ALGORITMUSOK ÉS BONYOLULTSÁGELMÉLET

Matematika MSc hallgatók számára

7. Előadás: Hálózatok, $\mathcal{P}$- és $\mathcal{NP}$-teljes problémák

Előadó: Hajnal Péter 2015. tavasz

1. Hálózatok és egy $\mathcal{P}$-teljes probléma

Emlékeztető. $L \in_T \mathcal{TIME}(t(n))/\mathcal{NTIME}(t(n))$ esetén mindig feltesszük, hogy $t(n)$ szép idő függvény, azaz Turing-géppel megvalósítható egy óra, ami n hosszú inputra „$t(n)$ idő után üt".

$\omega \in \Sigma^n$ inputon T futása a

$$\kappa_0(\omega) \to \kappa_1 \to \kappa_2 \to \ldots \to \kappa_\ell,$$

konfigurációsorozat, ahol $\kappa_0(\omega)$ az ω-hoz tartozó kiinduló konfiguráció, κ_{i+1} a κ_i konfiguráció rákövetkezője, és κ_ℓ az első olyan konfiguráció, ahol az állapot ELFOGAD vagy ELVET. Feltehető, hogy $\ell = t(n)$.

Észrevétel 1. κ konfigurációk kódolhatók bitsorozatokkal.

A fenti ábra (megjegyzéseivel) magáért beszél. A fej pozícióját is kell kódolnunk. Ezt „szétszórtuk", minden mező raktunk egy bitet. Így nem miden adott hosszúságú sorozat kódol konfigurációt. Az állapotot és karaktereket kódoló blokkok hossza függ $|S|$, $|\Sigma|$ és $|\Gamma|$ értékétől. Minden esetre konstans sok bit szükséges (ami a Turing-géptől függ). S elemeinek kódolására $\lceil \log_2 |S| \rceil$ bit legendő. Ha $|S|$ nem kettőhatvány, akkor lesznek olyan 0-1 bitsorozatok, amik a megfelelő pozíciókban állnak, de nem kódolnak állapotot. Ennek ellenére könnyű tervezni egy olyan tesztelő hálózatot, ami egy adott (megfelelő hosszú) kódról eldönti, hogy konfigurációt kódole.

Megjegyzés (FONTOS). A megállapodát úgy választhatjuk adott n hosszú ω input esetén $\lceil \omega \rceil$ hossza $\alpha_T \cdot n$ legyen. Ha T időigénye legfeljebb $t(n)$, akkor konfigurációk kódjának hossza $\beta_T \cdot t(n)$ legyen. (A konfigurációban a munkaszalag elvágható az első $t(n)$ mező után. Az elhagyott/levágott rész csak érintetlen mezőket tartalmaz.)

A továbbiakban n és a kódolási megállapodás mindig rögzített (ennek megfelelően a megfelelő kódok hossza mindig ismert).

Észrevétel 2. $\lceil \kappa_i \rceil \to \lceil \kappa_i^+ \rceil = \lceil \kappa_{i+1} \rceil$ egyszerű hozzárendelés/függvény.

Az észrevétel állítása matematikailag nem pontos, az „egyszerű" jelző értelmezése nem jól definiált.

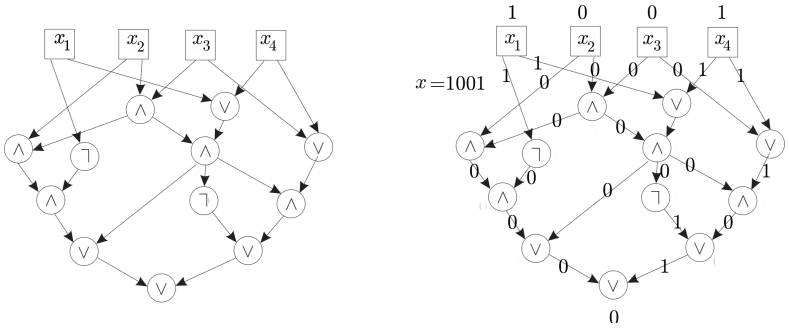
Azt mutatjuk, hogy egyszerűen meghatározható/kiszámolható egy kicsi (polinomiális méretű) hálózat, amely egy konfiguráció kódjából kiszámolja a rákövetkező konfiguráció kódját. Először azonban lássuk a szükséges fogalmakat.

Definíció. Egy hálózat egy $\overrightarrow{G}$ irányított gráf, amely nem tartalmaz irányított kört (azaz lerajzolható úgy, hogy minden él „lefelé haladjon"). Minden csúcs befoka $0, 1$ vagy 2. A 0 befokú csúcsok neve inputcsúcs. Legye I az input csúcsok halmaza. A nem inputcsúcsokra mint kapuk hívatkozunk. A kapuk halmaza $K = V(\overrightarrow{G}) - I$. Speciális csúcsot, esetleg csúcsokat nevezünk ki, amelykre mint output csúcsok hívatkozunk.

Legyen $\ell_I : I \to \{x_1, x_2, \ldots, x_n\} \cup \{0, 1\}$ egy címkézéese az output csúcsoknak. Legyen $\ell_K : K \to \{\neg, \vee, \wedge\}$ egy címkézése a kapuknak, amelyre teljesül, hogy egy kapunknak akkor és csak akkor $\neg$ a címkéje, ha befoka 1. Legyen $\ell = (\ell_I, \ell_K)$ az összes csúcsot címkéző függvény.

$(\overrightarrow{G}, \ell)$ címkézett irányított gráfot hálózatnak nevezzük.

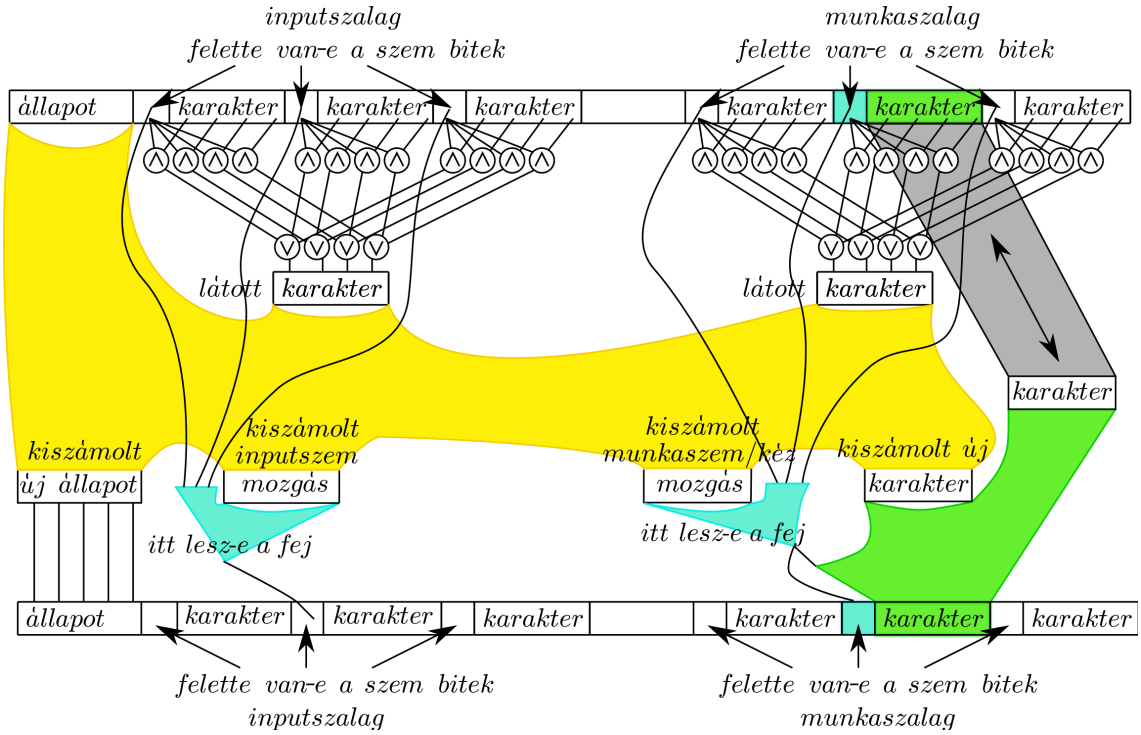
A fenti egy statikus számítási modell. Lássuk a fenti modell dinamikus változatát. Egy hálózat kiszámol egy Boole-függvényt a következő módon. Az $\{x_1, x_2, \ldots, x_n\}$ változók egy kiértékelését kiterjesztjük a kapuk egy kiértékeléséve: Feltesszük, hogy hálózatunkat úgy rajzoljuk le, hogy minden él lefelé haladjon. A kapuk kiértékelése fentről lefelé halad.



Ha egy kapuhoz jutunk, akkor azok a kapuk, ahonnan él vezet hozzá már kiértékeltek, azaz egy kiszámolt bitet rendeltünk hozzá. Az aktuális kapu által kiszámolt bit az hozzá vezető éleken áramló bit és a kapu címkéje alapján természetesen értelmezhető. A hálózat által kiszámolt bitsorozat az otuput csúcs(ok) által kiszámolt bit(ek).

Azaz a $\mathcal{C}$ hálózat egy $f_{\mathcal{C}}$ Boole-függvényt számol ki/valósít meg.

A fentiek „matematizálása": Egy konfigurációt kódoló bitsorozatból egy egyszerűen leírható, kicsi hálózat kiszámolja a rákövetkező konfiguráció kódját. A konstrukciónk egyszerű, de sok esetlegességet, megállapodást követel. Egy formális leírás helyett egy példán szemléltetjük milyen ötletek vezethetnek el egy megoldáshoz.



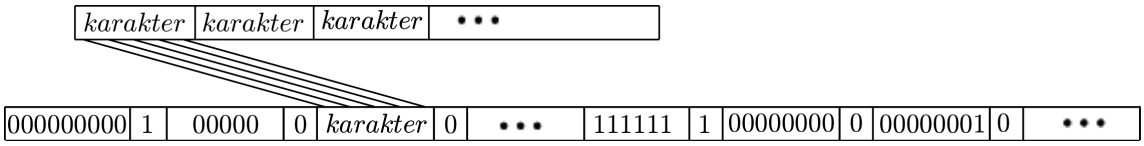
Egy mezőnél a szem/kéz-ott-van-e bitet és a mező tartalmát kódoló bitek mindegyikét össze-és-eljük. A kapott bitsorozat vagy a csupa 0 (ha nincs ott a szem/kéz, mindent 0-val és-eltünk) vagy a látott karakter kódja (ha ott a szem/kéz). Az inputszalag mezőinél így kapott bitsorozatok mindegyikének első, majd második és így tovább karakterét össze-vagy-olva kapjuk az inputszalagon látott karakter kódját. Hasonlóan számolhatunk a munkaszalagnál.

A sárga területben egy összetettebb számolás történik: konstans sok bitből számolunk ki konstans sok bitet (a konstanok függnek a Turing-géptől). Konkrét kidolgozása az átmenetifüggvénytől függ. Nem okozhat problémát akkor sem, ha az egyes bitek függését a nyilvánvaló DNF formula alapján írjuk fel. Ekkor is konstans sok kapuval dolgozva megoldjuk a feladatunkat.

A világos kék területen számolódik kis a fej pozícióját leíró egyik bit. Ez attól függ, hogy a fej ott volt-e vagy valamelyik szomszéd felett állt, illetve merre írja elő mozgását az átmeneti függvény. A hálózatunk ezen részét konkrétan felírhatnánk, de ez a munka felesleges az előző megjegyzésünk alapján. Ez a kék rész minden fej-pozíció-bithez ott van. Az átláthatóság kedvéért rajzoltuk fel csak egyszer az input- és egyszer A munkaszalaghoz. A zöld területen számoljuk ki a munkaszalag új tartalmát. Minden munkaszalag mezőhöz tartozik egy ilyen zöld blokk (egyszerűség kedvéért csak egyet tüntettünk fel). Az új karaktertől, a régitől és attól az információtól függ, hogy a fej ott áll-e. Ez a rész is könnyen megvalósítható lenne, ha tudnánk Γ elemeinek kódolásához használt bitek ℓ számát ismernénk. A zöld területen az $f(\epsilon, k_0, k_1) = k_\epsilon$ függvényt számoljuk ki, ami $1 + 2\ell$ bitből számol ki ℓ-et.

Észrevétel 3. $\lceil\omega\rceil \to \lceil\kappa_0(\omega)\rceil$ egy egyszerű hozzárendelés/függvény.

Ez a korábbiaknál egyszerűbb észrevétel. Ismét egy ábrára utalunk.



Feltettük, hogy a START állapot kódja $00\ldots0$ (hossza $\lceil \log_2 |S| \rceil$, példánkban 9). Az inputszalagon $\triangleright$ kódja $00\ldots0$, a $\triangleleft$ kódja $11\ldots1$ (hosszaik $\lceil \log_2 |\Sigma| \rceil$, példánkban 5). Az munkaszalagon $\triangleright$ kódja $0\ldots00$, a szűzkarakter kódja $0\ldots01$, (hossza $\lceil \log_2 |\Gamma| \rceil$, példánkban 8).

Először definiálunk egy problémát/nyelvet, majd összefoglaljuk a fenti megállapításainkat egy tételben.

Definíció. Legyen

$$\text{HÁLÓZAT-KIÉRTÉLÉS} = \{\lceil \mathcal{C}, \omega \rceil : \mathcal{C}(\omega) = 1\},$$

azaz az a döntési probléma, amely egy $\mathcal{C}$ hálózat és ω bitsorozat esetén eldönti, hogy az ω bitsorzatot adva az $x_1, x_2, \ldots$ inputkapuk értékeinek a hálózat az 1 bitet számolja-e ki (azaz kiértékeli $\mathcal{C}_n(\omega)$-t).

1. Tétel. *HÁLÓZAT-KIÉRTÉLÉS $\mathcal{P}$-teljes (az $\mathcal{L}$ redukcióra nézve).*

Bizonyítás. Legyen $L \in_T \mathcal{P}$. Legyen $t(n)$ egy polinom, T időkorlátja. Feltehető, hogy T olyan, hogy ELFOGAD/ELVET állapot elérése után „tartja" állapotát. Így $\omega \in L$? kérdés megválaszolása ($\omega \in \Sigma^n$) ekvivalens annak megállapításával, hogy az ω- történő futás során a $\kappa_{t(n)}$ konfigurációban az állapot ELFOGAD-e. Megállapodhatunk abban, hogy a konfigurációkat 0-1 sorozatokkal kódoljuk, hogy az állapotot kódoló blokkban az ELFOGAD állapot kódja a csupa 1 sorozat legyen.

A fentiek alapján bármi volt is $L \in_T \mathcal{P}$ $\omega \in \Sigma^n$ tetszőleges eleméhez felépíthető egy $\mathcal{T}_{T,\omega}$ hálózat, amely inputkapui ω-t kódolják (3. észrevétel) és bizonyos szintje a Turing-számítás konfiguráció-sorozatának elemeit kódolja (2. észrevétel). $t(n)$ ilyen szint felépítése után az érdekel minket, hogy egy bizonyos blokkban csupa 1-es bit szerepel-e. Ezt ÉS kapuk segítségével könnyen kifejezhetjük.

Ezzel leírtuk a redukáló algoritmust. A redukció elméleti része a korábbiakból adódik. A konstrukció/redukció logtár-beli. ■

2. Egy $\mathcal{NP}$-teljes nyelv

Definíció. Legyen

$$\text{HÁLÓZAT-SAT} = \{\lceil \mathcal{C} \rceil : \text{ van olyan } \omega \text{ bitsorozat, amelyre } \mathcal{C}(\omega) = 1\},$$

azaz az a döntési probléma, amely egy $\mathcal{C}$ hálózat esetén el kell döntenünk, hogy kielégíthető-e.

2. Következmény.

- *(i) Tetszőleges $L \in \mathcal{NP}$ nyelvre $L \prec_{\mathcal{L}}$ HÁLÓZAT-SAT*
- *(ii) HÁLÓZAT-SAT $\mathcal{NP}$-teljes*
- *(iii) $\mathcal{P} = \mathcal{NP} \Leftrightarrow$ HÁLÓZAT-SAT $\in \mathcal{P}$*

Bizonyítás. (i) $L \in_T \mathcal{NP}$, így egy tetszőleges $\omega \in L$ esetén létezik hozzá egy tanú: $\tau = (t_1, t_2, \ldots, t_{p(n)})$, amelyre $T(\omega, \tau)$ ELFOGAD állapotba jut. Azaz az előbb megkonstruált C hálózatra, $C(\lceil\omega\rceil, y_1, y_2, \ldots, y_{q(n)})$ az 1 értéket számolja ki, ha az y változók helyére a τ kódjának bitjeit írjuk. Megfordítva is igaz. Ha $C(\lceil\omega\rceil, y_1, y_2, \ldots, y_{q(n)})$-nek találunk egy kielégítését, akkor egy tanú kódját találjuk. Azaz $C(\lceil\omega\rceil, y_1, y_2, \ldots, y_{q(n)})$ kódjának legyártása (ami az emlékeztető alapján $\mathcal{L}$-ben megoldható) egy jó redukció.

(ii) Az (i) részből és abból, hogy HÁLÓZAT-SAT$\in \mathcal{NP}$ (tanú egy kielégítő bemenet), következik, hogy HÁLÓZAT-SAT $\mathcal{NP}$-teljes.

(iii) Mivel HÁLÓZAT-SAT $\in \mathcal{NP}$, így ha $\mathcal{P} = \mathcal{NP}$, akkor $\mathcal{P}$-beli is. Viszafelé, ha HÁLÓZAT-SAT$\in \mathcal{P}$, akkor tetszőleges $L \in \mathcal{NP}$ nyelvet redukáljunk HÁLÓZAT-SAT-ra, a redukált problémát $\mathcal{P}$-ben el tudjuk dönteni. A két lépés együtt is polinomiális és az L nyelv eldöntési problémáját oldja meg. Ebből $L \in \mathcal{P}$ következik, így $\mathcal{NP} \subseteq \mathcal{P}$, tehát $\mathcal{P} = \mathcal{NP}$ adódik. ■

A továbbiakban a legklasszikusabb $\mathcal{NP}$-teljes nyelvet imsertetjük.

3. Cook—Levin-tétel

Definíció. Legyen $V = \{x_1, x_2, \ldots, x_n\}$ egy változó halmaz. Legyen $L = V \dot\cup \overline{V}$ a literálok halmaza ($\overline{V}$ a negált változók halmaza, azaz $\{\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}$). L egy részhalmazát klóznak nevezzük. Esetünkben a klózra úgy gondolunk, hogy a hozzá tartozó literálokat $\vee$ logikai művelettel, azaz diszjunkcióval kapcsoljuk össze. Egy φ konjunktív normálformában lévő formula (CNF formula) klózok egy halmaza. Erre a klózhalmazra úgy gondolunk, hogy a klózok $\wedge$ logikai művelettel, azaz konjukcióval kapcsoljuk össze. Egy φ CNF formula kielégíthető, ha adható V egy kiértékelése (ami természetes módon kiterjeszthető L egy kiértékelésévé), amelyre minden klózban lesz igazzá kiértékelt literál.

$$\text{SAT} = \{\lceil\varphi\rceil : \varphi \text{ kielégíthető CNF}\}$$

Azaz SAT az a probléma, ahol adott egy CNF formula és el kel döntenünk, hogy kielégíthető-e.

3. Tétel (Cook—Levin-tétel). *SAT (CNF formula kielégíthetősége) $\mathcal{NP}$-teljes.*

Bizonyítás. Korábban már szerepelt, hogy SAT $\in \mathcal{NP}$, így elég adnunk egy visszavezetést HÁLÓZAT-SAT-ról SAT-ra, hiszen ekkor az előző következmény (i) pontja és a polinomiálsi redukcio tranzitívitása alapján tetszőleges $L \in \mathcal{NP}$ nyelvet vissza tudunk vezetni SAT-ra. Ezt is két lépésben tesszük, a HÁLÓZAT-SAT-ot visszavezetjük BOOLE-EGYENLETRENDSZER-SAT-ra, majd azt SAT-ra.

Definíció. $\varphi_i(x_1, x_2, \ldots, x_n) = \psi_i(x_1, x_2, \ldots, x_n)$, $i = 1, 2, \ldots, \ell$ egyenletrendszert, Boole-egyenletrendszernek nevezzük, ha φ_i és ψ_i Boole-formulák. Az $\{x_i\}_{i=1}^n$ változók egy 0-1/igaz-hamis értékadás az egyeneletrendszer megoldása, ha minden $i = 1, 2, \ldots, \ell$ esetén φ_i és ψ_i értéke ugyanaz.

BOOLE-EGYENLETRENDSZER-SAT az a nyelv, ami a megoldható/kielégíthető Boole-egyenletrendszerek kódját tartalmazza.

Legyen H egy hálózat. Minden csúcsával azonosítunk egy változót. Ez input-csúcsok esetén a csúcs címkéje. A többi csúcsra (kapukra) mind különváltozókat feleltetünk meg. Minden kapuhoz tartozik egy egyenlet

- $x_g = \neg x_h$, ha a g kapu negáció kapu és a h kapuból kapja inputját ($\overrightarrow{hg}$ él a hálózatban).
- $x_g = x_h \wedge x_{h'}$, ha a g kapu hálózatbeli címkéje konjunkció és inputjait h és h' kapukból kapja.
- $x_g = x_h \vee x_{h'}$, ha a g kapu hálózatbeli címkéje diszjunkció és inputjait h és h' kapukból kapja.
- $x_g = 1$, ha a g kapu a hálózat output kapuja.

Ezzel megkaptuk egy Boole-egyenletrendszert a hálózatból. Ha a hálózat 1-et számol ki egy értékadáson (az értékadás kielégíthetőséget bizonyít), akkor a kapuk által kiszámolt bitekkel együtt egy megoldását kapjuk az egyeneletrendszernek. Fordítva is igaz, ha az egyenletrendszer megoldásából kiemeljük az eredeti input változók értékadásait, akkor ezen a hálózat 1-et számol ki (sőt minden kapu a hozzárendelt változó megoldásbeli értékét számolja ki). Azaz a hálózatból legyártott egyenletrendszer megoldhatósága ekvivalens azzal, hogy a hálózat kielégíthető.

Az '=' jeleket '$\leftrightarrow$' logikai jelekre cserélve az egyes egyenleteknek megfelelő formulákat kapunk. Egy értékadás akkor és csak akkor teljesíti az egyenletet, ha igazzá teszi a hozzárendelt formulát. A kapott logikai kifejezések mindegyike legfeljebb három változót tartalmaznak. Könnyű őket CNF formára hozni. Ha az összes egyenletnek megfeleltetett CNF formulát 'és' logikai jellel összekapcsoljuk, szintén CNF formát kapunk. Így az egyenletrendszerhez hozzárendeltünk egy φ CNF formulát. A hozzárendelés $\mathcal{P}$-ben kiszámolható. Az egyenletrendszer megoldhatósága ekvivalens a formula kielégíthetőségével.

Vagyis „programunk" második redukcióját is megadtuk, a tételt bebizonyítottuk. ■