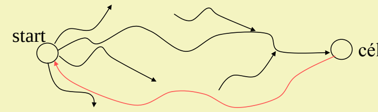
IV. Probléma dekompozíció

1. Visszafelé haladó keresés
2. Probléma redukció
3. Probléma dekompozíció
4. ÉS/VAGY gráfok

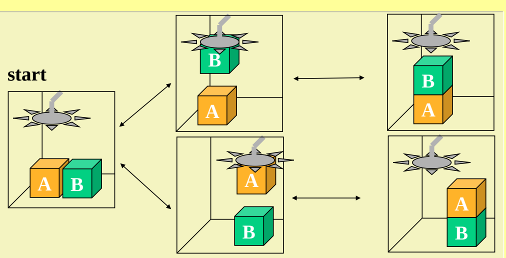
1. Visszafelé haladó keresés

- Ha a problématér a cél felől nézve egyszerűbb (kevesebb alternatívát mutat), mint a start felől nézve, akkor érdemes visszafelé haladó keresést alkalmazni.



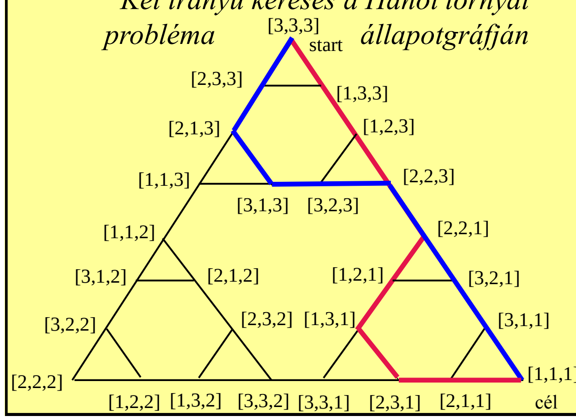
- A megtalált megoldási utat azonban a starttól a cél felé haladva kell értelmezni. (De vajon lehet-e, van-e minden élnek inverze?)

Kockavilág probléma

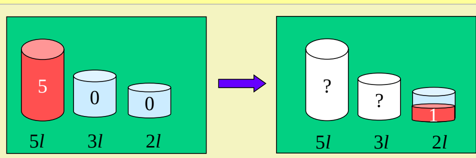


- A megoldási út megtalálása a célból a start felé haladva egyszerűbb.

Két irányú keresés a Hanoi tornyai probléma állapotgráfján



Miért nem oldja meg a kancsó-problémát egy visszafelé haladó keresés?



- Kiindulási célállapot kiválasztása nem egyszerű: Nem elérhető célállapot például a [*2,2,1*].
- A visszafelé haladó kereséssel talált [*4,0,1*]→[*5,0,0*] út nem értelmezhető a feladat megoldásaként.

Visszafelé haladó keresés feltételei

- A műveletek invertálhatóak legyenek (legalábbis a visszafelé haladó keresés által alkalmazottak).
- Nem kizáró feltétel, de nehezíti a visszafelé haladó keresést, ha nem ismert a konkrét célállapot.

Mit tegyünk, ha a fenti két feltétel nem áll fenn, de visszafelé haladó keresést akarunk megvalósítani?

2. Probléma redukció

- A cél az, hogy meghatározzuk egy (esetleg csak részben ismert) állapot közvetlen megelőző állapotait.
- Ahelyett, hogy azt vizsgálnánk, hogy
  - mely állapotokba vezet művelet a vizsgált állapotból,
- arra a kérdésre keressük a választ, hogy
  - mely állapotokból vezet művelet a vizsgált állapotba.

Redukció

$s^? \to t$

Van-e olyan t_1, ahonnan t-be el lehet jutni?

$s^? \to t_1$ ha $(t_1, t) \in A$

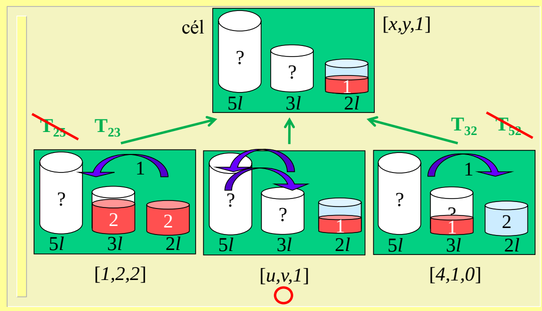
Visszafelé haladás

$s^? \to t$

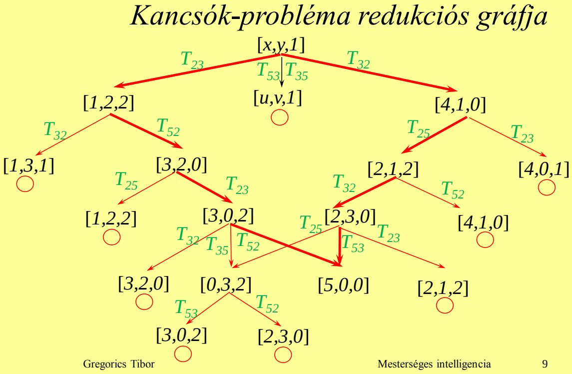
Van-e olyan t_1, ahová t-ből el lehet jutni?

$s^? \to t_1$ ha $(t, t_1) \in A$

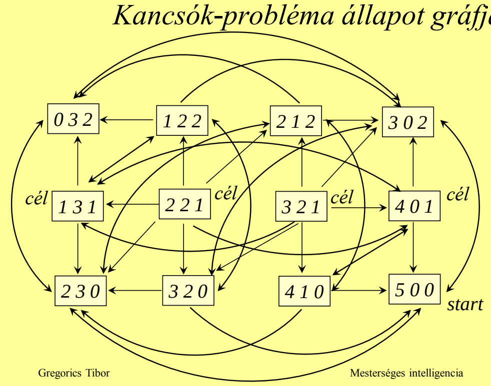
Kancsó-probléma redukálása



Kancsók-probléma redukciós gráfja



Kancsók-probléma állapot gráfja



Redukciós reprezentáció fogalma

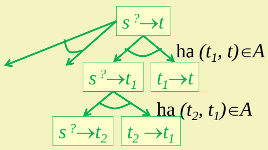
- A reprezentációhoz meg kell adnunk a feladat
  - állapottér-reprezentációját,
  - majd minden művelethez definiálunk egy redukciós műveletet, amely egy állapothoz azokat a megelőző állapotokat rendeli, amelyekből a rögzített művelet az aktuális állapotba vezet.
    - M művelethez tartozó redukciós művelet:
      $B_M \subseteq állapot \times állapot$ és
      $b \in B_M(a)$ ha $M(b)=a$

Megjegyzés

- A redukció során eljuthatunk „érdektelen” illetve „hamis” állapot-leírásokhoz.
- A probléma redukcióhoz gráfreprezentáció készíthető: ebben kell utat keresni.
- A talált (célból startba vezető) út visszafelé olvasva adja ki a megoldást, amely nem az inverze a talált útnak.

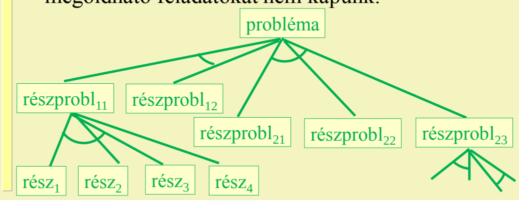
A redukció egy speciális dekompozíció

- A redukció során a megoldandó feladatot mindig két részre: egy nyilvánvalóan megoldható és egy további redukálást igénylő részfeladatra bontottuk.

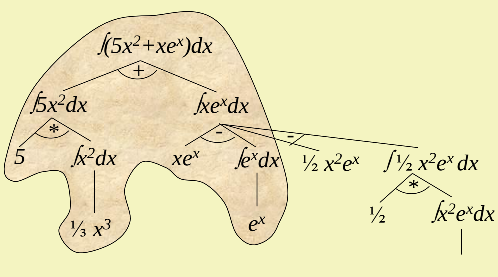


3. Dekompozíció

- A probléma dekompozíció lényege az, hogy egy feladatot részfeladatokra bontunk, majd azokat tovább részletezzük, amíg nyilvánvalóan megoldható feladatokat nem kapunk.



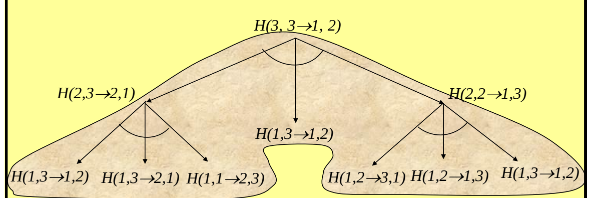
Integrálszámítás



Hanoi tornyai probléma megoldása dekompozícióval

$H(n, i\to j, k)$ helyett

$H(n-1, i\to k, j)$ $H(1, i\to j, k)$ $H(n-1, k\to j, i)$



Dekompozíciós reprezentáció fogalma

- A reprezentációhoz meg kell adnunk:
  - a feladat részproblémáinak általános leírását,
  - az eredeti problémát,
  - az egyszerű problémákat, amelyekről könnyen eldönthető, hogy megoldhatók-e vagy sem, és
  - a dekomponáló műveleteket:
    - D: *probléma* $\to$ *probléma*$^+$ és
      $D(p)=<p_1, \ldots , p_n>$

A dekompozíciós reprezentáció nehéz

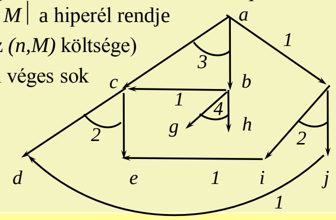
- Dekomponáló műveleteket nagyon nehéz megtalálni.
  - Nem biztos, hogy megtaláljuk.
  - Hamis dekomponáló műveletek.
  - Nem minden feladat dekomponálható.
- Az egyszerű probléma felismerése sem egyértelmű
  $\int sin(x)e^xdx = \ldots = sin(x)e^x - cos(x)e^x - \int sin(x)e^xdx$
- A megoldás kiolvasása sem nyilvánvaló.

Gráfreprezentáció

- A feladat problématerét nem egy közönséges irányított gráf írja le, hanem egy úgynevezett ÉS/VAGY gráf.
- A megoldást sem egy közönséges irányított út szimbolizálja, hanem egy speciális részgráf: a megoldásgráf
  - A megoldásgráfnak egyértelmű haladási irányt kell kijelölnie a startcsúcsból a célcsúcsokba.
- A probléma megoldása a megoldásgráfból olvasható ki.

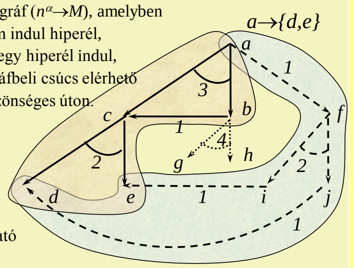
4. ÉS/VAGY gráfok

- Az $R=(N,A)$ élsúlyozott irányított hipergráf, ahol az
  - N a csúcsok halmaza,
  - $A \subseteq \{ (n,M) \in N \times 2^N \mid 0 \neq |M| < \infty \}$ a hiperélek halmaza, $|M|$ a hiperél rendje
  - ($c(n,M)$ az (n,M) költsége)
- Egy csúcsból véges sok él indulhat
- $0 < \delta \leq c(n,M)$



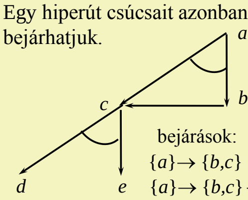
Az n csúcsból az M csúcshalmazba vezető irányított hiperút fogalma

- A hiperút egyértelmű haladási irányt kijelölő hiperélek halmaza, azaz
- egy olyan véges részgráf ($n^{\alpha} \rightarrow M$), amelyben
  - M csúcsaiból nem indul hiperél,
  - a többi csúcsból egy hiperél indul,
  - bármelyik részgráfbeli csúcs elérhető az n csúcsból közönséges úton.
- hiperútnak definiálható a hossza, a költsége



Különbség a közönséges út és a hiperút bejárása között

- Egy közönséges út bejárásán az úton fekvő csúcsoknak az élek által mutatott sorrendjében történő felsorolását értjük. Ez mindig egyértelmű.
- Egy hiperút csúcsait azonban többféle sorrendben is bejárhatjuk.



bejárások:

$\{a\} \rightarrow \{b,c\} \rightarrow \{c\} \rightarrow \{d,e\}$

$\{a\} \rightarrow \{b,c\} \rightarrow \{b,d,e\} \rightarrow \{c,d,e\} \rightarrow \{d,e\}$

Hiperút bejárása

- Az $n \rightarrow M$ hiperút egy bejárásán a hiperút csúcsaiból képzett halmazoknak azt a felsorolását értjük, amelyben
  - Az első az $\{n\}$ halmaz, a második az n csúcsból kivezető (a hiperúton fekvő) hiperél utódhalmaza, ha van ilyen hiperél.
  - Általában a felsorolás egy C halmaza után a $C-\{k\} \cup K$ halmaz következik, ha a hiperúton van olyan (k,K) hiperél, hogy $k \in C$ és $k \notin M$.
  - A bejárás az M csúcshalmazzal fejeződik be.

Megjegyzés

- A bejárásra úgy is tekinthetünk, mint a hiperút összes hiperélét tartalmazó hiperél-sorozatra, amelyben ugyanaz a hiperél többször is szerepelhet.

1. Közönséges irányított kört nem tartalmazó $n \rightarrow M$ hiperútnak véges sok különböző bejárása van, ezek mindegyike véges hosszú, és utolsó halmaza az M.
2. Közönséges irányított kört is tartalmazó hiperútnak nincs véges hosszú bejárása.
3. Az $n \rightarrow M$ hiperút (k,K) hiperéle legfeljebb annyiszor szerepel egy bejárás során, amennyi közönséges út vezet a hiperútban az n csúcsból k csúcshoz.

Dekompozíciós gráfreprezentáció

- Egy dekompozíciós reprezentációhoz tartozó (R,s,T) gráfreprezentációban
  - az $R=(N,A,c)$ egy olyan ÉS/VAGY gráf (dekompozíciós gráfban), ahol
  - N a részprobémákat,
  - A a dekomponáló műveleteket,
  - c azok költségeit szimbolizálják,
  - s az eredeti problémát,
  - T az egyszerű problémákat jelöli.

Megjegyzés

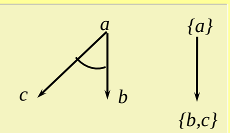
- A probléma megoldását egy $s \to M \subseteq T$ közönséges irányított kört nem tartalmazó hiperút, az úgynevezett megoldásgráf megtalálása jelenti. Az eredeti probléma megoldása ebből a megoldásgráfból nyerhető ki.
- A megoldás költsége többnyire nem függ a megoldásgráf költségétől, ezért nem cél az optimális megoldásgráf előállítása.

Keresés ÉS/VAGY gráfban

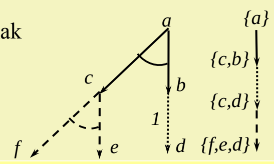
- Egy ÉS/VAGY gráfon folyó keresés a startcsúcsból kivezető hiperutak (köztük vannak a megoldásgráfok) között folyik.
- Minden ÉS/VAGY gráfnak megfeleltethető egy olyan közönséges irányított δ-gráf , amelynek megoldási útjai a megoldásgráfok bejárásai, ennél fogva a közönséges gráfokra felírt útkereső algoritmusok ÉS/VAGY gráfokbeli keresésekként adaptálhatók.
- Hogyan adható meg az ÉS/VAGY gráfoknak egy „megoldás tartó” átalakítása közönséges gráfokká?

Bejárások ábrázolása közönséges utakkal

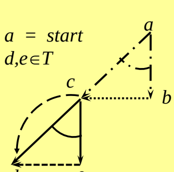
Egyetlen hiperélből álló útnak egyetlen bejárása van

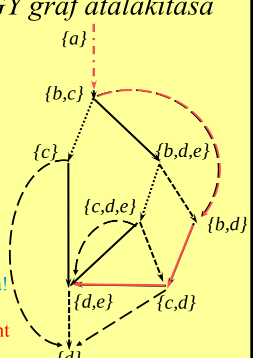


Több hiperélből álló útnak egyik bejárása



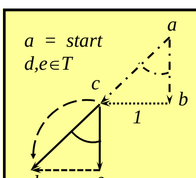
ÉS/VAGY gráf átalakítása

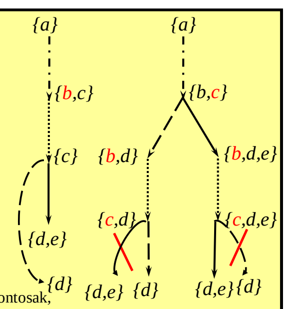




A startból induló hiperutak bejárásait közönséges útként rajzoljuk fel. (Az átalakítás nem költségtartó.)

Nem kell egy hiperútnak minden bejárása! Törölni kell a hamis bejárásokat, azokat, ahol ugyanaz a csúcs többször, de másként kerül helyettesítésre!





hamis bejárások!

1. Egy hiperútnak elég egy bejárását megadni, ezért az átalakítás egy lépésében elég egy csúccsal foglalkozni

2. Számunkra csak az $s \to M \subseteq T$ hiperutak a fontosak, ezért a célcsúcsokból már ne lépjünk tovább.

3. A hamis bejárások kizárása érdekében a közönséges gráfot fává kiegyenesítve adjuk meg, továbbá ha egy csúcshalmazból olyan k csúcsot választunk a továbblépéshez, amelyből egyszer már továbbléptünk egy (k,K) hiperéllel a k-t tartalmazó csúcshalmazhoz vezető úton , akkor itt is ezt a hiperélt használjuk fel a továbblépéshez.

Átalakító algoritmus

- Betesszük egy SOR-ba az {*s*} halmazt, mint startcsúcsot.
- Amíg a SOR nem üres addig kiveszünk a SOR-ból egy *C* halmazt, és generáljuk a *C*-ből kivezető éleket:
  - Ha *C* csupa hipergráfbeli célcsúcsból áll, akkor a *C* halmaz definíció szerint célcsúcsot jelöl, amelyből nem indítunk élt.
  - Különben legyen *k* *C*-beli nem célcsúcs.
    - Ha a startból a *C*-hez vezető úton van már olyan él, amelyet egy $(k,K) \in A$ hiperéllel generáltunk, akkor ugyanezzel a (k,K) hiperéllel generálunk most is egy $C\text{-}\{k\} \cup K$ utódot a *C*-hez.
    - Különben az összes $(k,K) \in A$ hiperéllel generálunk egy-egy $C\text{-}\{k\} \cup K$ utódot a *C*-hez.
  - Az utódokat betesszük a SOR-ba.

Tétel

1. Az átalakítással nyert közönséges gráfok minden megoldási útja egy $s \to M \subseteq T$ hiperútnak egy bejárását írja le.
2. Az átalakítás minden $s \to M \subseteq T$ hiperút valamelyik bejárásához véges lépésben megfeleltet egy közönséges megoldási utat.

- Megjegyzés:
  - Az átalakított gráf egy δ-gráf (költségek!)
  - Az átalakítást beépítik a keresésekbe.

Visszalépéses keresés ES/VAGY gráfokon

<u>Recursive procedure</u> *VL2*(*bejárás*) <u>return</u> *megoldás*

1. *C* := *vége*(*bejárás*)
2. <u>if</u> *csupacél*(*C*) <u>then</u> <u>return</u>(*nil*) <u>endif</u>
3. <u>if</u> *hossza*(*bejárás*) ≥ korlát <u>then</u> <u>return</u>(*hiba*) <u>endif</u>
4. <u>if</u> *C* ∈ *maradék*(*bejárás*) <u>then</u> <u>return</u>(*hiba*) <u>endif</u>

5a. *k := kivesz-egy-nemcélcsúcsot*(*C*)

5b. *hiperélek := kivezető-hiperélek*(*k*)

6. <u>while</u> not *üres*(*hiperélek*) <u>loop</u>
7.   *(k,K) :=* kivesz(*hiperélek*)
8.   *megoldás := VL2*(*hozzáfűz*(*bejárás, C-{k}∪K*))
9.   <u>if</u> *megoldás* ≠ *hiba* <u>then</u>
        <u>return</u>(*hozzáfűz*((*C, C-{k}∪K*)*, megoldás*))
    <u>endif</u>
10. <u>endloop</u>
11. <u>return</u>(*hiba*)

<u>end</u>