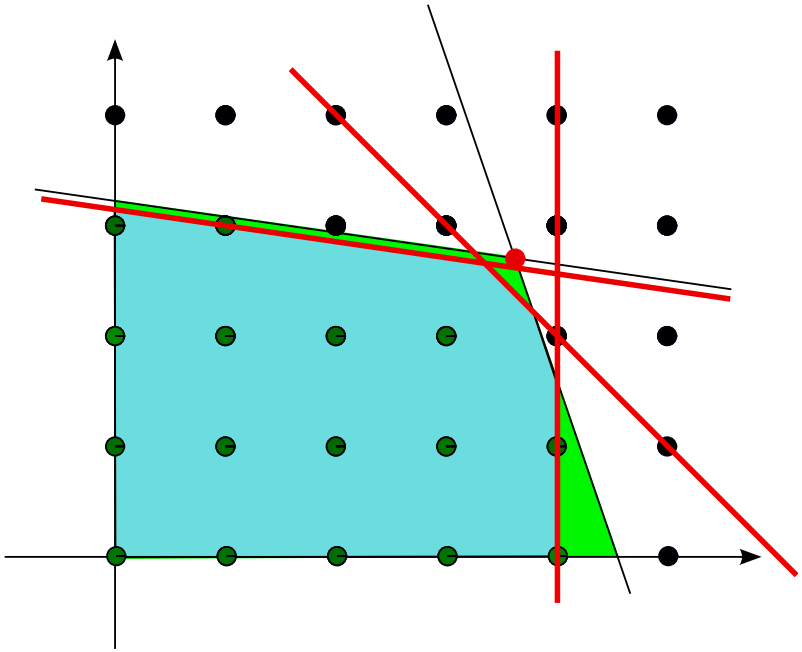
A fenti I-következmények által leírt félsíkok határolóegyeneseit piros színnel jelöltük. A következmények hozzáadása után kapott egyenlőtlenségrendszer relaxált megoldás halmaza a világos kék halmaz.



A leírt politóp csökkenése nyilvánvaló.

Példa (GEOMETRIA II)

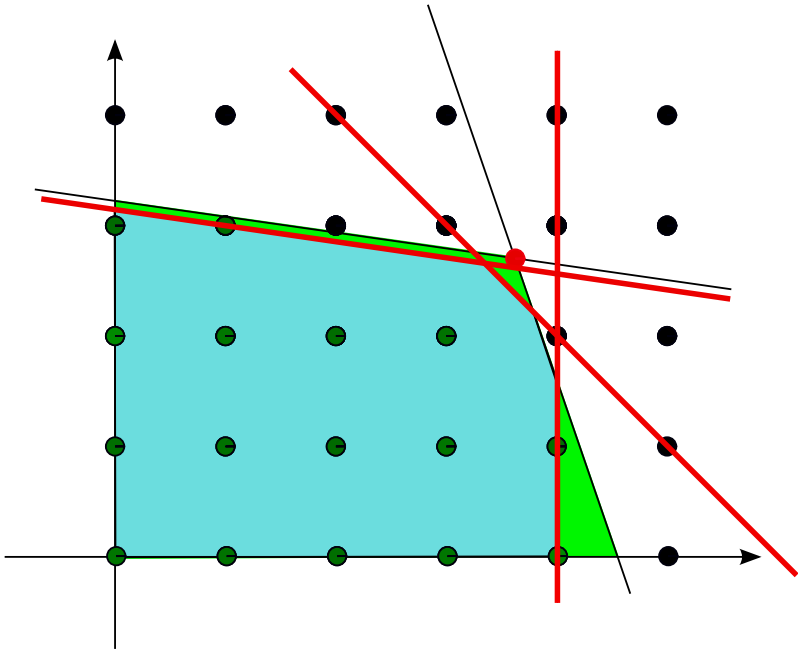
A fenti I-következmények által leírt félsíkok határolóegyeneseit piros színnel jelöltük. A következmények hozzáadása után kapott egyenlőtlenségrendszer relaxált megoldás halmaza a világos kék halmaz.



A leírt politóp csökkenése nyilvánvaló. A csökkenés során a sötét zöld pontok nem kerültek ki a megoldás halmazból.

Példa (GEOMETRIA II)

A fenti I-következmények által leírt félsíkok határolóegyeneseit piros színnel jelöltük. A következmények hozzáadása után kapott egyenlőtlenségrendszer relaxált megoldás halmaza a világos kék halmaz.



A leírt politóp csökkenése nyilvánvaló. A csökkenés során a sötét zöld pontok nem kerültek ki a megoldás halmazból. Az LP relaxáció által leírt politóp közeledett a sötét zöld pontok konvex burkához.

Általános séma

Általános séma

Ezek után egy lehetséges séma a kitűzött IP probléma megoldására a következő

Algoritmus

Általános séma

Ezek után egy lehetséges séma a kitűzött IP probléma megoldására a következő

Algoritmus

(R) // RELAXÁCIÓS lépés

Általános séma

Ezek után egy lehetséges séma a kitűzött IP probléma megoldására a következő

Algoritmus

(R) // RELAXÁCIÓS lépés
Oldjuk meg az IP feladat LP relaxációját.

Általános séma

Ezek után egy lehetséges séma a kitűzött IP probléma megoldására a következő

Algoritmus

(R) // RELAXÁCIÓS lépés
Oldjuk meg az IP feladat LP relaxációját.

(Sz) // SZERENCSE

Általános séma

Ezek után egy lehetséges séma a kitűzött IP probléma megoldására a következő

Algoritmus

(R) // RELAXÁCIÓS lépés
Oldjuk meg az IP feladat LP relaxációját.

(Sz) // SZERENCSE
Ha egész koordinátájú optimum helyet kapunk, akkor megoldottuk feladatunkat.

Általános séma

Ezek után egy lehetséges séma a kitűzött IP probléma megoldására a következő

Algoritmus

(R) // RELAXÁCIÓS lépés
Oldjuk meg az IP feladat LP relaxációját.

(Sz) // SZERENCSE
Ha egész koordinátájú optimum helyet kapunk, akkor megoldottuk feladatunkat.

(L) // LOGIKA

Általános séma

Ezek után egy lehetséges séma a kitűzött IP probléma megoldására a következő

Algoritmus

(R) // RELAXÁCIÓS lépés
Oldjuk meg az IP feladat LP relaxációját.

(Sz) // SZERENCSE
Ha egész koordinátájú optimum helyet kapunk, akkor megoldottuk feladatunkat.

(L) // LOGIKA
L- és I-következtetésekkel adjunk feladatunkhoz új lineáris egyenlőtlenségeket. Térjünk vissza a ralaxációs lépéshez.

Gomory-algoritmus

Gomory-algoritmus

- A fent leírt eljárás csak egy séma.

Gomory-algoritmus

- A fent leírt eljárás csak egy séma. Lényegi része az általános lépésben rejlik.

Gomory-algoritmus

• A fent leírt eljárás csak egy séma. Lényegi része az általános lépésben rejlik. Hogyan válasszuk a következtetéseket? Sok tisztázandó kérdés.

Gomory-algoritmus

- A fent leírt eljárás csak egy séma. Lényegi része az általános lépésben rejlik. Hogyan válasszuk a következtetéseket? Sok tisztázandó kérdés.
- Sok probálkozás/megoldás, sok viszonylag nagy speciális problémát kezelő algoritmus.

Gomory-algoritmus

• A fent leírt eljárás csak egy séma. Lényegi része az általános lépésben rejlik. Hogyan válasszuk a következtetéseket? Sok tisztázandó kérdés.

• Sok probálkozás/megoldás, sok viszonylag nagy speciális problémát kezelő algoritmus.

• Gomory adott egy eljárást, ahol belátta, hogy az algoritmusa véges sok lépésben megtalálja az egész optimumot.

Gomory-algoritmus

• A fent leírt eljárás csak egy séma. Lényegi része az általános lépésben rejlik. Hogyan válasszuk a következtetéseket? Sok tisztázandó kérdés.

• Sok probálkozás/megoldás, sok viszonylag nagy speciális problémát kezelő algoritmus.

• Gomory adott egy eljárást, ahol belátta, hogy az algoritmusa véges sok lépésben megtalálja az egész optimumot.

• Az új egyenlőtlenségek generálásához a szimplex módszert használja az eljárás.

Gomory-algoritmus

• A fent leírt eljárás csak egy séma. Lényegi része az általános lépésben rejlik. Hogyan válasszuk a következtetéseket? Sok tisztázandó kérdés.

• Sok probálkozás/megoldás, sok viszonylag nagy speciális problémát kezelő algoritmus.

• Gomory adott egy eljárást, ahol belátta, hogy az algoritmusa véges sok lépésben megtalálja az egész optimumot.

• Az új egyenlőtlenségek generálásához a szimplex módszert használja az eljárás. Az algoritmus leírására nincs időnk.

Gomory-algoritmus

• A fent leírt eljárás csak egy séma. Lényegi része az általános lépésben rejlik. Hogyan válasszuk a következtetéseket? Sok tisztázandó kérdés.

• Sok probálkozás/megoldás, sok viszonylag nagy speciális problémát kezelő algoritmus.

• Gomory adott egy eljárást, ahol belátta, hogy az algoritmusa véges sok lépésben megtalálja az egész optimumot.

• Az új egyenlőtlenségek generálásához a szimplex módszert használja az eljárás. Az algoritmus leírására nincs időnk.

• Már az Edmonds-féle poliéder tételnél láttuk, hogy az IP feladat LP leírásához lehet, hogy exponenciálisan sok egyenlőtlenségre van szükségünk.

Gomory-algoritmus

• A fent leírt eljárás csak egy séma. Lényegi része az általános lépésben rejlik. Hogyan válasszuk a következtetéseket? Sok tisztázandó kérdés.

• Sok probálkozás/megoldás, sok viszonylag nagy speciális problémát kezelő algoritmus.

• Gomory adott egy eljárást, ahol belátta, hogy az algoritmusa véges sok lépésben megtalálja az egész optimumot.

• Az új egyenlőtlenségek generálásához a szimplex módszert használja az eljárás. Az algoritmus leírására nincs időnk.

• Már az Edmonds-féle poliéder tételnél láttuk, hogy az IP feladat LP leírásához lehet, hogy exponenciálisan sok egyenlőtlenségre van szükségünk.

• A Gomory-algoritmusnál is ez a helyzet, általában nem polinomiális idejű.

Vége van!

Köszönöm a figyelmet!