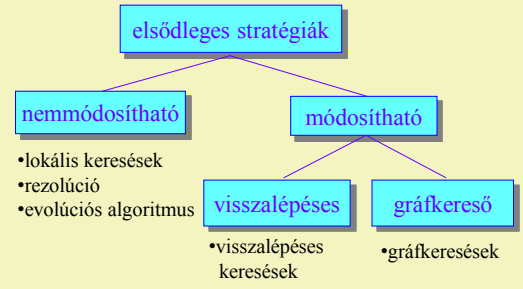
III. Keresések

Elsődleges vezérlési stratégiák

ADAT := *kezdeti érték*
while ¬ *terminálási feltétel*(ADAT) loop
  SELECT SZ FROM *alkalmazható szabályok*
  ADAT := SZ(ADAT)
endloop



1. Lokális keresések

- A lokális keresés olyan KR, amelynek globális munkaterületén a reprezentációs gráf egyetlen csúcsát, az aktuális csúcsot (kezdetben a startcsúcsot) és annak környezetét tároljuk.
- Az aktuális csúcsot minden lépésben annak környezetéből vett lehetőleg „jobb" csúccsal cseréljük le (keresési szabály).
- A jobbság eldöntéséhez egy célfüggvényt (rátermettségi függvényt, heurisztikus függvényt) használunk, amely várhatóan annál jobb értéket ad egy csúcsra, minél közelebb esik az a célhoz. (vezérlési stratégia)
- A keresés leáll, ha megtaláljuk a célcsúcsot.

Hegymászó algoritmus

- Minden lépésben az aktuális (*akt*) csúcs legjobb gyermekére lép, amelyik nem a szülője.

1. $akt := s$
2. **while** $akt \notin T$ **loop**
3. **if** $\Gamma(akt) - \pi(akt) = \emptyset$ **then return** nem talált megoldást
4. $akt := \mathbf{opt}_f(\Gamma(akt) - \pi(akt))$
5. **endloop**

Megjegyzés

- Hátrányok:
  - csak erős heurisztika esetén lesz sikeres: egy rossz döntés véglegesen tévútra vezetheti
  - lokális optimum hely környezetében és ekvidisztans felületen (azonos értékű szomszédos csúcsok között) eltévedhet; mivel nincs kör figyelés, kör mentén végtelen ciklusba esik
  - zsákutcában (szülőre nem léphet vissza) beragad
- A bajok okai:
  - túl kicsi az algoritmus memóriája ⟹ tabu keresés
  - mohó stratéga ⟹ szimulált hűtés algoritmusa

Tabu keresés

- Az aktuális csúcson (*akt*) kívül nyilvántartja még
  - Optimális csúcs (*opt*): az eddigi legjobb csúcs
  - Tabu halmaz: az utolsó néhány érintett csúcsot
- Minden lépésben
  - Az aktuális csúcs legjobb gyermekére lép, kivéve a tabu halmazban levőket
  - ha *akt* jobb, mint az *opt*, akkor *opt* az *akt* lesz
  - frissíti *akt*-tal a sorszerkezetű tabu halmazt
- Terminálási feltételek:
  - ha a célfüggvény az *opt*-ban optimális
  - ha az *akt* célfüggvény értéke vagy az *opt* sokáig nem változik.

Tabu keresés algoritmusa

1. *akt, opt, Tabu := startcsúcs, startcsúcs,* ∅
2. **while not** (*opt*∈*T* **or** *opt* régóta nem változik) **loop**
3. **if** Γ(*akt*)*Tabu* = ∅ **then return** nem talált megoldást
4. *akt* := **opt**$_f$(Γ(*akt*)*Tabu*)
5. *Tabu* := Módosít(*n*,*Tabu*)
6. **if** *f*(*akt*) jobb, mint *f*(*opt*) **then** *opt* := *akt*
7. **endloop**

Megjegyzés

- Előnyök:
  - tabu méreténél rövidebb köröket észleli, és ez segíthet a lokális optimum hely illetve az ekvidisztans felület leküzdésében is.
- Hátrányok:
  - a tabu halmaz méretét kísérletezéssel kell belőni
  - a keresés továbbra is beragadhat, ha az összes továbblépési irányt a tabu halmaz védi (habár ekkor megengedhető a tabu megsértése)

Szimulált hűtés

- A következő csúcs kiválasztása az aktuális (*akt*) csúcs gyermekei közül véletlenszerű.
- Ha az így kiválasztott *m* csúcs célfüggvény-értéke nem rosszabb, mint az *akt* csúcsé (itt $f(m) \le f(akt)$), akkor elfogadjuk, ha rosszabb ($f(m) > f(akt)$), akkor az *m* elfogadása is véletlenített: ennek valószínűsége fordítottan arányos az $|f(akt) - f(m)|$ különbséggel.

$$e^{f(akt)-f(m)} > random[0,1]$$

Hűtési ütemterv

- Egy ugyanannyival rosszabb célfüggvény értékű új csúcsot a keresés kezdetén nagyobb eséllyel fogadjuk, mint később.
- Ehhez elég a keresés idejétől fordított arányban változó együtthatókkal leosztani az $f(akt) - f(m)$ különbséget.

f(m)=120, f(akt)=107

T	*exp(-13/T)*
10^{10}	0.9999…
50	0.77
20	0.52
10	0.2725
5	0.0743
1	0.000002

- Adjunk ütemtervet: (T_k , L_k) *k=1,2,…*
  - Legyenek a *T* értékei: T_1 , T_2 , …
  - amelyeket rendre L_1 , L_2 , … lépésben használunk
  - A T_1 , T_2 , … legyen szigorúan monoton csökkenő.

Szimulált hűtés algoritmusa

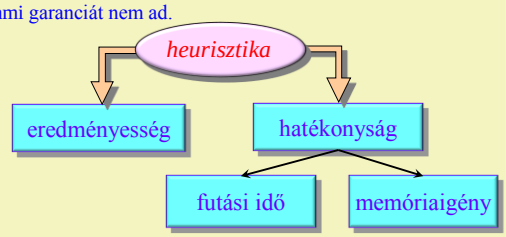
1. *akt := s; k := 1*
2. **while not**(*akt*∈*T* **or** *f*(*akt*) régóta nem változik) **loop**
3. **for** *i = 1 .. L*$_k$ **loop**
4. *m* := **select**(Γ(*akt*)\π(*akt*))
5. **if** $f(m) \le f(akt)$ **or**
6. $f(m) > f(akt)$ **and** $e^{\frac{f(akt)-f(m)}{T_k}} > rand[0,1]$
7. **then** *akt := m*
8. **endloop**
9. *k := k+1*
10. **endloop**

Lokális kereséssel megoldható feladatok

- A lokálisan hozott rossz döntés ne zárja ki a cél megtalálását
  - Ez biztos teljesül, ha reprezentációs-gráf erősen összefüggő
  - Jó heurisztikára épített célfüggvénnyel elkerülhetőek a zsákutcák vagy az olyan csúcsok, ahonnan már nem vezet út célcsúcsba.
  - Előnytelen, ha a reprezentációs-gráf egy irányított fa. (Ezért például az *n*-királynő probléma nem alkalmas lokális kereséssel való megoldáshoz, csak tökéletes heurisztikájú célfüggvény esetén)

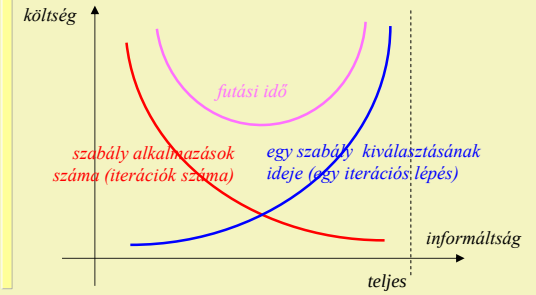
A heurisztika hatása a KR működésére

A heurisztika olyan, a feladathoz kapcsolódó ötlet, amelyet közvetlenül építünk be egy algoritmusba azért, hogy annak eredményessége és hatékonysága javuljon, habár erre általában semmi garanciát nem ad.



A KR futási ideje

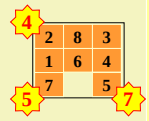
ADAT := *kezdeti érték*
while ¬ *terminálási feltétel*(ADAT) loop
  SELECT SZ FROM *alkalmazható szabályok*
  ADAT := SZ(ADAT)
endloop



Heurisztikák a 8-as (15-ös) tologató játékra

- Rossz helyen levő cellák száma:

$$W(a) = \sum_{a_{i,j} \neq 0 \wedge a_{i,j} \neq cél_{i,j}} 1$$



- Cellák célbeli helyüktől vett minimális távolságainak összege (Manhattan):

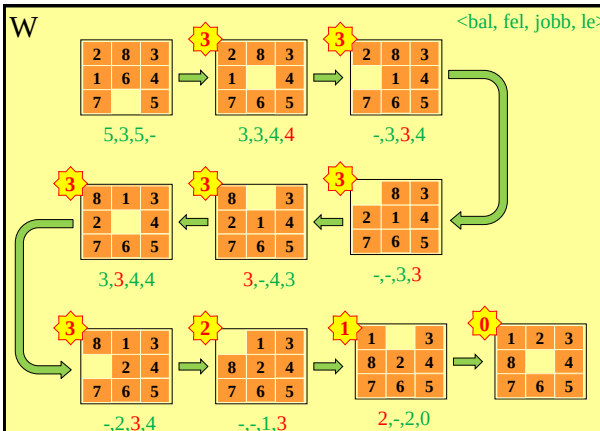
$$P(a) = \sum_{i,j} (|i - célbelisor(a_{i,j})| + |j - célbelioszlop(a_{i,j})|)$$

$célbelisor(a_{i,j})$ ~ az $a_{i,j}$ célállapotbeli helyének sora

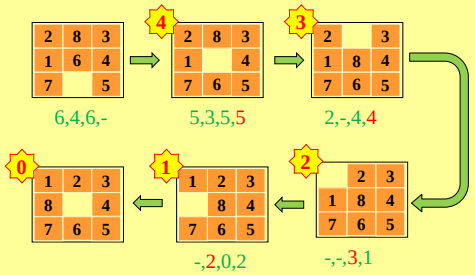
$célbelioszlop(a_{i,j})$ ~ az $a_{i,j}$ célállapotbeli helyének oszlopa

- „Széleken levő cellák legyenek jók” (frame):
  - Hány olyan lapocska van a szélen, amelyiket nem a célbeli szomszédja követ az óra járásával megegyező irányban?
  - Hány lapocska nincs még a szélen?
  - Hány sarokban (×2) nincs még a cél szerinti lapocska?

W <bal, fel, jobb, le>



P <bal, fel, jobb, le>



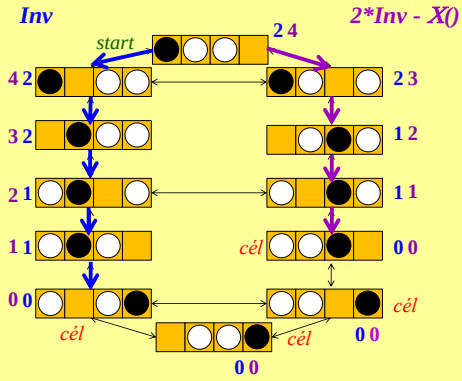
Heurisztikák a Fekete-fehér kirakóra

- Inverziószám:

  $I(v)$= minimálisan hány csere kell ahhoz, hogy minden fehér minden feketét megelőzzön

- $D(v) = 2*I(v)$
- Módosított inverziószám:

  $M(v) = 2*I(v) - (1$, ha v-nek része vagy)



Heurisztikák a Hanoi tornyai problémára

- Darab: $C(v) = \sum_{\substack{i=1..n \\ v[i]\neq 1}} 1$
- Súlyozott darab: $WC(v) = \sum_{\substack{i=1..n \\ v[i]\neq 1}} i$
- Összeg: $S(v) = \sum_{i=1..n} v[i]$
- Súlyozott összeg: $WS(v) = \sum_{i=1..n} i*v[i]$
- Módosított összeg: $EWS(a) = WS(v) - \sum_{v[i-1]>v[i]} 1 + \sum_{v[i-1]=v[i+1]\wedge v[i]\neq v[i-1]} 2$

Heurisztikák a Hanoi tornyai problémára

- Tökéletes: $P(v) = f(1)_1$ $\quad f:[1..n+1]\to\mathbb{N}\times\mathbb{N}$

  $f(i)_1$: célba éréshez szükséges lépések száma, ha csak az i és n közötti korongok számítanak, a kisebb korongok mintha nem is lennének.

  $f(i)_2$: az $i-1$-dik korongnak a helye ahhoz, hogy a nálánál nagyobb korongok mozgatásának megkezdésekor ne legyen „útban”.

  Az i-dik korong a nálánál nagyobb korongok mozgatása közben állandóan „útban” lesz (kivéve a legelső lépésnél, ha éppen az $f(i+1)_2$ rúdon van). Ezért az i-dik korongot minden nagyobb korong mozgatása előtt félre kell tenni (ettől megduplázódik a lépések eddig kalkulált száma), majd a legvégén a helyére.

$f(n+1) = (0, 1)$

$$f(i) = \begin{cases} (2*f(i+1)_1 \;,\; f(i+1)_2\;) & \text{ha } v[i]=f(i+1)_2 \\ (2*f(i+1)_1+1,\; 6-f(i+1)_2-v[i]\;) & \text{ha } v[i]\neq f(i+1)_2 \end{cases}$$