8.

Az NP nyelvosztály

Az ember s az idő mindig összetalálnak,
mint a keserűlapi saját árnyékával.
TAMÁSI ÁRON

Ebben a fejezetben először algoritmusok időbeli és tárhasználat szerinti hatékonyságát vizsgáljuk. Definiálunk néhány olyan jellemzőt, melyek lehetővé teszik a hatékonyság pontosabb értelmezését. E fogalmak segítségével megadható a (legalábbis elméleti értelemben) hatékonyan kezelhető algoritmikus problémák köre (P és FP osztályok). A legtöbb eddig tárgyalt feladat (nyelv, függvény) ezen osztályok egyikébe tartozik. Mint látni fogjuk, sok gyakorlati szempontból fontos feladat – a mai tudásunk szerint – kívül esik ezeken az osztályokon, vagyis nem oldható meg hatékony algoritmusokkal. E nehezebb feladatok közül kiemelkedő fontosságúak azok, amelyek az NP osztályba tartoznak. Itt most csak azt emeljük ki, hogy nagyon sok fontos algoritmikus feladat vezet NP-beli problémához. Az NP-beli nyelvekkel kapcsolatban első közelítésként a bevezető példáink egyikére, Arthur király kérésére utalunk. Képzeljük el, hogy a király csak annyit kérdez Merlintől, hogy párokba állíthatók-e a lovagok és az udvarhölgyek úgy, hogy a vonzalmakat is figyelembe vegyük. Merlin a válasz mellé egyszerűen ellenőrizhető bizonyítékot mellékelhet: a megfelelő párok listáját. Ennek birtokában a király hatékonyan ellenőrizheti a válasz helyességét. Az NP-beli nyelvek éppen ezzel a tulajdonsággal jellemezhetők. Az *igen* válasznak van gyorsan ellenőrizhető bizonyítéka.

Külön figyelmet fogunk szentelni az NP osztály legnehezebb feladatainak, az NP-teljes feladatoknak. Az NP-osztályra úgy gondolhatunk mint a természetesen felmerülő nyelvek nagy gyűjtőhelyére; az alján vannak a hatékonyan, gyorsan felismerhető nyelvek (a P osztály), a tetején pedig a nehéz, gyors algoritmussal nem megoldható problémák, az ún. NP-teljes nyelvek. Utóbbiakkal azért foglalko-

zunk behatóbban, mert szinte minden alkalmazási területen előfordulnak. Az NP-teljességgel kapcsolatos ismeretek gyakran segítenek annak megítélésében, hogy egy elénk kerülő algoritmikus feladat megoldására remélhetünk-e gyors módszert.

8.1. Idő- és tárkorlátok

Kétmilliárd férfi húszezer évet borotválkozik naponta. GARACZI LÁSZLÓ

Szeretnénk viszonylag pontos fogalmakat adni arra, hogy egy algoritmus gyors, illetve hogy hatékonyan bánik a tárral. Ezt úgy tesszük, hogy korlátozzuk az algoritmus (Turing-gép) számolási idejét vagy a felhasznált tárcellák számát. Nyilvánvalóan nem lenne jó abszolút, a bemenettől független korlátokat bevezetni, hiszen természetesnek tartjuk, hogy hosszabb inputon egy módszer több időt/tárat használ; már pusztán a bemenet elolvasása, értelmezése több munkát jelent. A két törekvés összebékítésének egy lehetséges módja, hogy az idő- és tárfelhasználást az *input hosszának függvényében* vizsgáljuk. Legyen $t : \mathbb{Z}^+ \to \mathbb{Z}^+$ egy függvény, melyre minden $n \in \mathbb{Z}^+$ esetén $t(n) \geq n$ teljesül.

Definíció ($t(n)$ időkorlátos TG):
Az M Turing-gép $t(n)$ időkorlátos, ha n hosszú inputokon legfeljebb $t(n)$ lépést tesz (más szóval $T_M(n) \leq t(n)$).

A $t(n)$ függvény tehát korlátozást ír elő az M gép lépéseinek a számára. A $t(n) \geq n$ azt a természetes feltevést tartalmazza, hogy a gép legalább végigolvashatja az inputot a megadott időkorláton belül maradva. Az M algoritmust (TG-t) akkor tekinthetjük gyorsnak, ha $t(n)$ egy lassan növekedő függvény.

A 7.1. pontbeli első példa, a hárommal való oszthatóságot ellenőrző M Turing-gép n hosszú inputokon legfeljebb $n+1$ lépést tesz, tehát mondhatjuk, hogy $n+1$ időkorlátos. A másik példa, az $f(n) := n+1$ függvényt kiszámító M gép nem lesz $t(n)$ időkorlátos semmilyen t függvénnyel, mert vannak olyan inputok, amiken a gép sohasem áll meg.

Feladat: Adjunk meg egy olyan $n + 1$ időkorlátos N Turing-gépet, melyre $L_N = L_M$ és $f_N = f_M$, ahol M a fenti TG.

A hatékony algoritmus fogalmának a birtokában megpróbálkozhatunk a feladatok hatékonyság szerinti osztályozásával. Azokat a feladatokat (nyelveket, függvényeket) tekintjük alacsony bonyolultságúnak, melyek megoldására létezik gyors algoritmus. Ezt készíti elő a következő fogalom.

Definíció:
$$TIME(t(n)) := \left\{ L \subseteq I^* \;\middle|\; \begin{array}{l} L \textit{ felismerhető egy } O(t(n)) \textit{ időkorlátos} \\ M \textit{ Turing-géppel} \end{array} \right\}.$$

A $TIME(t(n))$ nyelvosztályba tehát azok az L nyelvek tartoznak, amelyekhez létezik $ct(n)$ időkorlátos TG. A c állandó függhet L-től. A definíció lényeges eleme, hogy n hosszú x inputokon a számítás *mindig befejeződik* legfeljebb $ct(n)$ lépésben, tekintet nélkül arra, hogy $x \in L$ igaz-e. Ennek következményeként $TIME(t(n))$ *rekurzív nyelvekből áll.*

A legegyszerűbb példát a *lineáris időben* felismerhető nyelvek jelentik:

Példa: $TIME(n) = \{$*az* $O(n)$, *azaz lineáris időben felismerhető nyelvek*$\}$.

Algoritmikus időigény szempontjából a lineáris időben felismerhető nyelvek tekinthetők a legkönnyebbeknek. Szokásos még a $TIME(n^2)$ nyelvosztályt a négyzetes, $TIME(n^3)$-t pedig a köbös időben felismerhető nyelvek összességének nevezni. A következő nyelvosztály központi szerepet játszik a számítások elméletében:

Definíció: $\mathrm{P} = \cup_{k \geq 1} TIME(n^k)$, *a polinom időben felismerhető nyelvek osztálya.*

Példa: $L = \{0^n 1^n | \; n \geq 1\} \in TIME(n) \subseteq \mathrm{P}$. Könnyen szerkeszthető olyan kétszalagos Turing-gép, mely a fenti nyelvet lineáris időben ismeri fel.

Állítás: *Ha az L nyelv* $n^{\log n}$*-nél rövidebb időben nem ismerhető fel, akkor* $L \notin \mathrm{P}$.

Bizonyítás: Indirekt érveléssel tegyük fel, hogy $L \in \mathrm{P}$. Ekkor a P definíciója szerint van olyan $k > 0$, melyre $L \in TIME(n^k)$, és így alkalmas $c > 0$ konstansra $n^{\log n} \leq cn^k$ teljesülne végtelen sok n-re. Ez az egyenlőtlenség viszont ellentmondást jelent, hiszen a bal oldalon álló függvény gyorsabban nő, mint a jobboldali. □

Az időhöz hasonlóan kezelhetjük az algoritmusok tárfelhasználását és a nyelvek felismerésének tárigényét. Legyen $s : \mathbb{Z}^+ \to \mathbb{Z}^+$ olyan függvény, melyre minden $n \in \mathbb{Z}^+$ számmal igaz, hogy $s(n) \geq \log_2 n$.

Definíció ($s(n)$ tárkorlátos TG):
Az M Turing-gép $s(n)$ *tárkorlátos, ha n hosszú inputokon legfeljebb* $s(n)$ *tárcellát használ a* munkaszalagokon *(azaz* $S_M(n) \leq s(n)$*).*

Az $s(n) \geq \log_2 n$ kikötés enyhe és értelmes feltevés. Ennyi hely kell ugyanis ahhoz, hogy egy n cellából álló szalagrészt – például az input szalag érdemi részét – címezni tudjunk. Emlékeztetünk még itt arra, hogy ha M-nek csak egy szalagja van, akkor az a definíció szempontjából munkaszalagnak tekintendő. Az időosztályokkal analóg módon kapjuk a tárosztályok definícióját:

Definíció:

$$SPACE(s(n)) := \left\{ L \subseteq I^* \;\middle|\; \begin{array}{l} \textit{az } L \textit{ felismerhető egy } O(s(n)) \textit{ tárkorlátos} \\ M \textit{ Turing-géppel} \end{array} \right\}.$$

Példa: $SPACE(\log n)$ a *logaritmikus tárban felismerhető nyelvek* osztálya. Ha az L nyelv a $SPACE(\log n)$ osztályban van, akkor felismerhető egy olyan M TG-vel, ami n hosszú inputokon legfeljebb $c \log n$ tárcellát használ a munkaszalagjain. A c állandó itt is függhet L-től.

A nyelvekhez hasonló módon kaphatunk idő-, illetve tárkorlátokkal meghatározott függvényosztályokat. Csupán annyit kell tennünk, hogy a definíciókban a nyelveket felismerő TG-k helyett függvényeket kiszámoló TG-ket szerepeltetünk.

Definíció: $FTIME(t(n)) := \textit{az } O(t(n)) \textit{ időkorlátos TG-k által kiszámítható } f : I^* \to I^* \textit{ függvények osztálya.}$

Definíció: $FSPACE(s(n)) := \textit{az } O(s(n)) \textit{ tárkorlátos TG-k által kiszámítható } f : I^* \to I^* \textit{ (parciális) függvények osztálya.}$

Igen fontos osztály a P-vel analóg függvényosztály FP, a *polinom időben kiszámítható függvények osztálya*:

Definíció: $\mathrm{FP} := \cup_{k \geq 1} FTIME(n^k)$.

8.2. Tár–idő-tétel, nevezetes nyelvosztályok

Jóllehet a világot alkotó atomok száma mérhetetlenül nagy, nem végtelen, ezért csupán véges számú (még ha mérhetetlenül nagy is ez a szám) permutációt adhat. Ha végtelen az idő, a lehetséges permutációk száma egyszer kimeríthető, s akkor szükségszerűen megismétlődik a világegyetem. Ismét megszületsz majd egy anyaméhből, ismét kifejlődik a csontvázad, ismét ugyanebbe a kezedbe kerül ez a lap, ismét végigéled életed minden óráját a hihetetlen halál percéig. JORGE LUIS BORGES

(Így foglalja össze a nietzschei Örök Visszatérést.)

A következő állítás egy alapvető összefüggést rögzít a tár-, illetve időkorlátokkal definiált osztályok között. Ha egy nyelv felismerésére van tárkorlátos algoritmus, akkor van időkorlátos is. Az adódó időkorlát a tárkorlát exponenciális függvényével becsülhető. A gondolatmenet kezdőpontja az, hogy ha egy tárkorlátos számítás elég sokáig tart, akkor szükségképpen végtelen ciklusban van. A végtelen ciklusok felismerése, kezelése jelenti a probléma – és a bizonyításban szereplő konstrukció – érdemi részét.

Tétel (tár–idő-tétel): *Ha $L \in SPACE(s(n))$, akkor van olyan L-től függő c konstans, mellyel $L \in TIME(c^{s(n)})$ teljesül.*

Bizonyítás: Legyen M egy $S(n) = c_1 s(n)$ tárkorlátos ($c_1 \geq 1$) k-szalagos TG, mely felismeri L-et. Az M-ből kiindulva egy olyan $O(c^{S(n)})$-időkorlátos N TG-t fogunk konstruálni, melynek a nyelve szintén L. Nézzük M számításait az n hosszúságú inputokon. Egy ilyen számítás során a gép egy pillanatnyi helyzetét pontosan jellemzi az input (összesen n jel), a munkaszalagok tartalma (összesen legfeljebb $S(n)$ cella), a gép aktuális belső állapota, valamint a fejek helyzete a szalagokon. Egy ezeket rögzítő „jegyzőkönyvet" pillanatnyi helyzetleírásnak (PHL) nevezünk. Az M gép átmenetei és egy PHL ismeretében pontosan meg tudjuk mondani az M következő lépését, sőt a következő PHL-eket is. Ilyen értelemben egy PHL a számítás egy pillanatát hűen rögzítő fényképnek tekinthető. Mindebből számunkra most az a fontos, hogy ha a gép futása során egy PHL ismételten előfordul, akkor a gép biztosan *végtelen ciklusban van*, a két helyzet közötti lépéssor végtelen sokszor ismétlődik. A gép tárkorlátos voltából adódik, hogy egy számítás során csak véges sok PHL lehetséges. Így ha a gép elég sokáig fut, akkor egy PHL feltétlenül ismétlődik, tehát végtelen ciklusba kerültünk.

Nézzük ezt kicsit pontosabban! Hány darab PHL lehetséges összesen, ha a gépet n hosszú inputtal indítjuk? Érvényes a következő egyszerű felső becslés

($\#PHL$ a PHL-ek száma):

$$\#PHL \leq |Q||T|^{S(n)}(n+1)S(n)^k,$$

ahol $|Q|$ az M belső állapotainak száma, $|T|$ az M szalagjeleinek száma, az $(n+1)$ tényező az input fej lehetséges helyzeteinek a száma, az $S(n)^k$ tényező pedig a többi fej lehetséges helyzeteinek a száma. Használva, hogy $S(n) \geq s(n) \geq \log_2 n$, a $c_2 := 2^{k+1}|T|$ választással

$$\#PHL \leq \text{konstans} \cdot c_2^{S(n)}$$

adódik. Jelöljük t-vel a jobboldalon álló számot. Ha egy n-hosszú x inputon a gép t lépés után sem áll meg, akkor biztosan végtelen ciklusba került, hiszen ekkor van olyan PHL, ami két különböző lépés után előfordul. Kézenfekvő volna tehát M-et t lépés után lelőni. Az a baj ezzel az ötlettel, hogy nem feltétlenül tudunk eddig elszámolni, hiszen lehet, hogy a t korlát nem rekurzív függvénye n-nek. Olyan megoldásra van szükség, amely nem épít a t ismeretére.

Ennyi előkészület után lássunk az N gép konstrukciójához! A végtelen ciklusok felismerése és kezelése céljából megduplázzuk M-et. Legyen M_1 és M_2 a két példány. Ezeket a gépeket N részeinek tekintjük. Az M_1-et elindítjuk az x inputtal. Minden egyes lépése után ideiglenesen megállítjuk; ekkor M_2-t elindítjuk x inputtal a kezdő állapotból, és működtetjük legfeljebb addig a lépésig, ahol M_1 tart (jelölje ennek a lépésnek a sorszámát l). Az l sorszámot $O(S(n))$ extra cellán tároljuk és léptetjük. Ha valamely $j < l$-re az M_2 gép j-edik lépés utáni PHL-je megegyezik M_1-ével, akkor biztosan végtelen ciklusba kerültünk (PHL ismétlődik), tehát $x \notin L$. Ekkor N megáll elutasítva x-et. Ha ilyen ismétlődés nem fordult elő, akkor meglépjük M_1 következő, $l+1$-edik lépését, $l := l+1$, és ismételjük az előző eljárást. Természetesen ha M_1 megáll elfogadva (elutasítva) x-et, akkor N is megáll elfogadva (elutasítva) x-et.

Ezzel a megoldással biztosan felismerjük a végtelen ciklusokat: valójában már az első olyan l után megállunk, amikor egy PHL ismétlődik. Az új, összetett gép működése során ezért mindig teljesül az $l \leq t$ egyenlőtlenség. Innen már megbecsülhető N erőforrás-igénye. Beszámítva a számlálók (l és j) karbantartását és a PHL-ek összehasonlításának költségét is, a maximális futási idő legfeljebb $O(t^2) = O((c_2^{S(n)})^2) = O((c_2^2)^{c_1 s(n)})$, így $c = c_2^{2c_1}$ megfelel a tétel követelményeinek. További fontos tény, hogy a tárfelhasználás sem nőtt lényegesen: az összetett gép működéséhez $O(s(n))$ tárcella elegendő a munkaszalagokon. □

A bizonyítás minden nehézség nélkül átvihető nyelvek helyett függvényekre. Egyetlen számottevő módosítás célszerű: végtelen ciklus esetén, amikor az x inputra f nem értelmezett, legyen $f(x) = *$. Az így kapott (teljes) függvény szintén az $FSPACE(s(n))$ osztályba tartozik.

Tétel: *Ha $f \in FSPACE(s(n))$, akkor van olyan f-től függő c konstans, mellyel $f \in FTIME(c^{s(n)})$.*□

Nézzük most, hogy mi mondható idő- és tárkorlátokkal definiált nyelvosztályok esetén a komplemens nyelvek osztályairól. Emlékeztetésül: ha X nyelvek egy osztálya, akkor a co$X$ nyelvosztály éppen $I^* \setminus L$ alakú nyelvekből áll, ahol $L \in X$. Így pl. a co$TIME(t(n))$ nyelvosztályban a $TIME(t(n))$-beli nyelvek komplementerei vannak.

Egyszerűen adódik, hogy $TIME(t(n)) = \text{co}TIME(t(n))$. Legyen ugyanis M egy $ct(n)$ időkorlátos M TG, ami az L nyelvet ismeri fel. Ha megcseréljük az M elfogadó és elutasító (azaz nem elfogadó) állapotait, akkor a kapott N TG éppen az $I^* \setminus L$ nyelvet ismeri fel, és szintén $ct(n)$ időkorlátos. Analóg állítás érvényes tárosztályokra is; ennek az igazolása viszont kevésbé egyszerű.

Tétel: $SPACE(s(n)) = coSPACE(s(n))$.

Bizonyítás: Legyen $L \in SPACE(s(n))$. Alkalmazzuk a tár–idő-tétel szimulációját. Az adódó N TG szintén $O(s(n))$ tárkorlátos, és *minden inputra megáll*. Erre a gépre már működik időosztályoknál bevált ötlet: cseréljük fel az elfogadó és az elutasító állapotokat. □

Megemlítünk itt két további fontos nyelvosztályt, az *exponenciális időben felismerhető*, valamint a *polinom tárban felismerhető* nyelvek osztályait.

Definíció: EXPTIME $:= \cup_{k \geq 1} TIME(2^{n^k})$.

Definíció: PSPACE $:= \cup_{k \geq 1} SPACE(n^k)$.

Az EXPTIME osztályra úgy nézhetünk, mint a gyakorlatban előforduló nyelvek (eldöntési feladatok) univerzumára. Ezt úgy értjük, hogy nemigen van olyan praktikus feladat, ami ezen kívül levő, még nehezebb nyelvhez vezetne. Érvényesek a következő tartalmazási viszonyok:

Tétel: P $\subseteq$ PSPACE $\subseteq$ EXPTIME

Bizonyítás: Ha M egy $t(n)$ időkorlátos TG, akkor szükségképpen $ct(n)$ tárkorlátos is, hiszen egy lépésben legfeljebb annyi új cellára léphet, mint a szalagjainak száma. Ezt alkalmazva kapjuk, hogy $TIME(n^k) \subseteq SPACE(n^k)$, amiből P $\subseteq$ PSPACE következik. A másik állítást illetően legyen $L \in$ PSPACE. Ekkor $L \in SPACE(n^k)$, valamely k-ra. A tár–idő-tétel szerint ekkor van olyan $c > 0$, hogy $L \in TIME(c^{n^k}) \subseteq TIME(2^{\lceil c \rceil n^k}) \subseteq TIME(2^{n^{k+1}}) \subseteq$ EXPTIME. □

Megjegyezzük még – a $\subset$ jellel valódi tartalmazást jelölve –, hogy $TIME(t(n)) \subset \mathcal{R}$, $SPACE(s(n)) \subset \mathcal{R}$ és EXPTIME $\subset \mathcal{R}$. Ezek a tények a korábban már megismert *átlós eljárás* alkalmazásával igazolhatók. Az ötlet érzékeltetésére mutatunk egy rekurzív nyelvet, ami nincs az EXPTIME osztályban. Legyen

$$L = \{w \in I^*;\ \text{az } M_w \text{ TG létezik, és legfeljebb } 2^{2^{|w|}} \text{ lépésben elutasítja } w\text{-t}\}.$$

Itt $|w|$ jelöli a w szó hosszát. Először gondoljuk meg, hogy ha M egy $t(n)$ időkorlátos TG, akkor végtelen sok olyan $y \in I^*$ szó van, amire M_y létezik, és ugyanúgy viselkedik, mint M. Az utóbbi azt jelenti, hogy M_y is $t(n)$ időkorlátos és ugyanazt a nyelvet ismeri fel, mint M: $L_{M_y} = L_M$. Ilyen gépeket kaphatunk például úgy, hogy M belső állapotainak halmazát bővítjük olyan állapotokkal, amelyek sosem érhetők el a kiinduló helyzeteiből. (Tulajdonképpen az történik, hogy egy programot olyan részletekkel bővítünk, amelyekre sohasem kerül a vezérlés. A program ugyanúgy működik, mint az eredeti, de a leírása hosszabb lesz.)

Most megmutatjuk, hogy L nincs benne a $TIME(2^{2^{n-1}})$ nyelvosztályban. Ebből, EXPTIME $\subseteq TIME(2^{2^{n-1}})$ miatt következik, hogy $L \notin$ EXPTIME. Indirekt gondolkodva tegyük fel, hogy L felismerhető egy $c2^{2^{n-1}}$ időkorlátos M TG-vel. Legyen n_0 olyan nagy, hogy $c2^{2^{n-1}} < 2^{2^n}$ teljesüljön, ha $n > n_0$. Legyen w egy n_0-nál hosszabb szó, melyre M_w létezik, és ugyanúgy viselkedik mint M. Ez a gép is $c2^{2^{n-1}}$ időkorlátos, és $L = L_{M_w}$. A befejezés hasonlít a diagonális nyelvnél látottakra: ha $w \in L$, akkor M_w elfogadja w-t $c2^{2^{|w|-1}} < 2^{2^{|w|}}$ lépésben, amiből L definíciója szerint $w \notin L$ következik. Ugyanígy képtelenséghez vezet a $w \notin L$ feltevés is.

Feladat: Mutassuk meg, hogy a fenti L nyelv rekurzív.

A P, FP, PSPACE, EXPTIME osztályok *robusztusak* abban az értelemben, hogy nagymértékben függetlenek a gépmodelltől, amelynek a segítségével definiáltuk őket. Például szorítkozhattunk volna csak az egyszalagos Turing-gépekre, vagy használhattuk volna a RAM modellt. A TG-RAM szimulációkra kapott korlátokból azonnal következik például, hogy

$$\mathrm{P} = \left\{ L \subseteq I^* \;\middle|\; \begin{array}{l} \text{van olyan } k \in \mathbb{Z}^+ \text{ és } c > 0, \text{ hogy } L \text{ felismerhető} \\ cn^k \text{ logaritmikus költségű RAM-programmal} \end{array} \right\}.$$

Feladat: Mutassuk meg, hogy a lineáris időben felismerhető nyelvek osztálya $TIME(n)$ nem robusztus osztály. (Nézzük a palindrómák nyelvét egy-, illetve kétszalagos gépeken.)

Elméleti szempontból a polinom idejű algoritmusokat tekinthetjük hatékonyaknak. Ez több értelemben is megfelel a hatékonyságról meglevő tapasztalati képünknek. A lineáris, kvadratikus, sőt még a köbös algoritmusok gyorsaságával is általában elégedettek vagyunk. Bizonyos esetekben az ennél valamivel nagyobb kitevőjű módszerekkel is békét tudunk kötni. A sokkal nagyobb időigényű eljárások viszont (legalábbis amelyek viszonylag természetesen merülnek fel) használhatatlanok hosszú inputokon. Nézzünk mondjuk egy 2^n időigényű módszert $n = 1000$ hosszúságú input esetén! Ez a méret nem jelent gondot a mai gépeken n, n^2 sőt n^3 lépésszámú algoritmusoknál sem. Ezzel szemben a ma ismert fizikai elveken alapuló gépekre gondoldva teljesen reménytelennek tűnik, hogy 2^{1000} lépést megtehessünk egy emberöltő alatt.

Azt is hangsúlyoznunk kell, hogy nem minden polinom idejű módszer ad igazán hatékony megoldást. Egy cn^k futási idejű algoritmus is lehet elfogadhatatlan gyakorlati szempontból, ha c vagy k nagy. Aligha tekinthetünk jónak mondjuk egy n^{300} lépésszámú módszert. A polinom idejű algoritmus tehát a hatékony módszer *absztrakciójának*, elméletileg kezelhető általánosításának tekintendő. Két fontos tulajdonsága ebből a szempontból, hogy egyrészt tényleg tartalmazza az igazán hatékony (pl. lineáris, kvadratikus idejű) algoritmusokat, másfelől pedig robusztus fogalom az előbb tárgyalt értelemben. Ennek a kis fejtegetésnek az összegzéséül az *elméleti értelemben hatékonyan kezelhető feladatok* a következők:

P:	polinom időben felismerhető nyelvek
FP:	polinom időben számítható függvények

A legtöbb algoritmikus probléma, amivel korábban az adatszerkezetek elemeinél és a gráfalgoritmusok kapcsán találkoztunk, P-beli nyelv felismerésére, vagy általánosabban FP-beli függvény kiszámítására vezet. Például az $x_1, x_2, \ldots, x_n \in I^*$ szavak lexikografikusan rendezett sorrendjének előállítása megfogalmazható mint a következő (nyilvánvalóan FP-beli) f függvény kiszámításának a feladata:

$f : x_1 * x_2 * \ldots * x_n \to y_1 * y_2 * \ldots * y_n$, ahol $y_1 \leq y_2 \leq \ldots \leq y_n$ az x_i szavak lexikografikusan rendezett sorozata. Néhány további ilyen feladat:

1. Kupacépítés
2. Minimális költségű feszítőfa keresése gráfban (Prim és Kruskal módszere)
3. Maximális párosítás keresése páros gráfban (magyar módszer)
4. Maximális folyam számítása egész kapacitásokkal rendelkező hálózatban (Dinic algoritmusa)
5. Gráf csúcsainak 2 színnel való kiszínezése.

Nézzük meg közelebbről mondjuk a magyar módszert! Mint már láttuk, ennek az uniform költsége $O(ne)$, ahol n az éllistával adott G bemeneti gráf pontjainak,

e pedig az éleinek száma. Az éllistának $n + 2e$ cellája van. Az éllista egy cellája kb. $3 \log n$ biten elfér: ebből $\log n$ bit a cellához tartozó csúcs sorszáma. A cellában levő mutatóra mintegy $2 \log n$ bitet szánunk. Ezek elegendően hosszúak lesznek, mivel összesen legfeljebb n^2 cella van a szerkezetben. Az input hossza tehát körülbelül $3(n + 2e) \log n$ bit.

Feladat: Mutassuk meg, hogy a magyar módszer megvalósítható $O(ne \log n)$ logaritmikus költségű RAM-programmal. (Arra kell csak ügyelni, hogy a magyar módszer *egy* elemi lépésének a logaritmikus költsége ne legyen több, mint $O(\log n)$.)

A feladat szerint a magyar módszer logaritmikus költsége az input hosszának négyzetével arányos korlát alatt marad. A módszer tehát tényleg polinom idejű.

Az előzőekkel szemben ellenpontként álljon itt néhány (mai ismereteink szerint) polinom időben nem megoldható feladat. Ezek látszólag egymástól távol eső problémák, mégis algoritmikus szempontból sok közük van egymáshoz. Egyebek között ennek a kapcsolatnak fogunk utánajárni a következőkben.

1. Hamilton-körrel rendelkező gráfok felismerése
2. Utazó ügynök probléma (minimális költségű Hamilton-kör találása)
3. A 3 színnel színezhető gráfok felismerése
4. Maximális méretű klikk keresése gráfokban.

8.3. Nemdeterminisztikus Turing-gépek; az NP nyelvosztály

Az eddig tárgyalt gépmodellek (TG, RAM) valósághoz közelinek mondhatók abban az értelemben, hogy elég hűen tükrözik a számítások tényleges időigényét. A most terítékre kerülő modell ebből a szempontból alaposan elrugaszkodott a valóságtól. Hogy miért foglalkozunk vele mégis? Mert segítségével igen természetesen körülhatárolható egy érdekes és gazdag nyelvosztály. A modellt tehát mint definíciós eszközt érdemes szemlélni.

A *nemdeterminisztikus Turing-gép* (röviden NTG) definíciója abban tér el az egyszalagos Turing-gépétől, hogy a δ átmenetfüggvény nem egyértékű. A δ a lehetséges lépések olykor egynél több elemű véges halmazát jelöli ki:

$$\delta(q, a) \subseteq Q \times T \times \{\text{jobb}, \text{bal}, \text{helyben}\}.$$

Ennek megfelelően a gép esetleg több lehetőségből választhat. A q állapotban az a szalagjel mellett a következő lépést a $\delta(q, a)$ halmazból kell választani.

A gép futása (egy számítási út):

Kiindulási helyzet:
Mint az egyszalagos TG-nél, az $x \in I^*$ input szó a szalagon van, balra igazítva, utána üresjelek. A gép a q_0 kezdőállapotban van, a fej pedig a szalag első celláján.
Egy lépés:
A gép egy pillanatnyi állapotát a belső állapot, a szalag tartalma, valamint a fej helyzete határozza meg. A következő pillanatnyi állapotba úgy jut el, hogy a belső állapotból, valamint a fej alatt található szalagjelből álló párra alkalmazva a δ halmazértékű függvényt, választja annak egy tetszőleges elemét, majd az annak megfelelő belső állapotot veszi fel, miközben a megfelelő szalagjelet írja a fej alatti cellába, majd a megfelelő fejmozgatást végzi el. Ha az adott párra δ nincs értelmezve, akkor a gép megáll.

Példa: Tegyük fel, hogy a gép a q állapotban van, a fej alatti szalagjel a, és $\delta(q,a) = \{(q,a,helyben),\ (q_1,b,jobbra),\ (q_2,c,balra)\}$. Ekkor a gép a három lehetséges lépés közül választhat, mindegyik választás megengedett.

Definíció (input elfogadása): *Az M NTG* elfogadja *az $x \in I^*$ inputot, ha az M-et x bemenettel a kiinduló helyzetből indítva van legalább egy elfogadó (egy elfogadó állapotban véget érő) számítási út.*

A korábbiakkal összhangban jelölje L_M az M által ilyen értelemben elfogadott nyelvet. A definícióból közvetlenül adódik a következő:

Állítás: *Az $x \in I^*$ input szó pontosan akkor nincs L_M-ben, ha az M gépet x inputtal indítva nincs elfogadó számítási út.*□

Az NTG-k számításait néha hasznos egy gyökeres irányított faként elképzelni. A fa csúcsait a gép pillanatnyi helyzeteivel címkézhetjük. Egy csúcsnak annyi fia van, ahány lépés megengedett a csúcsnak megfelelő pillanatnyi helyzetből. A példabeli gép esetén, ha a pillanatnyi helyzetben a belső állapot q, a fej alatti szalagjel a, akkor a csúcsnak három fia van. A fa gyökerét az $x \in I^*$ inputnak megfelelő kiinduló helyzettel címkézzük. A gép pontosan akkor fogadja el az x inputot, ha a fában van olyan gyökértől levélig menő út, melynél a levélhez elfogadó állapot tartozik.

A közönséges Turing-gépek mintájára beszélhetünk időkorlátos nemdeterminisztikus gépekről. Legyen $t : \mathbb{Z}^+ \to \mathbb{Z}^+$ egy függvény, melyre minden $n \in \mathbb{Z}^+$ esetén teljesül a $t(n) \geq n$ egyenlőtlenség.

Definíció ($t(n)$ időkorlátos NTG):
Egy M nemdeterminisztikus Turing-gép $t(n)$ időkorlátos, ha n hosszúságú inputokon M minden számítási út mentén legfeljebb $t(n)$ lépést téve megáll.

Úgy is fogalmazhatunk, hogy egy tetszőleges n hosszúságú $x \in I^*$ input esetén a számításhoz rendelt fa magassága legfeljebb $t(n) + 1$. A követelmény egyaránt korlátozza az elfogadó és az elutasító számítási utak hosszát. Egy időkorlátos NTG tehát egyetlen számítási úton sem kerülhet végtelen ciklusba. A fejezet bevezetőjében említettük, hogy az NTG nem egy valósághű számítási modell. A működésükben éppen ez a költségszámítás jelenti az igazán irreális tényezőt. Nem ismert (és a mai tudásunk szerint nem látszik lehetségesnek) olyan tényleges fizikai megvalósításuk, mely egy $t(n)$ időkorlátos NTG működését $t(n)$-nel arányos időben szimulálni tudná. (Olykor-olykor felröppenek sajtókacsák, állítva, hogy sikerült igazi nemdeterminisztikus gépet konstruálni...)

Az időkorlátos gép fogalmával a kezünkben a $TIME$ osztályok mintájára definiálhatjuk a nemdeterminisztikus időkorlátokkal kijelölt nyelvosztályokat.

Definíció:
$NTIME(t(n)) := \{$*az* $O(t(n))$ *időkorlátos NTG-k által elfogadott nyelvek*$\}$.

NTG-k segítségével fogalmazható meg kényelmesen a számításelmélet egyik legérdekesebb nyelvosztályának, az NP -nek a definíciója.

Definíció: NP $:= \cup_{k \geq 1} NTIME(n^k)$.

Az NP nyelvosztály ugyanúgy épül fel az $NTIME(n^k)$ alakú nyelvhalmazokból, mint a P a $TIME(n^k)$ alakúakból. A definíciók közötti nyilvánvaló formai hasonlóság alapján az NP osztályt a P nemdeterminisztikus megfelelőjének tekinthetjük. Az NP a nemdeterminisztikus polinom idejű Turing-gépekkel felismerhető nyelvekből áll. Az egyszalagos (közönséges) TG-k felfoghatók NTG-nek is, sőt egy $t(n)$ időkorlátos TG tekinthető $t(n)$ időkorlátos NTG-nek is. Így azonnal adódik, hogy $TIME(n^k) \subseteq NTIME(n^k)$, amiből az egyesítésekre térve:

Állítás: P $\subseteq$ NP.□

Az NP nyelvosztály tehát tartalmazza P-t, a hatékonyan kezelhető nyelvek osztályát. A két osztály, a P és az NP közötti analógia korántsem teljes. Nem ismert például, hogy az NP osztály megegyezik-e a coNP nyelvosztállyal (emlékeztetőül: coNP $= \{L \subseteq I^*; I^* \setminus L \in$ NP$\}$). A témakörben dolgozó kutatók azt sejtik, hogy a két osztály különböző. Szintén nyitott kérdés, hogy P megegyezik-e NP-vel. Itt is a nemleges válasz látszik valószínűnek. A (P = NP?) kérdés **a számításelmélet egyik legfontosabb megoldatlan problémája**.

Állítás: P $\subseteq$ NP $\cap$ *co*NP.

Bizonyítás: P $\subseteq$ NP miatt coP $\subseteq$ coNP. Ezután az állítás azonnal adódik a P = coP egyenlőségből. □

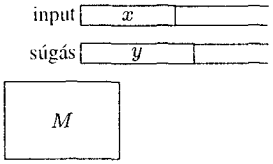
Egyszerű alkalmazásként megjegyezzük, hogy a P = NP egyenlőségből NP = coNP következne.

A nemdeterminisztikus számítások egy determinisztikus modellje

Először megsejtesz valamit. Ne nevessetek, ez a legfontosabb lépés. Utána kiszámítod a következményeket... RICHARD P. FEYNMAN
a fizikai felfedezésekről

Célunk itt nyelvek nemdeterminisztikus felismerésének determinisztikus modelljét adni. Ezzel végső soron kényelmes eszközt kapunk nyelvek (feladatok) NP-be tartozásának igazolására, amely jól használható konkrét nyelvek esetén.

Legyen M kétszalagos (közönséges) TG. Tegyük fel, hogy M inputja két részből áll, egyik része – mondjuk $x \in I^*$ – az első szalagon van, a másik része $y \in I^*$ a másikon. Az utóbbi, az y-t tartalmazó szalag csak olvasható. Ezt nevezzük az M *súgásszalagjának.*



A korábbi definícióval összhangban az M által felismert L_1 nyelv azon (x, y) szópárok halmaza ($x, y \in I^*$), melyekkel a kezdő helyzetből elindítva M elfogadó állapotban áll meg.

Definíció (nemdeterminisztikus felismerés): *Az M által nemdeterminisztikusan felismert L nyelv a következő:*

$$x \in L \text{ akkor, és csak akkor, ha van olyan } y \text{ súgás, hogy } (x, y) \in L_1.$$

Az M gép által nemdeterminisztikusan felismert L nyelvbe tehát pontosan azok az $x \in I^*$ szavak tartoznak, melyek kiegészíthetők alkalmas y-nal (súgással) úgy, hogy az (x, y) párt az M elfogadja. Itt fontos mozzanat, hogy a jó súgásnak csak a *létezése* követelmény. Semmit sem teszünk fel arról, hogy adott x-hez

miként található alkalmas y. Az y szót szokás még az $x \in L$ állítás *tanújának*, vagy az x elfogadásához vezető *sejtésnek* is nevezni. A *súgás* és *sejtés* szavakból kiszüremlő bizonytalanság arra hivatott utalni, hogy az y-nal szemben nincs kiszámíthatósági követelmény.

A következő tétel szerint az NP-beli L nyelveknél az M egy polinom idejű TG és a tanú (súgás) hossza is polinomkorlátosnak vehető.

Tétel (tanú-tétel): *Egy $L \subseteq I^*$ nyelvre a következő két állítás egyenértékű:*
(a) $L \in$ NP.
(b) Van olyan $c > 0$ állandó, továbbá egy $L_1, \in$ P nyelv, mely olyan $(x, y) \in (I^*)^2$ párokból áll, hogy $|y| \leq |x|^c$ és $x \in I^*$ esetén $x \in L$ pontosan akkor, ha van $y \in I^*$ úgy, hogy $(x, y) \in L_1$.

Bizonyítás: (Vázlat) $(a) \Rightarrow (b)$: Az $L \in$ NP feltétel miatt létezik egy n^{c_1} időkorlátos N NTG, mely felismeri L-et. Tegyük fel, hogy N-nek egy lépésnél legfeljebb d elágazási lehetősége van, vagyis d a $\delta(q, a)$ halmazok maximális elemszáma. Egy $c > 0$ szám és egy olyan M polinomkorlátos TG létezését kell megmutatnunk, mely (x, y) alakú inputokkal dolgozik, és $x \in L$ pontosan akkor teljesül, ha van olyan $y \in I^*$, $|y| \leq |x|^c$, hogy $(x, y) \in L_1 = L_M$.

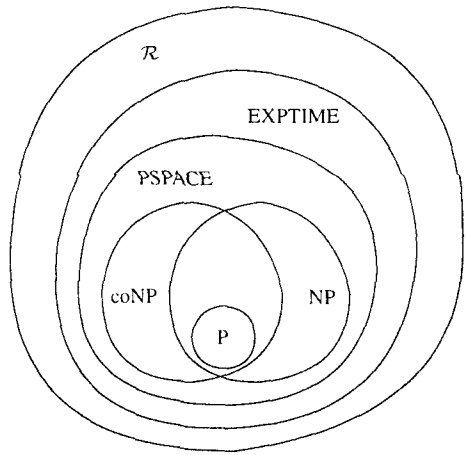
Legyen $x \in L$, $|x| = n$. Erre az inputra egy olyan $y = y_1 y_2 \cdots y_m$ alakú sorozatot ($m \leq n^{c_1}$) fogunk jó súgásnak tekinteni, mely az N egy elfogadó számítási útját írja le az x input mellett. Az y_j (bináris) szavak 1 és d közé eső egészek kódjai, amiből $|y| \leq n^{c_1} \lceil \log_2(d+1) \rceil \leq n^c$ alkalmas c konstanssal. Az y_j által ábrázolt szám mondja meg, hogy a j-edik lépésben N melyik lehetséges lépését kell választani (az adott helyzetben szóba jövő legfeljebb d lehetséges lépés közül) az elfogadó számítási úton. Az M TG egy „nagy" lépése a következő: a súgásszalagról megnézi, hogy N melyik lépését kell megtenni (ez legfeljebb $\lceil \log_2(d+1) \rceil$ bitet jelent), majd ezt meglépi. Ez N-től függő konstans időben megtehető. Az M pontosan akkor álljon meg, ha N szimulált „lépése" megállás, fogadja el az inputot, ha ez a megállás N elfogadó állapotában történt. Az M álljon meg elutasító állapotban akkor is, ha a súgás következő darabja nem értelmezhető N legális lépéseként. A fentiek alapján M az (x, y) összetett input hosszában lineáris ideig működik. Világos az is, hogy $x \in L$ esetén van olyan jó y súgás, melyre $|y| \leq n^c$. Ha viszont $x \notin L$, akkor $(x, y) \notin L_1 = L_M$ tetszőleges $y \in I^*$-ra. Ekkor ugyanis az (x, y) párt M vagy azért utasítja el, mert y nem kódja N legális számításának, vagy pedig azért, mert y elutasító számítási út kódja; elfogadóé nem lehet, hiszen elfogadó út nem létezik.
$(b) \Rightarrow (a)$: Egy $x \in L$ inputhoz a feltétel szerint van rövid tanú (súgás): ha $|x| = n$, akkor van olyan legfeljebb n^c hosszúságú y, hogy $(x, y) \in L_1$. Az ilyen rövid jó súgások száma legfeljebb $\leq 2^{n^c}$. Megmutatható, hogy ennyi lehetőség

nemdeterminisztikusan n^c lépésben végignézhető. Ezt itt nem részletezzük (de a következő feladatban körvonalazzuk). □

Feladat: Tegyük fel, hogy az M egy n^c időkorlátos TG, mely a tételbeli L_1 nyelvet ismeri fel. Ebből készítünk egy N NTG-t a következők szerint. Az N állapotai egyezzenek meg az M állapotaival, elfogadó (elutasító) állapotai az M elfogadó (elutasító) állapotaival. Az N-nek legfeljebb két lehetséges lépése lesz a q állapotban és az a szalagjel olvasásakor. Ezek legyenek azok a lépések, melyeket M tesz abban a két esetben, amikor
M állapota q, az inputfej alatti jel a, a súgásfej alatti jel 0;
M állapota q, az inputfej alatti jel a, a súgásfej alatti jel 1.
Természetesen N-nek csak egy szalagja van, így az M lépéseiből csak az állapotváltoztatást és az input szalaggal kapcsolatos teendőket az állapotváltoztatást és az *input szalaggal kapcsolatos teendőket kell megtenni. Bizonyítandó, hogy N éppen* L-et ismeri fel, és n^c időkorlátos.

Megjegyzések: 1. A (b) állításban az $|y| \leq |x|^c$ feltétel helyettesíthető az $|y| = |x|^c$ feltétellel. Ugyanis egyrészt feltehetjük, hogy c egész; ha kell, helyettesíthetjük $\lceil c \rceil$-vel. Másrészt egy y tanút kiegészíthetünk a végére ragasztott tetszőleges bitekkel, hogy a hossza éppen $|x|^c$ legyen.
2. A tanú-tételből következik, hogy NP $\subseteq$ PSPACE. Legyen ugyanis $L \in$ NP, x egy input szó, $|x| = n$. A legfeljebb n^c hosszú $y \in I^*$ szavakat mint tanújelölteket sorban generáljuk, és minden (x, y) szópárra lefuttatjuk M-et, egészen addig, amíg vagy elfogadó számítást kapunk (ekkor $x \in L$), vagy elfogynak a lehetséges jelöltek (ekkor $x \notin L$). Ez minden egyes y szóra $O(n^c)$ időt jelent. Ha egy y szóval végeztünk, akkor a kapott részeredményeket eldobhatjuk, és kezdhetjük az M számítását elölről a következő tanújelölttel. A szükséges adminisztráció megoldható egy n^c bites számláló segítségével. Az n^c hosszú tanújelölteket tekinthetjük úgy, mint n^c bites (binárisan ábrázolt) számokat. A tanújelöltek szisztematikus végignézése azt jelenti, hogy nullától $2^{n^c} - 1$-ig végiglépkedünk az egészeken, mindig az eggyel nagyobbat véve. Abban a kellemes – és igen egyszerű – helyzetben vagyunk, hogy ehhez a lépkedéshez nem kell sok memória. A következő szám (tanújelölt) könnyen és gyorsan megkapható pusztán az előző ismeretében.

Az eddig megismert bonyolultsági osztályok tartalmazási viszonyait szemlélteti a következő vázlat.



Nem tudjuk, hogy a P osztály valódi része-e a PSPACE osztálynak. A sejtés a korábbiakkal összhangban az, hogy igen. Az eddig vett nyitott kérdések közül ez a leggyengébb, amint azt az alábbi egyszerű következtetési lánc mutatja:

$$\text{NP} \neq \text{coNP} \Rightarrow \text{P} \neq \text{NP} \Rightarrow \text{PSPACE} \neq \text{P}$$

Egy „kép" az NP-beli nyelvekről:

A tanú-tétel determinisztikus jellemzése az egyik leghasznosabb eszköz nyelvek NP-be tartozásának belátására. Gondolva a polinom idejű számítások robusztusságára is, az L_1 nyelvről (döntési problémáról) elég látni, hogy kezelhető polinom idejű „programmal". Nem kell a nehézkes TG-modellel dolgozni.

Az NP nyelvosztály determinisztikus leírásának elemeit segíthet megjegyezni az alábbi táblázatban vázolt helyzet.

Bíró:	M polinom idejű algoritmus (pl. TG, RAM program)
Vádlott:	$x \in I^*$
Tanú:	$y \in I^*$, $\lvert y\rvert \leq \lvert x\rvert^c$
Vád:	Az állítás, miszerint $x \in L$.

Az M algoritmus (a kissé türelmetlen bíró, akinek időkorlátja van) nem tesz mást, mint ellenőrzi, hogy az y tanú(vallomás) tényleg bizonyítja-e a vádat, ami az $x \in L$ állítás. A tétel L_1 nyelve azon *(vádlott,tanú)* párokból áll, amelyeknél a bíró szerint a *tanú* bizonyítja a *vádlott*ról a vádat. Nem kell foglalkoznia a bizonyíték előtalálásával. Ez – a mai tudásunk szerint – sok érdekes esetben nem is tehető meg polinom időben. Csak az eléje tárt bizonyíték helyességével kell törődnie. Ezt a szemléletet fogjuk használni a következő példáknál.

8.4. Néhány NP-beli nyelv

8.4.1. 3 színnel színezhető gráfok

Rögzítsük gráfok egy nem túl pazarló kódolását bitsorozatokkal. Például egy n szögpontú (irányított, vagy irányítatlan) gráf adjacencia-mátrixát tárolhatjuk egy n^2 hosszúságú bitsorozatként. Egy ilyen kódolás eredményeként a gráfokat $\{0,1\}^*$-beli szavak reprezentálják, gráfok halmazainak pedig nyelvek felelnek meg. Emlékeztetőül: a $G = (V, E)$ gráf k színnel színezhető, ha a csúcsaihoz i egészeket (színeket) rendelhetünk úgy, hogy $1 \leq i \leq k$, és ha $(v, w) \in E$, akkor v és w különböző színűek. Legyen 3-SZÍN a 3 színnel színezhető gráfok kódjaiból álló nyelv.

Állítás: *3-SZÍN* $\in$ NP.

Bizonyítás: Elegendő megmutatni, hogy a három színnel való színezhetőségnek van rövid és hatékonyan ellenőrizhető tanúja. Egy n szögpontú 3 színnel színezhető G gráf jó színezése alkalmas tanú lesz. Egy ilyen színezés leírható $2n$ bittel (például legyen 01=piros, 10=sárga, 11=zöld). A bíró ellenőrzi, hogy az adott színezés jó-e. Ez a feladat polinom időben megoldható.

A tanú-tétel alkalmazásakor a G gráf felel meg az x inputnak, a színezés az y tanú. Az L_1 nyelv $(G, \textit{színezés})$ alakú párokból áll, ahol a *színezés* egy helyes 3-színezése G-nek. A bírót jelentő M algoritmusnak csak annyit kell tudnia, hogy ellenőrzi a két komponens harmóniáját: hogy a javasolt színezés tényleg jó-e.

Figyeljük meg, hogy tényleg teljesülnek a tanú-tétel (b) részének a követelményei. Ha $G \in$ 3-SZÍN, akkor van $(G, \textit{színezés})$ alakú pár L_1-ben. Ha $G \notin$ 3-SZÍN, akkor pedig nem létezhet ilyen pár. □

Feladat: Adjunk $O(n^2 \log n)$ logaritmikus költségű RAM programot, mely megoldja a bíró feladatát: adott n szögpontú G gráfra és a csúcsok egy adott színezésére eldönti, hogy utóbbi 3 színezése-e G-nek.

8.4.2. Hamilton-körrel rendelkező gráfok

A G irányítatlan gráf egy köre *Hamilton-kör*, ha abban G minden csúcsa pontosan egyszer szerepel. Legyen H a Hamilton-kört tartalmazó gráfokból álló nyelv.

Állítás: $H \in$ NP.

Bizonyítás: A $G \in$ H állításnak rövid tanúja egy Hamilton-kör. A Hamilton-kör leírható a csúcsoknak a kör mentén való bejárási sorrendjével, ami $O(n \log n)$

bitet jelent, ha G-nek n csúcsa van. A bíró nyelve (L_1 a tanú-tételben) $(G; [v_1, v_2, \ldots, v_n])$ alakú párokból áll, ahol a $[v_1, v_2, \ldots, v_n]$ Hamilton-köre a G gráfnak. A bíró dolga egyszerű: ellenőrzi, hogy tényleg a G köre-e a tanú, és hogy G minden csúcsa pontosan egyszer szerepel-e a listán. Ezek a feladatok polinom időben megoldhatók. □

Feladat: Adjunk minél hatékonyabb RAM programot a bíró feladatának megoldására.

A G irányított gráf egy irányított köre *Hamilton-kör*, ha abban G minden csúcsa pontosan egyszer szerepel. Legyen IH az irányított Hamilton-kört tartalmazó irányított gráfok nyelve. Az előzőekhez hasonlóan látható, hogy IH $\in$ NP.

8.4.3. Síkba rajzolható gráfok

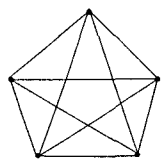
Egy gráf *síkba rajzolható*, ha a pontjainak kölcsönösen egyértelműen megfeleltethetők síkbeli pontok, az éleknek pedig a megfelelő csúcsokat összekötő egyenesszakaszok úgy, hogy a különböző szakaszok legfeljebb csak a végpontjaikban találkozhatnak. Legyen *Sík* a síkba rajzolható gráfok nyelve. Mint látni fogjuk, ennek a nyelvnek a komplementere is NP-beli. Az ilyen nyelvek esetén annak a ténynek is van rövid és hatékonyan ellenőrizhető tanúja, hogy egy input szó *nem* tartozik a nyelvbe.

Definíció (jól karakterizált nyelv): *Az $L \subseteq I^*$ nyelv jól karakterizált, ha $L \in$ NP $\cap$ co*NP.

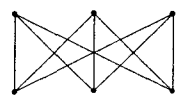
Állítás: *A Sík nyelv jól karakterizált.*

Bizonyítás: Egy gráf síkba rajzolhatóságának a tanúja egy síkba rajzolás. (Megmutatható, hogy ha $G \in$ *Sík*, akkor van a G-nek olyan síkba rajzolása is, ahol a pontok koordinátái $|V(G)|$-nél nem nagyobb természetes számok). A síkba rajzolás helyességét könnyű ellenőrizni. Lényegében végpontjaikkal adott szakaszok metszéseit kell számolni. Ezzel beláttuk, hogy *Sík* $\in$ NP.

A *Sík* $\in$ coNP állítás igazolásához azt kell megmutatni, hogy a $G \notin$ *Sík* ténynek is van hatékonyan ellenőrizhető tanúja. Ez pedig következik K. Kuratowski nevezetes tételéből : a G gráf nem síkba rajzolható $\Leftrightarrow$ G topologikusan tartalmaz teljes ötszöget vagy 3 ház – 3 kút gráfot.



teljes ötszög



3 ház – 3 kút

A $G \notin$ *Sík* ténynek egy G-beli topologikus teljes 5-ös vagy 3 ház – 3 kút gráf lehet a tanúja. A tanú tehát a tiltott részgráfot kijelölő 5 vagy 6 pontból áll és az ezek közül a megfelelőeket összekötő, közös belső pontot nem tartalmazó G-beli utakból. A bírónak csupán a megfelelő pontpárok közötti utak meglétét kell ellenőriznie, és még azt, hogy az utak diszjunktak. Ezek egyszerűen és gyorsan verifikálhatók. □

Megjegyzések: 1. Valójában a Kuratowski-tételből az erősebb *Sík* $\in$ P állítás is következik. A síkgráfok tehát polinom időben felismerhetők. Ezen felül a síkba rajzolásra is van hatékony algoritmus. Ezeket az állításokat nem részletezzük.
2. Szigorúan véve a *Sík* nyelv komplementere nem pontosan a síkba nem rajzolható gráfokból áll. A komplementer nyelvbe tartoznak még azok a szavak is, melyek nem kódjai gráfoknak. Ezekről azonban feltehetjük, hogy könnyen felismerhetők. Arról van csupán szó, hogy értelmes kódolást választottunk, aminél nem okoz gondot a hibás kódok felismerése. Ezt figyelembe véve a *Sík* $\in$ coNP állítás érdemi része tényleg az, hogy a síkba nem rajzolhatóságnak van hatékony tanúja.

Itt elgondolkodhatunk egy kicsit azon, hogy a 3-SZÍN és a H nyelvek hogyan viszonyulnak a coNP osztályhoz. Van-e annak rövid és hatékonyan ellenőrizhető tanúja, hogy egy gráf *nem* színezhető 3 színnel, illetve, hogy *nem* tartalmaz Hamilton-kört? Ilyen tanúk nem ismeretesek, és a szakértők úgy sejtik, hogy nem is léteznek. Ha igazuk van, akkor ezek a nyelvek mutatják az NP és a coNP osztályok különbözőségét.

Itt ismét visszautalunk Merlin szerencséjére az első fejezetből: a házasítási feladat megoldhatatlanságának volt rövid, a király által is gyorsan átlátható bizonyítéka (egy király legfeljebb polinom időt hajlandó ilyesmire pazarolni). Egy König-akadály a megoldhatatlanság gyors bizonyítéka. A varázsló nem lenne ilyen könnyű helyzetben, ha arról kellene Arthurt meggyőznie, hogy egy nagy gráfban *nincs* Hamilton-kör. Az NP-beli feladatok éppen azok, ahol az igenlő válasz esetén a varázsló elkerülheti a türelmetlen király haragját.

A gráfalgoritmusokról szóló fejezetben találkoztunk néhány minimax tétellel (Ford–Fulkerson, Menger, König). Az ilyen eredmények gyakran hordozzák azt az algoritmikus jelentést, hogy a megfelelő nyelv jól karakterizált. Példaként nézzük a Ford–Fulkerson-tételt; a másik kettő meggondolását az olvasóra hagyjuk.

Feladat: Álljon a *Folyam* nyelv azon $(\mathcal{H}, k)$ alakú párokból, amelyekre $\mathcal{H} = (G, s, t, c)$ egy egész kapacitásokkal rendelkező hálózat, $k \in \mathbb{Z}^+$, és $\mathcal{H}$-ban van olyan f folyam, aminek az értéke legalább k. Mutassuk meg, hogy *Folyam* $\in$ NP $\cap$ coNP. (Legyen $x \in I^*$. Az $x \in$ *Folyam* tény tanúja egy $\mathcal{H}$-beli f folyam, melyre $|f| \geq k$; $x \notin$ *Folyam* tanúja pedig egy $\mathcal{H}$-beli vágás, aminek a kapacitása kisebb, mint k.)

Megjegyezzük, hogy a feladatban foglaltnál erősebb $Folyam \in \mathrm{P}$ állítás is igaz; Dinic módszerével polinom időben találhatunk egy $\mathcal{H}$-beli f^* maximális folyamot. Nyilvánvalóan $(\mathcal{H}, k) \in Folyam$ egyenértékű azzal, hogy $|f^*| \geq k$.

8.4.4. A prímszámok nyelve

A $p > 1$ egész szám *prímszám*, ha a pozitív egészek közül csak az 1 és p számokkal osztható maradék nélkül. A $p > 1$ egész szám *összetett szám*, ha nem prímszám.

Jelölje Π a (binárisan ábrázolt) prímszámok nyelvét. Itt a komplementer tulajdonságnak, az összetettségnek van egyszerű tanúja. Ha m összetett szám, akkor ezt egy k valódi osztója ($1 < k < m$) tanúsítja. A k ismeretében m összetettsége az $m : k$ osztás elvégzésével igazolható. Tegyük fel, hogy m egy n-bites szám (az input hossza n). Az elemi iskolából ismert osztó algoritmussal az $m : k$ osztás $O(n^2)$ bit-művelettel elvégezhető. Van tehát polinom idejű bíró. Ezzel beláttuk, hogy $\Pi \in \mathrm{coNP}$. Lényegesen bonyolultabb a következő állítás.

Tétel (V. R. Pratt, 1975): $\Pi \in \mathrm{NP}$ *(tehát* Π *jól karakterizált).*

Pratt tételének bizonyításához szükségünk lesz egy számelméleti lemmára. Emlékeztetőül: $a \equiv b \pmod{m}$ jelöli azt a tényt, hogy $b - a$ osztható m-mel (a, b, m egész számok).

Lemma: *Legyen $p \geq 2$ egy egész szám. A p pontosan akkor prímszám, ha van olyan $1 \leq g < p$ egész, melyre teljesülnek az alábbiak:*

1. $g^{p-1} \equiv 1 \pmod{p}$,
2. $g^{\frac{p-1}{r}} \not\equiv 1 \pmod{p}$ *minden r prímszámra, melyre $r|p-1$.*

□

A lemma bizonyítását mellőzzük. A számelmélet elemeiben jártas olvasó számára megjegyezzük, hogy g a modulo p maradékosztályok multiplikatív csoportjának egy generátoreleme (ún. primitív gyök).

Szükségünk lesz még a *gyors hatványozás* algoritmusára. Ez egy ősrégi ötleten[1] alapul, és fontos építőköve a számítógépes aritmetikának. Gyors hatványozással egy a^m alakú hatvány (m egy pozitív egész) legfeljebb $2\log_2 m$ szorzással kiszámítható. Írjuk fel ugyanis az m kitevőt kettes számrendszerben $m =$

[1] Az i.e. 1650 táján íródott egyiptomi *Rhind-papirusz* szorzás helyett összeadással, vagyis a hatványozás helyett a szorzás elvégzésére ismerteti a módszert.

$e_0 + e_1 2^1 + e_2 2^2 + \cdots + e_k 2^k$, $k \leq \log_2 m$ és $e_j \in \{0, 1\}$. Számítsuk ki ismételt négyzetre emelésekkel sorra az a^{2^j} hatványokat $j = 1, 2, \ldots, k$. Ez k szorzást jelent. Végül szorozzuk össze az a^{2^j} hatványokat azokra a j értékekre, melyekre $e_j = 1$. Ez legfeljebb k további szorzást igényel. Úgy fogalmazhatunk, hogy a szorzások száma polinomiális (sőt lineáris) az m kitevő *méretében*. Ha a egész szám, akkor az a^m végeredmény mérete exponenciális is lehet a méretéhez képest. Nem ilyen rossz a helyzet, ha a gyors hatványozást $a^m (mod\, m_1)$ alakú maradékosztályok kiszámítására használjuk. Ezt a feladatot *moduláris hatványozásnak* nevezik. Ekkor minden egyes szorzás után redukálhatjuk a szorzatot modulo m_1. Ha itt a és m_1 legfeljebb n-bites számok, akkor a számítás során fellépő egészek hossza legfeljebb $2n$ bit lesz. A gyors hatványozás tehát polinom idejű algoritmust ad a moduláris hatványozás feladatára (binárisan ábrázolt a, m, m_1 bemenet esetén).

Pratt tételének bizonyítása: Olyan tanút kell javasolnunk, amelynek birtokában hatékonyan igazolható, hogy az inputként adott p prímszám. Próbáljunk a lemma által sugallt úton járni: adott g szám, valamint a $p - 1$ egész $r_1, \ldots, r_k$ prímosztói ismeretében *gyors hatványozással* ellenőrizhetők a lemma kongruencia-feltételei. Első közelítésben a g és az $r_1, \ldots, r_k$ számokat tekintjük tanúnak. Tanúsítanunk kell még, hogy $r_1, \ldots, r_k$ éppen a $p - 1$ prímosztói. Ennek a könnyebbik része ellenőrizni, hogy $p - 1$ előáll-e az r_i számok hatványainak szorzataként. Tanúsítani kell ezen kívül, hogy az r_i számok is prímek. Ezekhez ugyanolyan szerkezetű tanút használunk, mint a p-hez. Például az r_1-hez szükségünk lesz egy alkalmas g_1 számra, valamint az $r_1 - 1$ prímosztóira és azok tanúira is. Ez tehát egy rekurzíve megadott szerkezetű tanú lesz. Megmutatható, hogy ha a p input egy n bites szám, akkor az így felépített tanú összmérete $O(n^2)$, és a bíró algoritmusa n-ben polinomiális idejű. □

Megjegyzés: Nyitott kérdés, hogy $\Pi \in$ P teljesül-e, azaz van-e polinom idejű módszer a prímtulajdonság (összetettség) eldöntésére. Ha a válasz nemleges, akkor P $\neq$ NP $\cap$ coNP. A legjobb ismert prímfelismerő módszer (L. M. Adleman, C. Pomerance, R. S. Rumely, 1983) futási ideje n bites input esetén $n^{c \log\log n}$.

8.4.5. A felismerés és a keresés kapcsolata (prímtényezős felbontás)

A nyelvfelismerési problémák olyan feladatoknak felelnek meg, amelyeknél a válasz egyetlen bittel (igen-nem) kifejezhető. Gyakrabban találkozunk olyan problémákkal, amikor a várt eredmény összetett, például amikor egy mennyiség optimumát kell meghatározni, vagy egy halmaz adott tulajdonságú elemeit kell megkeresni. Ezeket az általánosabb feladatokat szokásos (bár nem túl pontos) szóhasználattal *keresési feladatoknak* nevezik. A keresési feladatokat felfoghatjuk valamely

$f : I^* \to I^*$ függvény kiszámításának problémájaként. Gyakran előfordul, hogy f számítása hatékonyan visszavezethető egy felismerési problémára (egy nyelv felismerésére). Erre a jelenségre szeretnénk egy példát mutatni a következőkben.

Álljon az F nyelv azokból az (a, c) pozitív egész párokból, melyekre igaz, hogy a-nak van c-nél nem nagyobb valódi osztója:

$$F = \left\{ (a,c) \;\middle|\; \begin{array}{l} 1 < c \leq a \text{ egészek és van olyan } 1 < b \leq c \text{ egész,} \\ \text{melyre } b \text{ osztója } a\text{-nak} \end{array} \right\}.$$

Állítás: $F \in \mathrm{NP} \cap co\mathrm{NP}$.

Bizonyítás: Az $(a, c) \in F$ ténynek egy jó b érték lesz a tanúja. A bíró ellenőrzi, hogy b osztja-e a-t, és az $1 < b \leq c$ egyenlőtlenségeket. Az $(a, c) \notin F$ ténynek alkalmas tanúját adják az a prímtényezős felbontásában szereplő $p_1, \ldots p_k$ prímek, valamint a p_i számok prímtulajdonságának tanúi (Pratt tétele). A bíró ellenőrzi, hogy a p_i számok prímek-e, a előáll-e ezek hatványainak szorzataként, és végül, hogy teljesülnek-e a $c < p_i$ egyenlőtlenségek. □

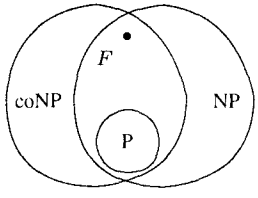
A kapcsolódó keresési feladat a *prímtényezős felbontás* problémája: egy binárisan adott a egésznek keressük a prímosztóit. Ez egyike a legtöbbet vizsgált algoritmikus kérdéseknek. Hatékony algoritmus kidolgozásával már Legendre és Gauss is foglalkoztak – a cél felől tekintve nem sok sikerrel. Ezekből a kutatásokból sok fontos és szép számelméleti eredmény született, de polinom idejű módszert máig sem sikerült találni. Az eddigi leggyorsabb algoritmus D. Shanks *törpe lépés - óriás lépés módszere*; ennek időigénye n bites inputon $c2^{n/4}$. A következő állítás szerint szoros összefüggés van az F és a prímtényezős felbontás algoritmikus nehézsége (bonyolultsága) között. Ha F hatékonyan felismerhető, akkor a prímtényezős felbontás problémája is hatékonyan megoldható.

Tétel: *Ha* $F \in \mathrm{P}$ *igaz lenne, akkor* $\{$*prímtényezős felbontás*$\} \in \mathrm{FP}$ *is igaz lenne.*

Bizonyítás: Meg fogjuk mutatni, hogy ha van gyors (polinom idejű) E eljárásunk F felismerésére, akkor ennek segítségével polinom időben megtalálhatjuk az a input egész prímtényezőit. Először az E eljárás egy hívásával teszteljük, hogy $(a, a-1) \in F$ teljesül-e. Ha nem, akkor megállhatunk, mert a prím, hiszen nincs 1-től és a-tól különböző pozitív osztója. Ellenkező esetben bináris kereséssel meghatározzuk a legkisebb olyan c értéket, melyre $(a, c) \in F$. Ez az E legfeljebb $\log_2 a$ számú hívásával megoldható. Először az $(a, \lceil a/2 \rceil)$ párt teszteljük, ha ez F-beli, akkor megyünk tovább lefelé, stb. A kapott c egész nyilvánvalóan az a legkisebb prímosztója; különben c nem volna minimális. Ezután az $a \leftarrow \frac{a}{c}$ értékkel ismételjük a fenti eljárást.

Az a egy prímosztóját így $O((\log_2 a)^d)$ időben leválasztjuk, ahol $d > 0$ egy állandó, hiszen a feltétel szerint egy E-hívás időigénye polinomiális az input hosszát jelentő $\log a$-ban. Végeredményben az a összes prímosztóját megkapjuk $O((\log a)^{d+1})$ költséggel, mert a prímosztók száma legfeljebb $\log_2 a$. □

Megjegyzés: Valójában nem ismert polinom idejű módszer az F nyelv felismerésére. A szakértők közül sokan úgy vélik, hogy $F \notin$ P. Ebből következne, hogy P $\neq$ NP $\cap$ coNP. Ennek a negatív ténynek, mármint hogy egy feladatra *nincs* hatékony módszer, fontos alkalmazásai vannak a biztonságos (titkos) kommunikáció területén. Vannak olyan kommunikációs protokollok, amelyeknél az üzenetek megfejtésének feladata összehasonlítható az F felismerésének feladatával. Ha a rendszer titkos kulcsait nem ismerő kívülálló meg tudná fejteni az üzeneteket, akkor F-et is kezelni tudná. A következő ábra az F nyelv (vélt) helyét mutatja a bonyolultsági osztályok térképén.



8.5. Karp-redukció, NP-teljesség

Az F nyelv és a prímfelbontás kapcsolatának taglalásakor tulajdonképpen összehasonlítottuk a két feladatot. Az eredmény úgy is fogalmazható, hogy a prímfelbontás problémája - polinom idejű extra munka erejéig - *nem nehezebb*, mint az F nyelv felismerése. Azt is mondhatjuk, hogy a prímfelbontás feladatát *visszavezettük* F felismerésének a feladatára. A kapott felbontó algoritmus futása során többször hívtuk az E eljárást. Most egy ennél szigorúbb visszavezetésfogalmat adunk meg.

Definíció: *Az $f : I^* \to I^*$ leképezés az $L_1 \subseteq I^*$ nyelv* Karp-redukciója *az $L_2 \subseteq I^*$ nyelvre, ha*

1. *Tetszőleges $x \in I^*$ szóra $x \in L_1$ pontosan akkor teljesül, ha $f(x) \in L_2$;*
2. *$f \in FP$, azaz f polinom időben számítható.*

A továbbiakban $L_1 \prec L_2$ jelöli azt a tényt, hogy L_1-nek van Karp-redukciója L_2-re. A definíció első követelménye azt fejezi ki, hogy f hűségesen transzformálja a nyelvbe tartozást. Ha $x \in I^*$ benne van L_1-ben, akkor $f(x)$ benne lesz L_2-ben; és fordítva: ha x nincs L_1-ben, akkor $f(x)$ sincs L_2-ben. A második követelmény szerint az f transzformáció gyorsan számítható. Van olyan csak az f-től függő $c > 0$ állandó, hogy az $f(x)$ szó $|x|^c$ lépésben megkapható x-ből. Ennek folyománya például, hogy $f(x)$ nem lehet túl hosszú: $|f(x)| \leq |x|^c$.

Egy Karp-redukcióval az L_1 felismerésének a problémáját visszavezetjük az L_2 felismerésének a problémájára. A Karp-redukció jelentős segítséget ad, hogy eligazodhassunk az NP-beli nyelvek dzsungelében. Elsősorban annak a megállapítására használható, hogy egy nyelv (döntési feladat) nehéz. A jellemző alkalmazási helyzetben egy eddig ismeretlen L' nyelvet hasonlítunk össze egy már ismert (mondhatni: hírhedt) L nyelvvel. Egy $L \prec L'$ Karp-redukció arra enged következtetni, hogy L' nehéz nyelv, feltéve, hogy ez igaz L-re. Ezt alapozza meg a következő állítás. Látni fogjuk, hogy $L_1 \prec L_2$ esetén L_1 hatékonyan felismerhető, ha van hatékony módszerünk L_2 felismerésére. Az utóbbi eljárást egy inputra csak egyszer kell hívni. Egy $L_1 \prec L_2$ redukció létezéséből arra következtethetünk, hogy L_2 felismerése – polinom idejű extra munka erejéig – legalább olyan nehéz, mint L_1-é.

Első példaként ismertetünk egy Karp-redukciót az irányított Hamilton-kört tartalmazó gráfok nyelvéről az irányítatlan esetre. Egy olyan ötlet lesz segítségünkre, mellyel az irányított élek kódolhatók irányítatlan élekkel.

Állítás: $IH \prec H$.

Bizonyítás: Legyen $G = (V, E)$ egy irányított gráf. Ebből egy $G' = (V', E')$ irányítatlan gráfot készítünk úgy, hogy G ismeretében G' gyorsan megépíthető legyen; továbbá G-ben pontosan akkor legyen irányított Hamilton-kör, ha G'-ben van irányítatlan Hamilton-kör.

Évégből vegyünk G minden $v \in V$ csúcsához egy 2 hosszúságú irányítatlan utat: $v_{be} - v_* - v_{ki}$. Az $u \to v \in E$ élnek feleltessük meg az $u_{ki} - v_{be}$ élet. Formálisan:

$$\begin{aligned} V' &= \{v_{be}, v_*, v_{ki} \mid v \in V\}, \\ E' &= \{(v_{be}, v_*), (v_*, v_{ki}) \mid v \in V\} \cup \{(u_{ki}, v_{be}) \mid u \to v \in E\}. \end{aligned}$$

Ha G-nek n csúcsa és e éle van, akkor G'-nek $3n$ csúcsa és $2n+e$ éle lesz. Világos, hogy G ismeretében G' polinom időben, azaz $(n+e)^c$ lépésben megkapható. A Karp-redukció második (hatékonysági) követelménye tehát teljesül.

Nézzük most az első feltételt. G minden F irányított Hamilton-körének természetesen megfeleltethető G' egy F' Hamilton-köre. Az F egy $u \to v$ élének az F'-ben az $u_* - u_{ki} - v_{be} - v_*$ út felel meg. Ha tehát $G \in$ IH, akkor $G' \in$ H. A fordított irányú követelményhez tegyük fel, hogy G'-ben van egy $F' \subseteq E'$ Hamilton-kör. Járjuk be ezt a kört egy csillagos pontjából kezdve úgy, hogy a *ki* indexű szomszéd felé lépjünk először. Ekkor a körön a bejárás szerint szomszédos két csillagos pont közötti útdarab csak $u_* - u_{ki} - v_{be} - v_*$ alakú lehet. Ebből pedig azonnal adódik, hogy az

$$F = \{u \to v \mid \{u_{ki}, v_{be}\} \in F'\}$$

élhalmaz irányított Hamilton-köre G-nek. A $G \mapsto G'$ megfeleltetés tényleg Karp-redukció. □

A $G \mapsto G'$ leképezést szemlélve elmondhatjuk, hogy IH felismerése nem nehezebb, mint a H nyelvé. Az IH egy G bemenetéből a leképezéssel olcsón kaphatjuk a H feladat G' bemenetét. Ha a ($G' \in$ H?) kérdést meg tudjuk válaszolni, akkor megkaptuk a választ a ($G \in$ IH?) kérdésre is. Most pedig nézzük mindezt általánosabban.

Állítás:
1. Ha $L_1 \prec L_2$ *és* $L_2 \in$ P, *akkor* $L_1 \in$ P.
2. Ha $L_1 \prec L_2$ *és* $L_2 \in$ NP *akkor* $L_1 \in$ NP.
3. Ha $L_1 \prec L_2$, *akkor* $\overline{L}_1 \prec \overline{L}_2$, *ahol* $\overline{L}_i = I^* \setminus L_i$.
4. Ha $L_1 \prec L_2$ *és* $L_2 \in$ *co*NP, *akkor* $L_1 \in$ *co*NP.
5. Ha $L_1 \prec L_2$ *és* $L_2 \in$ NP $\cap$ *co*NP, *akkor* $L_1 \in$ NP $\cap$ *co*NP.

Bizonyítás: Legyen $f : I^* \to I^*$ az L_1 Karp-redukciója L_2-re. A definíció szerint $f \in$ FP, mondjuk $f \in FTIME(n^k)$. Jelöljön az $x \in I^*$ egy input szót, melyre szeretnénk eldönteni, hogy $x \in L_1$ teljesül-e, és legyen n az x hossza. Vegyük ezután sorra az állításokat.
1. A célunk itt az, hogy polinom idejű algoritmust adjunk az L_1 felismerésére. Az f függvény mellett még használhatjuk az $L_2 \in$ P feltételt, mondjuk legyen $L_2 \in TIME(n^l)$. Először kiszámítjuk az $f(x)$ szót, ennek időigénye legfeljebb $c_1 n^k$, amiből $|f(x)| \leq c_1 n^k$ is következik. A második lépésben L_2 felismerő algoritmusával eldöntjük, hogy $f(x) \in L_2$ igaz-e. Ennek az időigénye legfeljebb $c_2(c_1 n^k)^l$. A Karp-redukció 1. tulajdonsága szerint az ($x \in L_1$?) kérdésre adandó válasz megegyezik az ($f(x) \in L_2$?) kérdésre adott válasszal, tehát az x inputot pontosan akkor fogadjuk el, ha a második lépésben $f(x)$-et elfogadtuk. A számítás összideje $O(n^{kl})$. Ezzel igazoltuk, hogy $L_1 \in$ P.
2. Az $f(x) \in L_2$ tény egy y tanúja megfelelő lesz az $x \in L_1$ tanújának is, és az L_2-höz tartozó bíró egy kis módosítással jó lesz az L_1 bírójának is. Ha ugyanis

$|y| \leq |f(x)|^c$, akkor $|y| \leq c_1^c |x|^{kc}$, tehát y az x-hez képest polinomiális méretű. Az L_1 bírója az (x, y) bemenet esetén először kiszámítja $f(x)$-et, majd átadja az $(f(x), y)$ párt az L_2 bírájának. Az L_1 bírója pontosan akkor fogadja el az (x, y) párt, ha az L_2 bírója elfogadja az $(f(x), y)$ párt. Ez a Karp-redukció definíciója szerint éppen azt jelenti, hogy $x \in L_1$. Az is világos, hogy L_1 bírójának az időigénye becsülhető n egy polinomjával.

3. Az $L_1 \prec L_2$ redukciót megvalósító f jó, hiszen a definíció 1. követelménye így is fogalmazható: tetszőleges $x \in I^*$ szóra $x \notin L_1$ pontosan akkor teljesül, ha $f(x) \notin L_2$.

4. Közvetlenül adódik az előző két állításból.

5. A 2. és a 4. állítások következménye. □

A megelőző állítások szerint ha L_2 benne van valamelyik nevezetes P-t tartalmazó nyelvosztályban, akkor L_1 is. Ez érvényes a PSPACE és EXPTIME osztályokra is; az első állítás gondolatmenete működik ezekben az esetekben is.

Feladat: Igazoljuk, hogy a $\prec$ reláció tranzitív: ha $L_1 \prec L_2$ és $L_2 \prec L_3$, akkor $L_1 \prec L_3$ is teljesül.

A Karp-redukció fogalma lehetőséget ad arra, hogy könnyebb-nehezebb viszonyt állapítsunk meg nyelvek között. Az NP osztály egyik érdekes tulajdonsága, hogy vannak benne ilyen értelemben *legnehezebb* nyelvek (feladatok). Ezek az NP-*teljes nyelvek*.

Definíció: *Az $L \subseteq I^*$ nyelv* NP-*teljes, ha*

1. $L \in$ NP,

2. tetszőleges $L' \in$ NP nyelv esetén létezik $L' \prec L$ Karp-redukció.

Egy NP-teljes nyelv tehát legalább olyan nehéz, mint bármely más NP-beli nyelv. Ha egy ilyen nyelvről kiderülne, hogy P-beli (coNP-beli), akkor ugyanez igaz lenne minden NP-beli nyelvre. Az NP-beli nyelveknek és a megfelelő (tanú)keresési feladatoknak a kapcsolata miatt az NP-teljes nyelveket szokás még *univerzális kereső feladatoknak* is nevezni. A P $\neq$ NP sejtés szellemében úgy gondoljuk, hogy az NP-teljes nyelvekre nem léteznek polinom idejű felismerő módszerek. Ez nem bizonyított tény, de óriási mennyiségű közvetett bizonyíték szól mellette: a gyors algoritmusokat kereső kísérletek egyhangú kudarca. Ilyen értelemben az NP-teljes feladatok nehézségét a tudomány mai állása mellett *tapasztalati ténynek* tekintjük. Azt is hangsúlyozzuk, hogy az NP-teljes feladatok nem megoldhatatlanok. Korábban láttuk, hogy

$$\text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME},$$

tehát az NP-beli problémák legnehezebbjei is legyűrhetők exponenciális idejűnél nem rosszabb algoritmussal.

8.6. A SAT nyelv és a Cook–Levin-tétel

Fiam, tanácsolom tehát,
próbáld először a logikát.
...
hogy ezután már ne ingatag
járja útját a gondolat,
fel s le bolyongva botorul,
lidércként keresztül-kasul.

JOHANN WOLFGANG GOETHE
(Mefisztó tanácsai a Diáknak.)

A következőkben egy alapvető fontosságú NP-teljes nyelvet ismertetünk. Szükségünk lesz ehhez a *Boole-formula* fogalmára. Egy Boole-formula a 0 („hamis"), 1 („igaz") logikai konstansokból, 0-1 értékű változókból (mondjuk ezek $x_1, \ldots, x_n$) és a változók $\overline{x}_1, \ldots, \overline{x}_n$ negáltjaiból a $\wedge$ („és"), illetve a $\vee$ („vagy") műveleti jelekkel és zárójelekkel felépített kifejezés. Szokás a változókat és negáltjaikat közös névvel illetve *literál*oknak nevezni. Egy Boole-formula változóinak (logikai) értékeket adhatunk. A változók ilyen *kiértékelése* után a kézenfekvő módon beszélhetünk a formula értékéről, ami 0 vagy 1 lesz. A ϕ Boole-formula *kielégíthető*, ha van a változóinak olyan kiértékelése, melynél ϕ értéke 1.

Példa: Az $(x_1 \vee x_2) \wedge \overline{x}_3$ formula kielégíthető ($x_1 = 1$, $x_2 =$ értéke tetszőleges, $x_3 = 0$). Az $x_1 \wedge \overline{x}_1$ formula pedig nem kielégíthető.

Jelölje SAT a kielégíthető Boole-formulák nyelvét (a jelölés a *kielégíthetőség* szó angol megfelelőjéből, a *satisfiability* szóból származik). Ebben az esetben sem túl fontos, hogy miként kódoljuk a formulákat bináris szavakká. Egyik lehetséges eljárás, hogy a formula elemeinek (változók, állandók, műveleti jelek, zárójelek) kódjait fűzzük össze a kézenfekvő módon. A SAT nyelv benne van az NP osztályban: egy formula kielégíthetőségének tanúja egy jó kiértékelés. A bíró kiszámolja a formula értékét az adott kiértékelésnél. A következő eredmény szerint a SAT a legnehezebbek egyike az NP osztályban.

Tétel (S. A. Cook, L. Levin, 1971): *A SAT egy* NP*-teljes nyelv.*

Bizonyítás: (Vázlat) A SAT $\in$ NP állítással már foglalkoztunk. A tétel érdemi részéhez azt kell igazolni, hogy tetszőleges $L \in$ NP nyelvre létezik egy $L \prec$ SAT Karp-redukció.

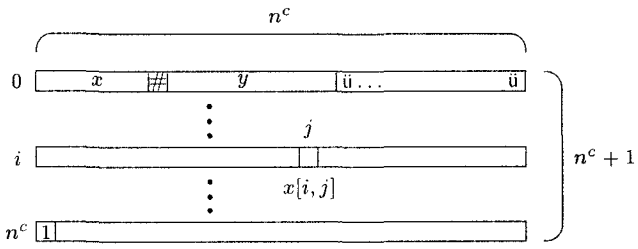
Mit jelent egy ilyen redukció? Az $(x \in L?)$ kérdés tetszőleges $x \in I^*$ inputjához meg kell adnunk egy ϕ Boole-formulát, mely pontosan akkor kielégíthető, ha

$x \in L$. Ennek az $x \mapsto \phi$ leképezésnek hatékonyan (azaz polinom időben) számíthatónak kell lennie. Egyszerűbben fogalmazva: az x inputra a ϕ formulát meg kell tudnunk adni $O(|x|^k)$ lépésben.

A tanú-tétel szerint az $L \in$ NP feltétel azt jelenti, hogy van olyan $L_1 \in$ P nyelv (a bíró nyelve) és $d > 0$ konstans, hogy

$$x \in L \text{ pontosan akkor teljesül,}$$
$$\text{ha van olyan } y \in I^*, \text{ melyre } |y| \leq |x|^d, \text{ és } (x, y) \in L_1.$$

Legyen M egyszalagos, polinomkorlátos TG, mely az L_1 nyelvet ismeri fel. Az egyszerűség kedvéért feltesszük még, hogy M szalagjelei 0,1 és ü. Tegyük fel, hogy az (x, y) összetett bemenet az M szalagján $x\#y$ alakban van, ahol $\#$ egy elválasztó bitsorozat. Jelölje n az x input szó hosszát, és tegyük fel, hogy $n^{1+d} + |\#|$ hosszúságú inputokon M legfeljebb n^c lépést tesz. Ilyen c kitevő létezik, mert M polinomkorlátos. Esetleg az M egy kis módosítása árán még azt is feltehetjük, hogy az M pontosan akkor fogadja el az inputot, ha megállás után a szalag első bitje 1 (ha elfogadó állapotba kerül, 1-et ír a szalag elejére). Az M működését az $x\#y$ inputon az alábbi tártérkép mutatja. Az i-edik sor M szalagját ábrázolja az i-edik lépés megtétele utáni helyzetben.



A ϕ Boole-formulával kapcsolatos követelményeket úgy is fogalmazhatjuk, hogy ϕ hatékonyan számítható x-ből, és pontosan akkor kielégíthető, ha van olyan rövid y, melyre M az $x\#y$ inputot elfogadja. A ϕ konstrukciójához szükségünk lesz a gép pillanatnyi helyzeteit leíró 0-1 értékű változókra: $(i = 0, \ldots, n^c;$ $j = 1, \ldots, n^c;$ $s = 0, 1, \ldots, |Q| - 1)$

$$0x[i,j] = \begin{cases} 1 & \text{ha az } i\text{-edik lépés után a } j\text{-edik cella tartalma 0} \\ 0 & \text{különben} \end{cases}$$

$$1x[i,j] = \begin{cases} 1 & \text{ha az } i\text{-edik lépés után a } j\text{-edik cella tartalma 1} \\ 0 & \text{különben} \end{cases}$$

$$\ddot{u}x[i,j] = \begin{cases} 1 & \text{ha az } i\text{-edik lépés után a } j\text{-edik cella tartalma ü} \\ 0 & \text{különben} \end{cases}$$

$$f[i,j] = \begin{cases} 1 & \text{ha az } i\text{-edik lépés után a fej } j\text{-edik cellán áll} \\ 0 & \text{különben} \end{cases}$$

$$q[i,s] = \begin{cases} 1 & \text{ha az } i\text{-edik lépés után } M \text{ belső állapota } q_s \\ 0 & \text{különben.} \end{cases}$$

Az M működését szeretnénk leírni ezekből a változókból épített Boole-formulákkal. Többféle tényt kell ilyen módon megfogalmaznunk:

1. Le kell írnunk olyan állításokat, hogy a szalag egy mezőjén minden időpontban éppen egy szalagjel van; a fej minden időpontban a szalag egyetlen celláján van; a gép minden időpontban egyetlen állapotban van. Példaként formalizáljuk az első típusba eső, a szalagjelekkel foglalkozó állításokat: minden i, j ($0 \leq i \leq n^c$, $1 \leq j \leq n^c$) párra

$$(0x[i,j] \vee 1x[i,j] \vee \ddot{u}x[i,j]) \wedge$$
$$\wedge(\overline{0x[i,j]} \vee \overline{1x[i,j]}) \wedge (\overline{0x[i,j]} \vee \overline{\ddot{u}x[i,j]}) \wedge (\overline{1x[i,j]} \vee \overline{\ddot{u}x[i,j]}).$$

Összesen $(n^c+1)n^c$ ilyen formulánk lesz. Hasonlóan fogalmazhatjuk meg a fej helyzetével és a belső állapottal kapcsolatos állításokat.

2. Le kell írnunk a gép átmeneteit is (lényegében a δ függvényt). Ezt is példával mutatjuk be. Tegyük fel, hogy a q_s belső állapotra és az 1 szalagjelre $\delta(q_s, 1) = (q_k, 0, bal)$ (M a q_s állapotból 1-es szalagjel olvasása esetén a q_k állapotba megy át, 0-t ír a fej alatti cellába, majd a fej balra lép). Ezt a változóinkkal így fejezhetjük ki: tetszőleges $0 \leq i < n^c$, $1 \leq l \leq n^c$ esetén

$$((q[i,s] \wedge 1x[i,l] \wedge f[i,l]) \longrightarrow (q[i+1,k] \wedge 0x[i+1,l] \wedge f[i+1,l-1]).$$

Emlékeztetőül: az $u \to v$ (ha u igaz, akkor v is igaz) kifejezés az $\overline{u} \vee v$ Boole-formula egy másik írásmódja. Összesen $O(n^{2c})$ ilyen formulánk lesz, mert a δ táblázat mérete független n-től.

Az előbbi formulák leírják, hogy mi és hogyan változik. Azt is meg kell mondanunk, hogy más változás nincs; a szalag azon cellái, amelyek felett *nincs* a fej, megtartják régi értéküket.

$$\overline{f[i,l]} \longrightarrow (0x[i,l] \leftrightarrow 0x[i+1,l])$$

Ugyanilyen formulák kellenek még az 1 és az ü jelekkel is. Ezek együttes száma szintén $O(n^{2c})$.

3. A gép kezdő helyzetének (a q_0 belső állapotban van, a fej az első cellára mutat) a leírása így néz ki:

$$q[0,0] \wedge f[0,1]$$

Legyen ϕ_m az m-edik lépés utáni helyzetet leíró formulák $\wedge$-e (konjunkciója). A $\psi = \phi_0 \wedge \ldots \wedge \phi_{n^c}$ formula azt fejezi ki, hogy a tártérkép a 0. sor alatt úgy van kitöltve, ahogy azt M tenné.

Jelölje $\psi(x)$ azt a formulát, melyet úgy kapunk ψ-ből, hogy $\wedge$-el hozzákapcsoljuk azt az állítást, mely szerint az input # előtti része éppen az x szóval egyezik meg (pl. $\psi(x) = \psi \wedge 0x[0,1] \wedge 1x[0,2] \wedge 0x[0,3] \ldots$, ha $x = 010\ldots$). Végül legyen

$$\phi = 1x[n^c, 1] \wedge \psi(x).$$

Az $f[i,j], q[i,s], 0x[i,j], 1x[i,j], \ddot{u}x[i,j]$ változók valamely kiértékelésénél a ϕ formula értéke pontosan akkor lesz igaz, ha van M-nek olyan legfeljebb n^c lépéses elfogadó számítása, melynél az input első fele x. A ϕ kiértékelésekor a tényleges szabadsági fok az y súgásnak megfelelő $0x[0,j], 1x[0,j], \ddot{u}x[0,j]$ változók értékadásában van. A ϕ tehát pontosan akkor lesz kielégíthető formula, ha az inputnak a # utáni része kitölthető úgy, hogy ebből a helyzetből indulva M elfogadó állapotban álljon meg a munkája végén. Ez pedig éppen azt jelenti, hogy $x \in L$. A vázolt módon a ϕ formula x és M ismeretében polinom időben megadható, tehát a kapott $x \mapsto \phi$ függvény tényleg egy $L \prec$ SAT Karp-redukció. □

A bizonyítás szerint a gép működését le lehet írni egyetlen Boole-formulával. Eme figyelemre méltó tény a logika kifejező erejét mutatja. A logikának ez a nagyon erős leíró képessége fontos szerepet játszik a számítástudomány más területein is; említhetjük itt a logikai programozás nyelveinek hajlékonyságát, vagy a modern adatbázisok lekérdező felületeinek (mint pl. az SQL) gazdag és emberi léptékű kifejező erejét.

8.7. További NP-teljes feladatok

Ebben a részben további NP-teljes feladatokkal ismerkedünk meg. Felvetődik a kérdés, hogy miért foglalkozunk ezekkel ilyen behatóan, ha ezekre – széles körben elfogadott sejtés szerint – nem adható hatékony algoritmus. Azért szentelünk ennyi figyelmet az NP-teljes feladatoknak, mert szinte mindenütt ott vannak. Tapasztalati tény, hogy a fontos, érdekes algoritmikus problémák között igen gyakran találunk ilyeneket (a szakirodalomban dokumentált NP-teljes feladatok száma jóval ezer felett van). Az is gyakori jelenség, hogy egy kezelhető (mondjuk P-beli) probléma paramétereinek látszatra ártatlan változtatásával már NP-teljes feladatot kapunk. Ha egy problémáról tudjuk, hogy NP-teljes, akkor azt is tudjuk, hogy

nem érdemes sokat fáradozni hatékony algoritmus keresésén. Néhány lehetséges kerülőúttal a következő fejezetben foglalkozunk. Az NP-teljességgel kapcsolatos ismeretek tehát segítenek eligazodni abban, hogy egy adott algoritmikus feladatra milyen (mennyire gyors, milyen pontos) megoldás keresése lehet reális célkitűzés.

Az alábbi egyszerű állítás segítségével igazolhatjuk nyelvek NP-teljességét.

Állítás: *Ha az* L_1 *nyelv* NP*-teljes,* $L_2 \in$ NP *és* $L_1 \prec L_2$*, akkor* L_2 *is* NP*-teljes.*

Bizonyítás: Mivel $L_2 \in$ NP, elég megmutatni, hogy $L' \prec L_2$ minden NP-beli L' nyelv esetén. A feltételek szerint $L' \prec L_1$ és $L_1 \prec L_2$. Legyenek f és g a redukciókat megvalósító FP-beli függvények. Ekkor $g \circ f$ (az a leképezés, mely az x szóhoz a $g(f(x))$ szót rendeli) egy $L' \prec L_2$ Karp-redukció. Teljesül tehát a definíció második követelménye is, így L_2 tényleg NP-teljes. □

A Cook–Levin-tétel után szinte minden NP-teljességet bizonyító gondolatmenet az előző állításból adódó utat használja. Nem kell már *minden* NP-beli nyelvet az L_2-re redukálni; elég ezt megtenni *egyetlen* NP-teljes L_1 nyelvvel.

Ha az L_2 nyelvről csak azt tudjuk, hogy van olyan NP-teljes L_1 nyelv, melyre $L_1 \prec L_2$, akkor L_2-t NP-*nehéz* nyelvnek nevezzük. Az előző állítás szerint L_2 pontosan akkor NP-teljes, ha NP-beli és ugyanakkor NP-nehéz is.

8.7.1. Konjunktív normálformájú formulák kielégíthetősége és a 3-SAT

A ϕ Boole-formula *konjunktív normálformájú*, ha $\phi = \phi_1 \wedge \cdots \wedge \phi_k$ alakú, ahol a ϕ_i formulákban literálok szerepelnek kizárólag $\vee$ jelekkel összekapcsolva. A ϕ_i formulákat ekkor a ϕ *tagjainak* nevezzük. Az ítéletkalkulus azonosságai (disztributivitás, De Morgan szabályok) segítségével könnyen látható, hogy minden formulához van egy vele ekvivalens (minden kiértékelésre ugyanazt az értéket adó) konjunktív normálformájú formula.

Feladat: Mutassuk meg, hogy egy Boole-formula konjunktív normálformájú ekvivalensének kiszámítására nincs polinom idejű algoritmus. (Előfordulhat, hogy az eredmény mérete nem polinomiális az input méretéhez képest.)

Ha a ϕ_i formulákban legfeljebb k literál szerepel, akkor ϕ-t k*-konjunktív normálformájú* formulának (szokásos rövidítéssel k-CNF-nek) nevezzük.

Példa: Az alábbi formula egy 4-CNF:

$$(\overline{x}_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_4 \vee \overline{x}_7 \vee \overline{x}_{18}) \wedge (x_9 \vee \overline{x}_{10} \vee \overline{x}_{13}).$$

Jelölje k-SAT a kielégíthető k-CNF-ekből álló nyelvet. Egy k-CNF kielégíthetőségének a tanúja lehet egy kielégítő kiértékelés. A tanú a formula értékének kiszámításával hatékonyan ellenőrizhető, tehát k-SAT $\in$ NP. Ez a nyelv könnyű, ha $k = 2$:

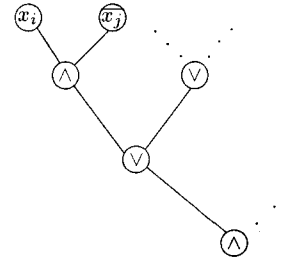
Feladat: Mutassuk meg, hogy 2-SAT $\in$ P. (Legyen ϕ egy 2-CNF. Jelölje ϕ_0, illetve ϕ_1 azokat a 2-CNF-eket amelyeket ϕ-ből az $x_1 = 0$, illetve $x_1 = 1$ helyettesítések után kaptunk. A ϕ_i-ről gyorsan kideríthetjük a következők *valamelyikét*:
1. ϕ_i kielégíthető,
2. ϕ_i nem kielégíthető,
3. ϕ_i kielégíthetősége ekvivalens egy ϕ_i' 2-CNF kielégíthetőségével, aminek tagjai ϕ-nek is tagjai, és amelyben az x_1 változó nem szerepel.)

Ha eggyel több literált engedünk meg a zárójelekben, akkor a helyzet alaposan megváltozik; a könnyű problémából nehéz lesz.

Tétel: *A 3-SAT nyelv* NP-*teljes.*

Bizonyítás: Az előzőek alapján elég megmutatni, hogy SAT $\prec$ 3-SAT. Olyan polinom időben számítható megfeleltetést kell adnunk, mely egy ϕ formulához (a SAT bemenetéhez) egy ψ 3-CNF-et rendel úgy, hogy ϕ akkor és csak akkor kielégíthető, ha ψ kielégíthető.

Legyenek $x_1, x_2, \ldots, x_n$ a ϕ formulában szereplő változók, és írjuk fel a ϕ egy kifejezésfáját, majd számozzuk meg a fa csúcsait, minden csúcshoz egyedi sorszámot rendelve. A fa csúcsai a ϕ részformuláinak felelnek meg. A csúcs feletti leveleknek értéket adva a részformula kiszámítható; e megfeleltetés révén beszélhetünk a csúcshoz tartozó értékről. Az i sorszámú csúcshoz a (ϕ változóitól és egymástól is különböző) z_i Boole-változót rendeljük.



Ezen felül az i sorszámú csúcshoz rendelünk még egy ϕ_i Boole-formulát. Egyszerű szavakba foglalva: a ϕ_i formula azt fogja kifejezni, hogy a z_i értéke legyen

az i-edik csúcshoz tartozó érték az x_j változók egy adott kiértékelésekor. Innen a ϕ_i formulák definíciója már kézenfekvő (az $u \leftrightarrow v$ kifejezés az $(\overline{u} \vee v) \wedge (u \vee \overline{v})$, azaz a logikai ekvivalencia rövidítése):
Ha az i-edik csúcs levél, amelynek tartalma az u literál (x_j, vagy $\overline{x}_j$), akkor legyen $\phi_i : z_i \leftrightarrow u$.
Ha az i-edik csúcsban $\wedge$ van, és a fiai a k-adik és l-edik csúcsok, akkor legyen $\phi_i : z_i \leftrightarrow (z_k \wedge z_l)$.
Ha az i-edik csúcsban $\vee$ van, és a fiai a k-adik és l-edik csúcsok, akkor legyen $\phi_i : z_i \leftrightarrow (z_k \vee z_l)$.

Jelölje m a kifejezésfa gyökerének a sorszámát. Következő lépésként vegyünk mindegyik ϕ_i-hez egy ekvivalens konjunktív normálformájú formulát. Legyen ez ψ_i. A ϕ_i formulában legfeljebb 3 változó van (bizonyos x-ek és bizonyos z-k), ezért ψ_i egy 3-CNF. Legyen $\psi = (\wedge\psi_i) \wedge z_m$. A ψ egy 3-CNF, lévén ilyenek konjunkciója. A ψ értéke a változóinak a kiértékelésekor pontosan akkor lesz 1 (vagyis „igaz"), ha ennél a kiértékelésnél mindegyik ψ_i és z_m is igaz. Mindez egyenértékű azzal, hogy minden i-re a z_i változó értéke megegyezik a ϕ kifejezésfa szerinti számításánál az i-edik csúcshoz tartozó részeredménnyel, és (a z_m tag jelenléte miatt) a ϕ értéke is 1. Úgy is fogalmazhatunk, hogy az x_i értékek tetszőleges megválasztása után a ψ_i formulák egyértelműen igazzá tehetők: a z_i változónak az i-edik csúcsnál fellépő értéket kell adni. A ψ ezek szerint pontosan akkor kielégíthető, ha ϕ is ilyen tulajdonságú. Más szóval $\phi \in$ SAT pontosan akkor teljesül, ha $\psi \in$ 3-SAT. Végül megjegyezzük, hogy a ψ előállítására adott recept RAM-on lineáris időben implementálható. A $\phi \mapsto \psi$ megfeleltetés tényleg egy Karp-redukció. □

8.7.2. 3 színnel színezhető gráfok

Korábban már láttuk, hogy a 3-SZÍN, a három színnel színezhető gráfok nyelve az NP osztályban van. A következő állítás szerint ez a nyelv is az NP legnehezebbjei közül való.

Tétel: *A 3-SZÍN nyelv* NP-*teljes.*

Bizonyítás: Elegendő megadni egy 3-SAT $\prec$ 3-SZÍN Karp-redukciót. Pontosabban fogalmazva, egy tetszőleges ψ 3-CNF-hez építenünk kell egy G gráfot úgy, hogy $\psi \in$ 3-SAT pont akkor igaz, ha $G \in$ 3-SZÍN. A konstrukciónak hatékonynak, polinom időben elvégezhetőnek kell lennie.

Tegyük fel, hogy a ψ formulában m változó és z nyitózárójel van: $\psi = (\psi_1) \wedge \ldots \wedge (\psi_z)$. Feltehető, hogy minden egyes ψ_i pontosan 3 tagú (például $x_1 \vee x_2$ helyett vehetjük az $x_1 \vee x_2 \vee x_2$ formulát). A G gráfnak $2m + 5z + 2$

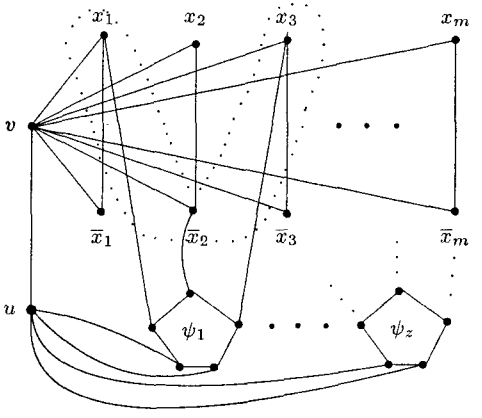
csúcsa lesz. Van két különálló csúcs: u és v. A ψ-beli x_i változónak és negáltjának is megfelel egy-egy csúcs; ezek jele x_i, illetve $\overline{x}_i$. Végül a ψ_j részformulának 5 csúcs fog megfelelni; ezek $\psi_j^1, \ldots, \psi_j^5$.

$$V(G) = \cup_{i=1}^{m}\{x_i, \overline{x}_i\} \cup \cup_{j=1}^{z}\{\psi_j^1, \ldots, \psi_j^5\} \cup \{v, u\}.$$

A G élei a következők:

$$E(G) = \left\{ \begin{array}{ll} (v,u), & \\ (v,x_i), & i = 1,\ldots,m, \\ (v,\overline{x}_i), & i = 1,\ldots,m, \\ (x_i,\overline{x}_i), & i = 1,\ldots,m, \\ (u,\psi_j^4), & j = 1,\ldots,z, \\ (u,\psi_j^5), & j = 1,\ldots,z, \\ (\psi_j^k,\psi_j^{k+1}), & j = 1,\ldots,z,\ k \text{ modulo } 5 \text{ fut}, \\ (\psi_j^k,x_i), & \text{ahol } \psi_j\ k\text{-adik tagja } x_i,\ j = 1,\ldots,z,\ k = 1,2,3, \\ (\psi_j^k,\overline{x}_i), & \text{ahol } \psi_j\ k\text{-adik tagja } \overline{x}_i,\ j = 1,\ldots,z,\ k = 1,2,3. \end{array} \right\}$$

A G gráfban ötszögek felelnek meg a ψ_j formuláknak. Ezek „alsó" pontjai u-val vannak összekötve, a felső pontjai pedig egy-egy a ψ_j-ben előforduló literállal. A következő ábra azt az esetet szemlélteti, amikor $\psi_1 = x_1 \vee \overline{x}_2 \vee x_3$.



A G mérete lineáris ψ méretében, és az is látható, hogy G könnyen (polinom időben) felépíthető ψ ismeretében. Igazolnunk kell még, hogy

$$G \in \text{3-SZÍN pontosan akkor, ha } \psi \in \text{3-SAT}.$$

Vegyük először G egy 3-színezését. Nevezzük v színét pirosnak, u-ét sárgának, a harmadik szín legyen a zöld. Vegyük észre, hogy egy literál színe csak zöld vagy sárga lehet a v-vel összekötő él miatt. A ψ formula y literáljának adjuk az *igaz* értéket, ha a G-ben az y csúcs zöld; ha y sárga színű, akkor az értéke legyen *hamis*. Ez megfelel az x_i változók egy kiértékelésének. Az $(x_i, \overline{x}_i)$ él megléte miatt az x_i és $\overline{x}_i$ csúcsok közül az egyik sárga, a másik zöld. Megmutatjuk, hogy így a ψ egy kielégítő kiértékelése adódik. Ez egyenértékű azzal az állítással, hogy mindegyik ψ_i-ben van zöld literál. Ez pedig igaz, hiszen ellenkező esetben a ψ_i-hez tartozó ötszög csúcsaira csak két szín volna megengedett (piros és zöld); az ötszög viszont nem színezhető két színnel.

A fordított irányú állításhoz vegyük ψ egy kielégítő kiértékelését. Feleltessük meg, mint előbb, a literálok *igaz* értékének a zöld színt, a *hamis* értéknek pedig a sárga színt. Színezzük v-t pirosra, u-t pedig sárgára. Az eddig kapott részleges színezéssel nincs baj, már csak a ψ_i formulákhoz tartozó ötszögeket kell kiszínezni. Egy ilyen ötszög minden csúcsára tiltott a zöld és a sárga színek egyike. Az alsó két pontra a sárga, a felső csúcsokra viszont nem mindenütt a sárga, hiszen ψ_i-ben van zöld literál. Az alábbi egyszerű feladat szerint ilyenkor az ötszög kiszínezhető. □

Feladat: Az ötpontú kör kiszínezhető 3 színnel úgy is, hogy minden egyes csúcsára kijelölünk egy tiltott színt, de nem minden csúcsra ugyanazt a színt.

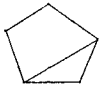
Feladat: Adjunk lineáris (uniform) költségű algoritmust egy gráf két színnel való színezhetőségének eldöntésére.

8.7.3. Maximális méretű független pontrendszer gráfokban

A G gráf csúcsainak S részhalmaza *független halmaz*, ha S-beli pontok között nem fut G-ben él. A G gráf fontos jellemzője a legnagyobb méretű független ponthalmazának elemszáma. Ennek a kiszámítása nehéz algoritmikus probléma, amint azt a MAXFTLEN nyelv mutatja.

$$\text{MAXFTLEN} = \left\{ (G,k) \;\middle|\; \begin{array}{l} G \text{ egy gráf, } k \in \mathbb{Z}^+, \text{ és} \\ G\text{-nek van } k \text{ elemű független csúcshalmaza} \end{array} \right\}.$$

Példa: Az ábrán látható G gráf esetén $(G,1), (G,2) \in$ MAXFTLEN, $(G,3) \notin$ MAXFTLEN.

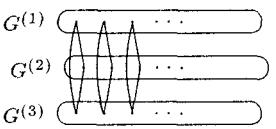


Tétel: *A MAXFTLEN nyelv* NP*-teljes.*

Bizonyítás: Nyilvánvaló, hogy MAXFTLEN $\in$ NP: a $(G,k) \in$ MAXFTLEN ténynek tanúja lehet egy k-elemű $S \subseteq V(G)$ független csúcshalmaz. Elegendő ezután megadni egy 3-SZÍN $\prec$ MAXFTLEN Karp-redukciót. Legyen a G gráf a 3-SZÍN egy inputja. Olyan (G_1,k) párt kell hatékonyan megadni (G_1 egy gráf, k egy pozitív egész), melyre érvényes a következő:

$$G \in \text{3-SZÍN pontosan akkor, ha } (G_1,k) \in \text{MAXFTLEN.} \qquad (*)$$

A G_1 gráfot a következőképpen kapjuk a G-ből: veszünk három példányt G-ből, ezek legyenek $G^{(1)}, G^{(2)}, G^{(3)}$, és kössük össze élekkel a G azonos csúcsainak megfelelő pontokat. Így a $v \in V(G)$ csúcs három példánya $v^{(i)} \in G^{(i)}$ $(i = 1,2,3)$ egy háromszöget feszít G_1-ben.



A G_1 csúcsainak és éleinek számára a következőket kapjuk:

$$|V(G_1)| = 3|V(G)| \text{ és } |E(G_1)| = 3|V(G)| + 3|E(G)|.$$

Legyen $k = |V(G)|$. A G ismeretében a (G_1,k) pár hatékonyan megadható. A tétel bizonyításához ezután elegendő a $(*)$ állítást igazolni.

Tekintsük először a G egy 3 színnel való színezését, ahol a színek 1,2,3. Álljon a $H \subseteq V(G_1)$ halmaz azokból a $v^{(i)}$ pontokból, melyekre $v \in V(G)$ színe i. Másképp fogalmazva: egy i színű G-beli pontnak az i-edik szinten levő példányát vesszük H-ba. A színezés definíciójából azonnal kapjuk, hogy a H halmaz független, és $|H| = k$. Van tehát G_1-ben k független pontból álló halmaz.

Fordítva, legyen H egy k elemű független halmaz G_1-ben. Nyilvánvaló, hogy ekkor H minden oszlopból ($v^{(1)}, v^{(2)}, v^{(3)}$ hármasból) pontosan egy csúcsot tartalmaz. Legyen a $v \in V(G)$ csúcs színe i, ha $v^{(i)} \in H$. Ez jó színezése lesz G-nek hiszen ha az u és v csúcsok azonos színűek, akkor $u^{(i)}, v^{(i)} \in H$ valamely i-re. Ekkor pedig H függetlensége miatt $(u,v) \notin E(G)$. □

A MAXFTLEN nyelvre vonatkozó tételből könnyen adódik két rokon gráfelméleti probléma NP-teljessége. Legyen

$$\text{MAXKLIKK} = \{(G,k) |\ G\text{-ben van } k \text{ pontú teljes részgráf}\}.$$

Következmény: *A MAXKLIKK nyelv* NP*-teljes.*

Bizonyítás: Nyilvánvaló, hogy MAXKLIKK $\in$ NP. Az $X \subseteq V(G)$ csúcshalmaz pontosan akkor független a G gráfban, ha X klikk (teljes részgráfot feszít) a G gráf $\overline{G}$ komplementer gráfjában. (A $\overline{G}$ pontjai ugyanazok, mint G pontjai, és (u, v) él $\overline{G}$-ben pontosan akkor, ha (u, v) nem él G-ben.) Ebből azonnal adódik, hogy a $(G, k) \mapsto (\overline{G}, k)$ megfeleltetés egy MAXFTLEN $\prec$ MAXKLIKK Karp-redukció. □

A G gráf csúcsainak egy X részhalmaza *éllefogó halmaz*, ha a G bármely élének van X-beli végpontja. Legyen

$$\textit{Éllefogás} = \{(G, k) |\ G\text{-ben van } k \text{ pontú éllefogó halmaz}\}.$$

Feladat: Mutassuk meg, hogy a $H \subseteq V(G)$ halmaz akkor, és csak akkor független halmaz G-ben, ha $V(G) \setminus H$ egy éllefogó halmaz.

Következmény: *Az Éllefogás nyelv* NP*-teljes.*

Bizonyítás: Nyilvánvaló, hogy *Éllefogás* $\in$ NP. Másfelől az előző feladatból azonnal következik, hogy a $(G, k) \mapsto (G, |V(G)| - k)$ megfeleltetés egy MAXFTLEN $\prec$ *Éllefogás* Karp-redukció. □

Megjegyezzük, hogy a MAXFTLEN, 3-SZÍN és *Éllefogás* nyelvek akkor is NP-teljesek maradnak, ha még azt is kikötjük, hogy G egy síkba rajzolható gráf.

Feladat: Mutassuk meg, hogy síkba rajzolható gráfokra a MAXKLIKK feladat megoldható polinom időben.

Feladat: Mutassuk meg, hogy páros gráfokra az *Éllefogás* feladat polinom időben megoldható. (König Dénes minimax tétele.)

8.7.4. A 3 dimenziós házasítás és az X3C feladat

A következő algoritmikus probléma a korábban megismert párosítási feladat általánosításának tekinthető. Legyenek U_1, U_2, U_3 azonos méretű véges halmazok, mondjuk legyen $|U_i| = t$. Tegyük fel, hogy adott még $U_1 \times U_2 \times U_3$ valamely S részhalmaza. Egy ilyen részhalmaz (u_1, u_2, u_3) alakú hármasokból áll, ahol $u_i \in U_i$. Szeretnénk eldönteni, hogy kiválasztható-e S-ből egy *3 dimenziós házasítás*. Ezen az S egy olyan t-elemű S' részhalmazát értjük, mely minden U_i-beli pontot lefed. Utóbbi követelmény pontosabban így fogalmazható: tetszőleges $v_i \in U_i$ elemhez van olyan $s \in S'$, melynek i-edik komponense v_i. Az $|S'| = t$

feltétel miatt egy 3 dimenziós házasítás az $U_1 \cup U_2 \cup U_3$ elemeit pontosan egyszeresen fedi le.

Jelölje 3-DH az olyan U_1, U_2, U_3; $S \subseteq U_1 \times U_2 \times U_3$ rendszereket, melyeknél S-ből kiválasztható egy 3 dimenziós házasítás.

Tétel: *A 3-DH feladat* NP*-teljes.*

Bizonyítás: Nyilvánvaló, hogy 3-DH $\in$ NP. Egy alkalmas S' halmaz játszhatja a tanú szerepét. A bizonyítás érdemi része egy 3-SAT $\prec$ 3-DH redukció lesz. Legyen $\phi \in$ 3-CNF. Tegyük fel, hogy ϕ-ben m változó $(x_1, \ldots, x_m)$ és z nyitózárójel van: $\phi = (\phi_1) \wedge \ldots \wedge (\phi_z)$.

Az U_1 halmazt úgy képezzük, hogy minden egyes literálnak (x_i vagy $\bar{x}_i$) megfeleltetünk z különböző elemet, így $t = 2mz$ és

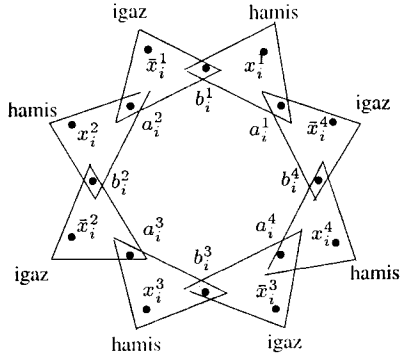
$$U_1 = \{x_i^j, \bar{x}_i{}^j \mid 1 \leq i \leq m, 1 \leq j \leq z\}.$$

Az x_i változó helyettesítéseinek a következő hármasokat feleltetjük meg:

$$E_i^{igaz} = \{(\bar{x}_i{}^j, a_i^j, b_i^j) \mid 1 \leq j \leq z\},$$

$$E_i^{hamis} = \{(x_i^j, a_i^{j+1}, b_i^j) \mid 1 \leq j \leq z\},$$

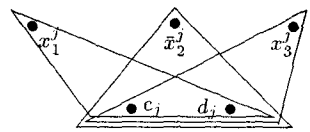
ahol a j szerinti indexelést ciklikusan végezzük, azaz $a_i^{z+1} = a_i^1$. Könnyen látható, hogy az $\{a_i^j \mid j = 1, \ldots, z\} \cup \{b_i^j \mid j = 1, \ldots, z\}$ halmaz $E_i^{igaz} \cup E_i^{hamis}$-beli hármasokkal csak úgy fedhető le egyrétűen egy rögzített i-re, hogy vagy az E_i^{igaz}, vagy az E_i^{hamis} rendszert vesszük S'-be. Előbbi esetben az U_1 halmazból a z darab $x_i = igaz$ értéknek megfelelő $x_i^1, \ldots, x_i^z$ elem, utóbbi esetben pedig a z darab $x_i = hamis$ értéknek megfelelő $\bar{x}_i{}^1, \ldots, \bar{x}_i{}^z$ elem marad lefedetlenül.



A ϕ_j részformulának feleltessük meg a következő hármasokat:

$$F_j = \{(x_i^j, c_j, d_j) \mid x_i \text{ literálja } \phi_j\text{-nek}\} \cup \{(\bar{x}_i^j, c_j, d_j) \mid \bar{x}_i \text{ literálja } \phi_j\text{-nek}\}.$$

Az F_j halmaznak legfeljebb 3 eleme van, mert a ϕ_j legfeljebb 3 literált tartalmaz. Például a $\phi_j = x_1 \vee \overline{x}_2 \vee x_3$ részformulának megfelelő hármasok így szemléltethetők:



Rögzített j-re a c_j, valamint a d_j elemek egyrétű lefedése az F_j halmaz egyetlen elemének a kijelölését jelenti. Az eddigiek alapján az alábbi H halmaz

$$\{a_i^j \mid i = 1, \ldots, m,\ j = 1, \ldots, z\} \cup \{c_j \mid j = 1, \ldots, z\} \cup$$
$$\cup \{b_i^j \mid i = 1, \ldots, m\ ; j = 1, \ldots, z\} \cup \{d_j \mid j = 1, \ldots, z\}$$

pontosan akkor fedhető le egyrétűen $\bigcup_{i=1}^{m}(E_i^{igaz} \cup E_i^{hamis}) \cup \bigcup_{j=1}^{z} F_j$-beli hármasokkal, ha a $\phi_1, \ldots, \phi_z$ formulák együttesen kielégíthetők. A c_j-t fedő hármas felel meg egy *igaz* értékű literálnak ϕ_j-ből, az a_i^j elemek pedig biztosítják, hogy ez a kijelölés megfelel a változók egy kiértékelésének (ha valahol az u literált választjuk igaznak, akkor a $\bar{u}$ literált sehol sem választhatjuk igaznak).

Már csak az van hátra, hogy kiegészítsük az eddigi konstrukciót a 3-DH egy inputjává. Az a_i^j, valamint a c_j elemeket az U_2 halmaz elemeinek tekintjük, a b_i^j, valamit a d_j elemeket pedig az U_3 halmaz elemeinek. Ki kell egészíteni az U_2 és U_3 halmazokat $t = 2mz$ eleművé. Az eddigi hármasok rendszerét pedig úgy kell bővíteni, hogy a H halmaz fenti szerkezetű egyrétű fedései, és csak azok legyenek kiegészíthetők az $U_1 \cup U_2 \cup U_3$ halmaz egyrétű lefedésévé. Ezek nyilvánvalóan teljesülnek, ha beveszünk U_2-be, valamint U_3-ba még egyenként $z(m-1)$ elemet: $e_1, \ldots, e_{z(m-1)}$-et, illetve $f_1, \ldots, f_{z(m-1)}$-et, továbbá a hármasok rendszerét kiegészítjük a

$$G = \left\{ (x_i^j, e_k, f_k), (\bar{x}_i^j, e_k, f_k) \;\middle|\; \begin{array}{l} i = 1, \ldots, m, \\ j = 1, \ldots, z, \\ k = 1, \ldots, z(m-1) \end{array} \right\}$$

halmazzal. Beláttuk tehát, hogy az eredeti $\phi \in$ 3-CNF pontosan akkor kielégíthető, ha az

$$\begin{aligned} U_1 &= \{x_i^j, \bar{x}_i{}^j \mid i = 1, \ldots, m,\ j = 1, \ldots, z\}, \\ U_2 &= \{a_i^j, c_j, e_k \mid i = 1, \ldots, m,\ j = 1, \ldots, z,\ k = 1, \ldots, z(m-1)\}, \\ U_3 &= \{b_i^j, d_j, f_k \mid i = 1, \ldots, m,\ j = 1, \ldots, z,\ k = 1, \ldots, z(m-1)\}, \\ S &= G \cup \bigcup_{i=1}^{m} (E_i^{igaz} \cup E_i^{hamis}) \cup \bigcup_{j=1}^{z} F_j \end{aligned}$$

inputra a 3-DH feladat megoldható. A konstrukció polinom időben elvégezhető; a tételt igazoltuk.□

Egy szorosan kapcsolódó érdekes probléma a *Pontos fedés hármasokkal*: adott egy U véges halmaz, és U háromelemű részhalmazainak egy $\mathcal{F} = \{X_1, X_2, \ldots, X_k\}$ családja. Eldöntendő, hogy az $\mathcal{F}$-ből kiválaszthatók-e páronként diszjunkt halmazok, melyek együttesen lefedik U-t. Jelölje X3C azokat az $(U, \mathcal{F})$ párokat, melyekre az $\mathcal{F}$-ből kiválasztható U ilyen értelemben vett pontos fedése.

Példa: Legyen $U = \{1, 2, 3, \ldots, 9\}$,

$$\begin{aligned} \mathcal{F} &= \{\{1,2,3\}, \{1,4,5\}, \{1,6,7\}, \{1,8,9\}\} \text{ és} \\ \mathcal{F}' &= \{\{1,2,3\}, \{4,5,6\}, \{1,6,7\}, \{7,8,9\}\}. \end{aligned}$$

Ekkor $\mathcal{F}$-ből nem választható ki pontos fedés, hiszen az 1 benne van mindegyik hármasban. Tehát $(U, \mathcal{F}) \notin$ X3C. A $\mathcal{F}'$ viszont tartalmazza U egy pontos fedését, ezért $(U, \mathcal{F}') \in$ X3C.

Állítás: *Az X3C nyelv* NP*-teljes.*

Bizonyítás: X3C $\in$ NP teljesül; tanú lehet egy pontos fedés. Megmutatjuk ezután, hogy 3-DH $\prec$ X3C. Legyen $U_1, U_2, U_3; S$ a 3-DH egy inputja. Az elemek esetleges átnevezése után feltehetjük, hogy az U_i halmazok páronként diszjunktak. Legyen $U = U_1 \cup U_2 \cup U_3$, és $\mathcal{F} = \{\{a,b,c\} \mid (a,b,c) \in S\}$. Könnyű meggondolni, hogy az így kapott megfeleltetés Karp-redukció. Ennek a részleteit az olvasóra bízzuk. □

8.7.5. Hamilton-kört tartalmazó gráfok és az Utazó ügynök probléma

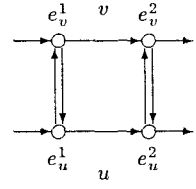
Korábban már foglalkoztunk az irányítatlan, illetve irányított Hamilton-kört tartalmazó gráfokkal, megállapítva, hogy a H és IH nyelvek NP-beliek. Most megmutatjuk, hogy ezek is nehéz nyelvek.

Tétel: *Az IH nyelv* NP*-teljes.*

Bizonyítás: Megmutatjuk, hogy *Éllefogás* $\prec$ IH. Legyen $G = (V, E)$ egy irányítatlan gráf, k pedig egy pozitív egész. Feltehetjük, hogy $k \leq |V(G)|$. A (G, k) pár segítségével hatékonyan (polinom időben) elkészítünk egy $G' = (V', E')$ irányított gráfot, melyben pontosan akkor lesz irányított Hamilton-kör, ha G élei lefoghatók k csúccsal. A G minden $e = (u, v) \in E$ éléhez elkészítjük a

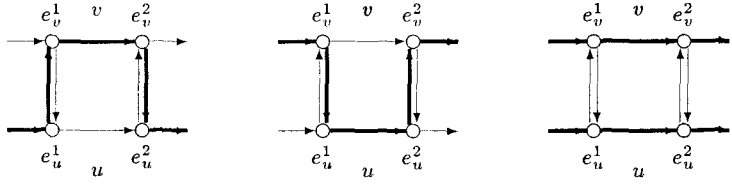
$$G'_e = (V'_e, E'_e) = \{e_u^1, e_u^2, e_v^1, e_v^2\},$$
$$\{e_u^1 \to e_u^2,\ e_v^1 \to e_v^2,\ e_u^1 \to e_v^1,\ e_v^1 \to e_u^1,\ e_u^2 \to e_v^2,\ e_v^2 \to e_u^2)\}\}$$

irányított gráfot.



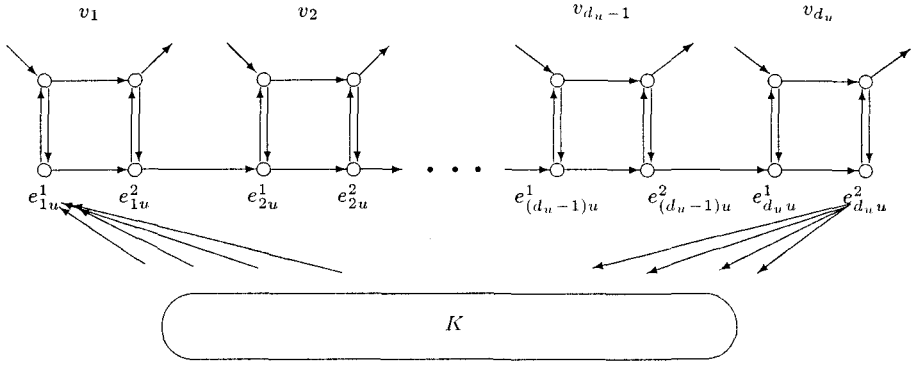
A G' gráfot jórészt ilyen G'_e alakú kis részgráfokból fogjuk felépíteni. Egy G'_e gráfba kívülről csak az e_u^1, illetve az e_v^1 csúcsokba futnak élek, kifelé pedig csak az e_u^2, illetve az e_v^2 csúcsokból mennek élek. Ezekkel a megkötésekkel könnyű belátni, hogy a felépítendő G' gráf egy Hamilton-köre csak a következő háromféle módon haladhat át a $G'_{u,v}$ részgráfon:

(1) $\cdots \to e_u^1 \to e_v^1 \to e_v^2 \to e_u^2 \to \cdots$
(2) $\cdots \to e_v^1 \to e_u^1 \to e_u^2 \to e_v^2 \to \cdots$
(3) $\cdots \to e_u^1 \to e_u^2 \to \cdots \to e_v^1 \to e_v^2 \to \cdots$



Ezek után a G' gráf felépítése a következő: Vegyük G minden egyes e éléhez a G'_e gráfot. A G minden $u \in V$ csúcsának megfelelően fűzzük fel egy irányított útra tetszőleges sorrendben (pl. az éllista szerinti sorrendben) az u-hoz csatlakozó

éleknek megfelelő G'_e részgráfoknak az u csúcshoz tartozó felét ($\{e^1_u, e^2_u\}$). Vegyünk még egy k elemű K ponthalmazt, és K összes eleméből vezessünk élet az előbb megadott utak kezdőpontjához, továbbá vezessünk K minden pontjába élet az utak végpontjaiból. Az $u \in V$ csúcshoz tartozó irányított út így szemléltethető (az u foka d_u):



A formális leírások kedvelői számára legyen E'_u a következő élhalmaz:

$$E'_u = \{w \to e^1_{1u},\ e^2_{1u} \to e^1_{2u},\ \ldots,\ e^2_{(d_u-1)u} \to e^1_{d_u u},\ e^2_{d_u u} \to w \mid w \in K\},$$

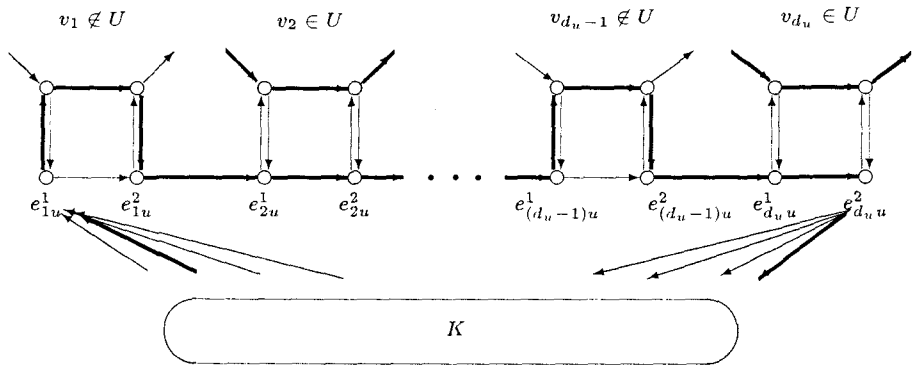
ahol $e_1, \ldots, e_{d_u}$ az u-hoz csatlakozó G-beli élek listája; ezután G' így adható meg:

$$V' = K \cup \bigcup_{e \in E} V'_e, \quad E' = \bigcup_{e \in E} E'_e \cup \bigcup_{u \in V} E'_u$$

Tegyük fel most, hogy G' tartalmaz egy H irányított Hamilton-kört. A kis gráfokkal kapcsolatos (1)-(3) bejárási lehetőségeket figyelembe véve állíthatjuk, hogy ha valamely $u \in V$ csúcs esetén H tartalmaz legalább egy élet az E'_u halmazból, akkor ezen él a H egy ilyen szerkezetű darabján van:

$$\cdots \to w_j \to e^1_{1u} \rightsquigarrow e^2_{1u} \to e^1_{2u} \rightsquigarrow e^2_{2u} \to \cdots \to e^1_{d_u u} \rightsquigarrow e^2_{d_u u} \to w_{j+1} \to \cdots.$$

Itt $w_j, w_{j+1} \in K$, és egy hullámos nyíl vagy egy (1) típusú 3 hosszú utat jelöl a G'_{e_i} részgráfon belül vagy pedig az $e^1_{iu} \to e^2_{iu}$ élet (utóbbi esetben szükségképpen a H egy másik darabja tér vissza G'_{e_i} maradék két csúcsát meglátogatni).



A K halmaznak k eleme van, ezért a H Hamilton-kör k darab ilyen útból tevődik össze:

$$\begin{aligned} w_1 \to u_1\text{-nek megfelelő rész} &\to w_2 \to u_2 \to \cdots \\ \cdots &\to w_k \to u_k\text{-nak megfelelő rész} \to w_1. \end{aligned}$$

A H mindegyik kis G'_e ($e \in E$) csúcsait bejárja. Ez azt jelenti, hogy az $u_1, \ldots, u_k$ csúcsok együttesen lefogják G éleit. Van tehát G-nek k elemű éllefogó ponthalmaza.

Fordítva, tegyük fel, hogy $U = \{u_1, \ldots, u_k\}$ egy éllefogó pontrendszer G-ben. Ekkor a következő utak összefűzve egy Hamilton-kört adnak a G' gráfban:

$$\begin{aligned} \cdots \to w_j \to e^1_{1u_j} \rightsquigarrow e^2_{1u_j} \to e^1_{2u_j} &\rightsquigarrow e^2_{2u_j} \to \cdots \\ \cdots \to e^1_{d_{u_j}u_j} &\rightsquigarrow e^2_{d_{u_j}u_j} \to w_{j+1} \to \cdots, \end{aligned}$$

ahol $w_{k+1} = w_1$; a hullámos nyilak G'_e-beli utakat jelölnek, mégpedig ha e-nek a másik végpontja nincs U-ban, akkor a 3 hosszú utat, ha pedig benne van, akkor az 1 hosszút. Utóbbi esetben az e másik végpontjának megfelelő részben található meg az út párja.

Beláttuk tehát, hogy G-ben pontosan akkor van k elemű éllefogó pontrendszer, ha G'-ben van irányított Hamilton-kör. A G' gráf konstrukciója hatékonyan elvégezhető. Ezzel a bizonyítást befejeztük. □

A DAG-ok algoritmusainak tárgyalásakor (6.4.2. rész) foglalkoztunk a két pont közötti leghosszabb egyszerű utak meghatározásának a feladatával. Kiderült, hogy ez villámgyorsan, vagyis lineáris uniform költséggel megoldható, amikor a bemeneti G gráf egy DAG. Az előző tétel masszív érvet szolgáltat amellett, hogy a feladat nehézzé válik, ha a G tetszőleges irányított gráfot jelenthet. Ha volna

ugyanis egy gyors (mondjuk polinom idejű) módszerünk, amely megmondaná a G gráf u, v pontjaira a leghosszabb egyszerű $u \rightsquigarrow v$ irányított utak $l(u, v)$ hosszát, akkor az IH nyelvet is gyorsan el tudnánk dönteni.

Feladat: Igazoljuk az előző mondatban foglalt állítást. (Legyen a $G = (V, E)$ irányított gráf az IH egy bemenete. Rendeljünk egységnyi súlyokat az éleihez. Mutassuk meg, hogy a G egy $u \to v$ éle pontosan akkor éle egy irányított Hamilton-körnek, ha a G-ből az $u \to v$ törlésével kapott gráfban $l(v, u) = |V| - 1$.)

Az IH nyelv NP-teljességét tudva az irányítatlan Hamilton-köröket tartalmazó gráfokról szóló hasonló állítással elég egyszerűen elboldogulunk.

Tétel: *A H nyelv* NP*-teljes.*

Bizonyítás: Korábban már láttuk, hogy H $\in$ NP, és az első Karp-redukció, amit megnéztünk, egy IH $\prec$ H leképezés volt. □

Az irányítatlan Hamilton-kör probléma nevezetes általánosítása az *Utazó ügynök* feladat: adott egy G irányítatlan gráf pozitív egészekkel súlyozott élekkel. A cél minél rövidebb összsúlyú Hamilton-kört találni G-ben. (Az ügynök szeretné minél kisebb költséggel sorba látogatni az adott városokat, mindegyiket csak egyszer, napszálltával hazatérve.) A feladathoz kézenfekvően kapcsolódó Ut nyelv olyan (G, k) párokból áll, melyekben G egy súlyozott élű irányítatlan gráf, k egy nemnegatív egész, és G-ben van k-nál nem nagyobb súlyú Hamilton-kör. Világos, hogy $Ut \in$ NP. A H feladat az Ut feladat egy speciális esetének tekinthető: minden élsúly 1 és $k = |V(G)|$. Az a leképezés tehát, amely a G gráfhoz a csupa egyessel súlyozott élű G-t és a $k = |V(G)|$ korlátot rendeli, egy H $\prec Ut$ Karp-redukciót ad meg. Érvényes a következő:

Tétel: *Az* Ut *nyelv* NP*-teljes.* □

8.7.6. A Hátizsák feladat és néhány más rokon probléma

Itt olyan kérdésekről lesz szó, melyekben az eddigiekhez képest több szerep jut a számoknak. Az első a *Hátizsák* feladat: adottak az $s_1, \ldots, s_m > 0$ súlyok, ezek $v_1, \ldots, v_m > 0$ értékei, valamint a b megengedett maximális összsúly. Tegyük fel, hogy az s_i, v_i, b számok egészek. A feladat az, hogy találjunk egy olyan $I \subseteq \{1, .., m\}$ részhalmazt, melyre $\sum_{i \in I} s_i \leq b$, és ugyanakkor $\sum_{i \in I} v_i$ a lehető legnagyobb. Szemléletes képként gondoljunk arra, hogy van egy b teherbírású hátizsákunk. Ebbe akarunk belerakni bizonyosakat az s_i súlyú tárgyak közül úgy, hogy a hátizsákban levő dolgok összértéke a lehető legnagyobb legyen. Másfelől

a hátizsákot kímélni kell: a bepakolt tárgyak összsúlya nem lehet több b-nél. Az I indexhalmaz mondja meg, hogy a súlyok közül melyeket raktuk a hátizsákba.

Polinom idejű extra munka erejéig azonos nehézségű NP-beli feladatot kapunk, ha bevezetünk még egy $k > 0$ értékkorlátot és egy adott $s_1, \ldots, s_m$; $v_1 \ldots, v_m; b; k$ inputra azt kérdezzük, hogy van-e olyan kímélő kitöltése a hátizsáknak, melynek az értéke legalább k. Az így kapott döntési feladat (nyelv) neve legyen *Hát*.

$$\textit{Hát} = \{(s_1, s_2, \ldots, s_m; v_1, v_2, \ldots, v_m; b; k) | s_i, b, k > 0 \text{ egészek, és}$$
$$\text{van olyan } I \subseteq \{1, \ldots, m\} \text{ melyre } \sum_{i \in I} s_i \leq b \text{ és } \sum_{i \in I} v_i \geq k\}.$$

Feladat: Mutassuk meg, hogy *Hát* $\in$ P-ből *Hátizsák* $\in$ FP következne. (A prímtényezős felbontás és az F feladat viszonyát tisztázó állítás ötlete használható: bináris kereséssel meghatározhatjuk a legnagyobb elérhető k-t.)

Igazából úgy gondoljuk, hogy a valóságban nem teljesül a feladat feltétele, a *Hát* nyelv nem ismerhető fel polinom időben. Ezt alátámasztandó megmutatjuk, hogy a *Hát* nyelv NP-teljes. Egészen pontosan azt a speciális esetet fogjuk szemügyre venni, amikor a tárgyak súlya és értéke azonos, valamint a súlykorlát megegyezik az értékkorláttal: $s_i = v_i$ $(i = 1, 2, \ldots, m)$ és $b = k$. Ez *Részhalmazösszeg probléma*. Itt arról az egyszerűbb algoritmikus kérdésről van szó, hogy pontosan tele lehet-e rakni a hátizsákot az adott tárgyakból válogatva. A megfelelő nyelv

$$\text{RH} = \left\{ (s_1, \ldots, s_m; b) \;\middle|\; \begin{array}{l} s_i, b > 0 \text{ egészek,} \\ \text{és van olyan } I \subseteq \{1, \ldots, m\} \text{ hogy } \sum_{i \in I} s_i = b \end{array} \right\}.$$

Tétel: *Az RH nyelv* NP*-teljes.*

Bizonyítás: Világos, hogy RH $\in$ NP; alkalmas I indexhalmaz lehet a tanú. Ezután megadunk egy X3C $\prec$ RH Karp-redukciót. Vegyük X3C tetszőleges inputját: ez egy U véges halmaz és egy $\mathcal{F} = \{X_1, \ldots, X_m\}$ halmazrendszer, melyre $X_i \subseteq U$ és $|X_i| = 3$. Ebből az RH feladat egy inputját fogjuk hatékonyan elkészíteni. Tegyük fel, hogy $U = \{1, 2, \ldots, t\}$, és legyenek

$$\begin{aligned} r &= 1 + \binom{t}{2}, \\ z_j &= r^{j-1} \quad (j = 1, \ldots, t), \\ s_i &= \sum_{j \in X_i} z_j \quad (i = 1, \ldots, m), \\ b &= \sum_{j=1}^{t} z_j. \end{aligned}$$

Lényegében az történik, hogy az X_i halmazokat elég nagy (itt konkrétan r) alapú számrendszerben felírt számokkal kódoljuk. Az $x = (s_1, \ldots, s_m; b)$ együttest fogjuk az RH inputjaként értelmezni. Az x hossza $O(mt \log t)$, mert $\log s_i = O(\log z_t) = O(t \log t)$, ezért x a megadott formulák szerint (t-ben) polinom időben előállítható. A megfeleltetés tehát teljesíti a Karp-redukcióval szemben támasztott hatékonysági követelményt.

Nézzük ezután a másik előírást. Legyen $I \subseteq \{1, \ldots, m\}$ egy indexhalmaz. Az $f_j = |\{i | i \in I \text{ és } j \in X_i\}|$ számokat az I-hez tartozó *fedési számoknak* nevezzük ($j = 1, \ldots, t$). Az f_j fedési szám azt mondja meg, hogy az $i \in I$ indexekhez tartozó X_i halmazok közül hány tartalmazza a $j \in U$ elemet. Ezután megmutatjuk, hogy az I-hez tartozó $\sum_{i \in I} s_i$ részhalmazösszeg és az I fedési számainak $f_1, f_2, \ldots, f_t$ sorozata kölcsönösen egyértelműen meghatározzák egymást. A részhalmazösszeg így írható:

$$\sum_{i \in I} s_i = \sum_{i \in I} \sum_{j \in X_i} z_j = \sum_{j=1}^{t} |\{i | i \in I \text{ és } j \in X_i\}| r^{j-1} = \sum_{j=1}^{t} f_j r^{j-1}.$$

A $j \in U$ pontot legfeljebb $\binom{t}{2}$ számú hármas tartalmazza, ezért a jobb oldalon $\binom{t}{2} < r$ miatt r^{j-1} együtthatója kisebb, mint r. Eszerint az f_j fedési számok éppen az I-nek megfelelő részhalmazösszeg r alapú számrendszerbeli jegyei. Az r alapú számrendszerbeli felírás egyértelműsége miatt a b szám tehát akkor és csak akkor lesz az I-nek megfelelő részhalmazösszeg, ha az összes fedési szám egy, ami éppen annyit tesz, hogy az $\{X_i | i \in I\}$ halmazrendszer pontosan fedi U-t. Tehát b akkor és csak akkor részhalmazösszeg, ha létezik pontos fedés. □

A Részhalmazösszeg probléma egy érdekes speciális esete a *Partíció* feladat, ahol a b megcélzott összeg az s_i számok összegének a fele:

$$\textit{Partíció} = \left\{ (s_1, \ldots, s_m) \;\middle|\; \begin{array}{l} s_i > 0 \text{ egészek, és van olyan} \\ I \subseteq \{1, \ldots, m\}, \text{ hogy } \sum_{i \in I} s_i = \frac{1}{2} \sum_{i=1}^{m} s_i \end{array} \right\}.$$

Arról a kérdésről van szó, hogy az $\{1, 2, \ldots, m\}$ halmaz felbontható-e két egyenlő részhalmazösszeget adó részre. Nyilvánvaló, hogy *Partíció* $\in$ NP.

Feladat: Mutassuk meg, hogy a *Partíció* nyelv NP-teljes, megadva egy RH $\prec$*Partíció* redukciót. (Vegyük az RH egy $x = (s_1, \ldots, s_m; b)$ inputját. Feltehető, hogy $b \leq s = \sum_{i=1}^{m} s_i$. Az x-hez rendeljük a *Partíció* következő inputját: $(s_1, \ldots, s_m, s + 1 - b, b + 1)$. Itt a számok összege $2s + 2$. Az utolsó két szám nem lehet egy partíció ugyanazon osztályában, mert az összegük túl nagy: $s + 2$.)

A *Ládapakolás* feladat a következő: adottak az $s_1, \ldots, s_m$ súlyú tárgyak, $0 < s_i \leq 1$, s_i racionális, és egy $k > 0$ egész. Eldöntendő, hogy a tárgyakat bele lehet-e pakolni legfeljebb k számú egységnyi súlykapacitású ládába.

Feladat: Mutassuk meg, hogy *Ládapakolás* feladat nyelve NP-teljes. (Adjunk meg egy *Partíció*$\prec$*Ládapakolás* Karp-redukciót; elegendő két ládára szorítkozni, azaz a $k = 2$ eset is megteszi.)

A *Ládapakolás* problémával a következő fejezetben is találkozunk. A feladat (közelítő) megoldására szolgáló algoritmusokat fogunk ismertetni.

8.7.7. Lineáris programozás

Itt egy igen széles körben előforduló feladatról lesz szó. Alkalmazásaival találkozhatunk egyebek között a gazdasági számítások és a mérnöki tervezés területén. Egyszerűen fogalmazva, változók lineáris függvényét kell optimalizálni (maximalizálni, vagy minimalizálni) egy olyan tartományban, amelyet a változókra kirótt lineáris egyenlőtlenségek írnak le. Kezdjük egy nevezetes, gazdasági tónusú példával:

Példa: *Termékszerkezet-feladat*
Egy üzem n-féle terméket tud termelni; ennek során m-féle erőforrást használ fel. Az erőforrások jelenthetnek pl. munkaórákat, anyagokat, gépidőket, stb. Legyen c_j a j-edik termék egységnyi mennyiségére eső nyereség, b_i az i-edik erőforrásból rendelkezésre álló mennyiség, továbbá a_{ij} az i-edik erőforrásból az a mennyiség, ami a j-edik termék egy egységének előállításához szükséges. Az üzem vezetésének a célja egy olyan termékszerkezet kialakítása, ami az előbb leírt körülmények között maximalizálja a nyereséget.

Jelölje x_j a j-edik termékből előállítani kívánt mennyiséget. A feladat így fogalmazható:

$$
\begin{array}{lll}
(*) & \text{maximalizáljuk a} & \displaystyle\sum_{j=1}^{n} c_j x_j \text{mennyiséget,} \\
& \text{feltéve, hogy} & \displaystyle\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, 2, \ldots, m, \\
& \text{továbbá} & x_j \geq 0, \quad j = 1, 2, \ldots, n.
\end{array}
$$

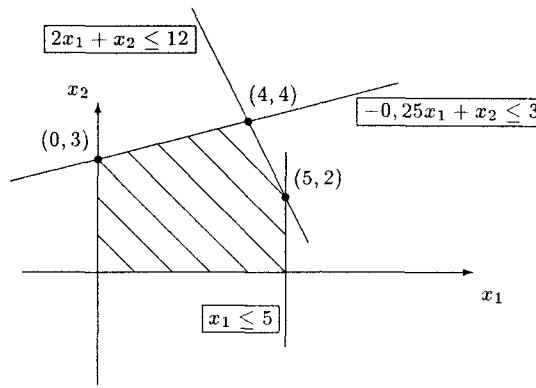
A $(*)$ alakban felírható feladatokat – ahol a c_j, b_i és a_{ij} tetszőleges racionális számok –, *lineáris programozási feladat*oknak nevezzük. A feladat *megoldásán* egy olyan racionális számokból álló $(x_1, x_2, \ldots, x_n)$ vektort értünk, melyre a $\sum_{j=1}^{n} c_j x_j$ mennyiség (az ún. *célfüggvény* értéke) maximális az előírt egyenlőtlenségeket teljesítő vektorok között.

Egy lineáris programozási feladatot tehát az n, m paraméterekkel, a célfüggvényt jelentő $(c_1, \ldots, c_n)$ vektorral, valamint a feltételeket leíró $A = [a_{ij}]$ $(1 \leq i \leq m, 1 \leq j \leq n)$ mátrixszal és $(b_1, \ldots, b_m)$ vektorral adhatunk meg.

Példa: Legyen $n = 2$, $m = 3$, $c_1 = c_2 = 1$,

$$A = \begin{pmatrix} 1 & 0 \\ 2 & 1 \\ -0{,}25 & 1 \end{pmatrix},$$

továbbá $b_1 = 5$, $b_2 = 12$ és $b_3 = 3$. Az $x_1 + x_2$ mennyiség maximumát keressük az $x_1 \leq 5$, $2x_1 + x_2 \leq 12$, $-0,25x_1 + x_2 \leq 3$, $x_1 \geq 0$ és $x_2 \geq 0$ egyenlőtlenségek által meghatározott tartományon. Ez egy sokszöglemez az (x_1, x_2)-síkon:



A második egyenlőtlenség ötszörösével adjuk hozzá a harmadik négyszeresét: azt kapjuk, hogy $9x_1 + 9x_2 \leq 72$. Innen már látszik, hogy a célfüggvény optimuma legfeljebb 8; ezt a $(4, 4)$-ben el is éri. A $(4, 4)$ pont az egyik csúcsa a feltételeket teljesítő pontokból álló alakzatnak.

Sokáig nyitott kérdés volt, hogy a lineáris programozási feladat mint algoritmikus probléma milyen nehéz. A megoldására kidolgozott eljárások közül a legnépszerűbb és leghasznosabb az ún. *szimplex-módszer*. Ennek ismertetése itt nem célunk. A módszert George Dantzig javasolta a negyvenes évek végén, és mostanra szinte művészi tökéllyel megírt, rendkívül csiszolt implementációi születtek[2]. Úgy tűnik, a gyakorlatilag fontos feladatok széles körében igen gyors: $O(m^2 n)$ aritmetikai művelet elegendő a „praktikus" feladatok megoldására. Ezzel szemben ismertek olyan példák, amelyeken a módszer exponenciális nagyságrendű műveletet végez. A szimplex-módszer a feltételek által adott alakzat csúcsaiban keresi

[2] Az egyik legnépszerűbb ezek közül a MINOS nevű rendszer.

az optimumot[3]. Megadhatók olyan $2d$ feltételt és d változót tartalmazó ($m = 2d$, $n = d$) lineáris programozási feladatok, amelyeknél a szimplex-módszer egyik legjobb megfigyelt viselkedésű változata $2^d - 1$ csúcsot vizsgál meg.

Nagy visszhangot kiváltó áttörést jelentett L. G. Hacsijan 1979-ben javasolt eljárása, az *ellipszoid-módszer*. Az ellipszoid-módszer az első polinom idejű algoritmus a lineáris programozási feladatok megoldására. Hacsijan eredménye úgy is fogalmazható, hogy a probléma az FP osztályban van. Az algoritmus a gyakorlati feladatok körében eddig nem tudott versenyezni a szimplex-módszerrel. Jelentősége ezért elsősorban elméleti. Ilyen értelemben viszont erős eszköznek számít. Több érdekes kombinatorikus optimalizálási feladatra az ellipszoid-módszer segítségével sikerült polinom idejű algoritmust nyerni.

1984-ben Narendra K. Karmarkar javasolt egy az addigiaktól merőben eltérő szemléletű algoritmust, a *belső pont módszert*. Ez – csakúgy, mint Hacsijan módszere – szintén polinomkorlátos algoritmus. A Karmarkar gondolatai nyomán született eljárások a gyakorlatban is versenyezni tudnak a szimplex-módszerrel. Több esettanulmány mutat arra, hogy nagy méretű (gyakorlati) feladatok megoldására a belső pont módszerek hatékonyabbak a szimplex-módszernél. Egyes szerzők szerint $m + n > 2500$ esetén az új technikák már sokkal gyorsabbak.

A lineáris programozás feladata tehát polinom időben megoldható. Hasonló mondható a megfelelő igen-nem feladatról. Ez annyiban különbözik $(*)$-tól, hogy a c_j, b_i és a_{ij} adatok mellett még egy racionális szám, a K korlát is része a bemenetnek. Az eldöntendő kérdés pedig az, hogy van-e olyan racionális számokból álló $(x_1, x_2, \ldots, x_n)$ vektor, melyre a $(*)$-beli egyenlőtlenségek teljesülnek, és $\sum_{j=1}^{n} c_j x_j \geq K$.

Egész értékű lineáris programozás

A lineáris programozási probléma „diszkrét" változata az *egész értékű lineáris programozási feladat*. Ezen olyan, a $(*)$ alakban felírható feladatot értünk, amelyben a c_j, b_i és a_{ij} adatok egészek, és az optimumot az egész koordinátájú $(x_1, x_2, \ldots, x_n) \in \mathbb{Z}^n$ vektorok körében keressük. A megfelelő nyelv (igen-nem probléma) a racionális esethez hasonlóan kapható meg. Álljon az EP nyelv azokból a $(K;\ c_1, \ldots, c_n;\ b_1, \ldots, b_m;\ a_{11}, a_{12}, \ldots, a_{mn})$ bemenetekből

[3]Az előző síkbeli példához hasonlóan magasabb dimenzióban ($n > 2$) is beszélhetünk az egyenlőtlenségek által kijelölt tartomány csúcsairól. Ezek az olyan $(x_1, \ldots, x_n)$ pontok közül kerülnek ki, amelyekre az $n + m$ egyenlőtlenség közül n-ben egyenlőség igaz. Megmutatható, hogy ha a feladatnak van egyáltalán megoldása (véges optimuma), akkor a csúcspontok egyike is megoldás lesz.

$(K, c_j, b_i, a_{ij} \in \mathbb{Z})$, amelyekhez van olyan $(x_1, x_2, \ldots, x_n) \in \mathbb{Z}^n$ vektor, hogy

$$\sum_{j=1}^{n} c_j x_j \geq K, \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, 2, \ldots, m, \text{ és } x_j \geq 0, \quad j = 1, 2, \ldots, n.$$

Megmutatható – ennek a részleteire nem térünk ki –, hogy az EP nyelv az NP osztályban van. Egy megfelelő $(x_1, x_2, \ldots, x_n) \in (\mathbb{Z}^+)^n$ vektor szolgálhat tanúként. Az is igaz, hogy az egész értékű optimalizálási feladat és az EP nyelv felismerése polinomiálisan ekvivalens nehézségűek. Tehát ha létezne polinom idejű algoritmus EP felismerésére, akkor az optimalizálási feladat FP-ben lenne. A következő állítást úgy is értelmezhetjük, hogy ez nem túlságosan valószínű.

Tétel: *Az EP nyelv* NP-*teljes.*

Bizonyítás: Megmutatjuk, hogy RH $\prec$ EP, ahol RH a Részhalmazösszeg feladat nyelve. Az RH egy $(s_1, \ldots, s_n; b)$ bemenetéhez rendeljük a következő egyenlőtlenségeket: $x_j \leq 1$, $j = 1, 2, \ldots, n$, és $\sum_{j=1}^{n} s_j x_j \leq b$. A célfüggvény legyen $\sum_{j=1}^{n} s_j x_j$, a korlát pedig $K = b$. Ezzel az EP egy bemenetét kaptuk; n változónk és $n + 1$ feltételünk van.

Az így szerkesztett bemenet pontosan akkor van EP-ben, ha létezik olyan $(x_1, x_2, \ldots, x_n) \in (\mathbb{Z}^+)^n$ vektor, amelyre teljesülnek a feltételi egyenlőtlenségek, és $\sum_{j=1}^{n} s_j x_j \geq b$. Az utolsó feltétel miatt ekkor $\sum_{j=1}^{n} s_j x_j = b$. Az x_j számok nemnegatív egészek, így az értékük csak 0 vagy 1 lehet. Legyen $I = \{j;\ x_j = 1\}$. Az $\{s_j;\ j \in I\}$ részhalmaz összege éppen b.

Fordítva, ha az $\{s_j;\ j \in I\}$ részhalmaz összege b, akkor az

$$x_j = 0, \text{ ha } j \notin I \text{ és } x_j = 1, \text{ ha } j \in I$$

feltételekkel megadott vektor teljesíti az egyenlőtlenségeket. A megfeleltetés tényleg Karp-redukció. □

Az egész értékű lineáris programozás két szempontból fontos. Egyfelől rendkívül sok érdekes optimalizálási feladat fogalmazható meg ebben a keretben. Az alkalmazási területeknek se szeri, se száma (pl. termékterítés, gyártásszervezés, VLSI-tervezés, kommunikációs és szállítási hálózatok tervezése). Ennek az erős modellezési képességnek a szemléltetésére ajánljuk az olvasónak, hogy fogalmazzon meg néhány további NP-teljes feladatot egész programozási kérdésként:

Feladat: Adjunk meg (minél egyszerűbb) $L \prec$ EP Karp-redukciókat, ahol L a SAT, a 3-DH, a 3-SZÍN, vagy a *Ládapakolás* nyelvet jelenti.

A másik tényező, ami miatt az egész értékű programozás megkülönböztetett figyelmet érdemel, hogy ezzel a feladattípussal sokat foglalkoztak és foglalkoznak a kutatók. Megoldási módszerek gazdag választéka áll rendelkezésre. Ezek a módszerek ugyan nem polinom idejűek – ilyenek nem is remélhetők, hiszen nehéz feladatról van szó –, de számos esetben használhatónak bizonyultak.

8.7.8. Minimális késésszámú ütemezés

Végezetül megemlítünk egyet a sokféle NP-teljes ütemezési feladat közül. A *Minimális késésszámú ütemezés* feladatának bemenete a következő: elvégzendő munkák (véges) D halmaza; minden feladat egy időegységet igényel, és egyszerre csak egy munkával tudunk foglalkozni. Adott még ezen kívül egy $<$ részbenrendezés (irreflexív, tranzitív reláció) D-n. A $d_1 < d_2$ viszony azt jelenti, hogy a d_1 munkát előbb kell elvégezni, mint d_2-t. Minden egyes $d \in D$ munkához adott a pozitív egész $h(d)$ határidő. Végül adott egy $T > 0$ tűréshatár. A T tűréshatár azt jelenti, hogy legfeljebb T olyan feladat lehet, amivel nem készülünk el határidőre.

Az eldöntendő kérdés az, hogy lehet-e a feladatokat úgy ütemezni, hogy legfeljebb T kivétellel minden feladatot határidőre befejezzünk. Pontosabban fogalmazva: van-e olyan $d_1, d_2, \ldots d_k$ sorrendje a D elemeinek, melyre
1. $d_i < d_j$ esetén $i < j$ teljesül; továbbá
2. legfeljebb T számú i index kivételével fennáll a $h(d_i) \leq i$ egyenlőtlenség?

Az 1. feltétel a $<$ részbenrendezés által meghatározott gráf egy topologikus rendezését követeli meg. Ez önmagában könnyen teljesíthető, hiszen a gráf DAG. A 2. feltétel jelenlétével viszont nehéz feladatot kapunk. Megmutatható, hogy az adott tűréshatárral megoldható ütemezési feladatok nyelve NP-teljes. A MAXKLIKK nyelv viszonylag egyszerűen redukálható erre a nyelvre.