UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Proiectarea Algoritmilor

30. Algoritmul A*

Bibliografie

[1] C. Giumale – Introducere in Analiza Algoritmilor - cap. 7

[2] http://www.gamasutra.com/features/19990212/pathdemo.zip

[3] http://www.policyalmanac.org/games/aStarTutorial.htm

[4] http://www.ai.mit.edu/courses/6.034b/searchcomplex.pdf

A*

- Variantă a BF*.
- Nu poate fi aplicat mereu → trebuie demonstrat că păstrează ordinea soluțiilor unde soluțiile problemelor sunt drumuri în spațiul stărilor! (vezi Giumale pentru detalii!)
- Costul unui drum este aditiv (= suma costurilor arcelor) și crescător în lungul drumului.
- Folosește două funcții de cost:
  - h(n) - distanța estimată de la nodul curent până la nodul țintă;
  - g(n) - distanța parcursă de la nodul inițial până la nodul curent;
  - f(n) = g(n) + h(n).

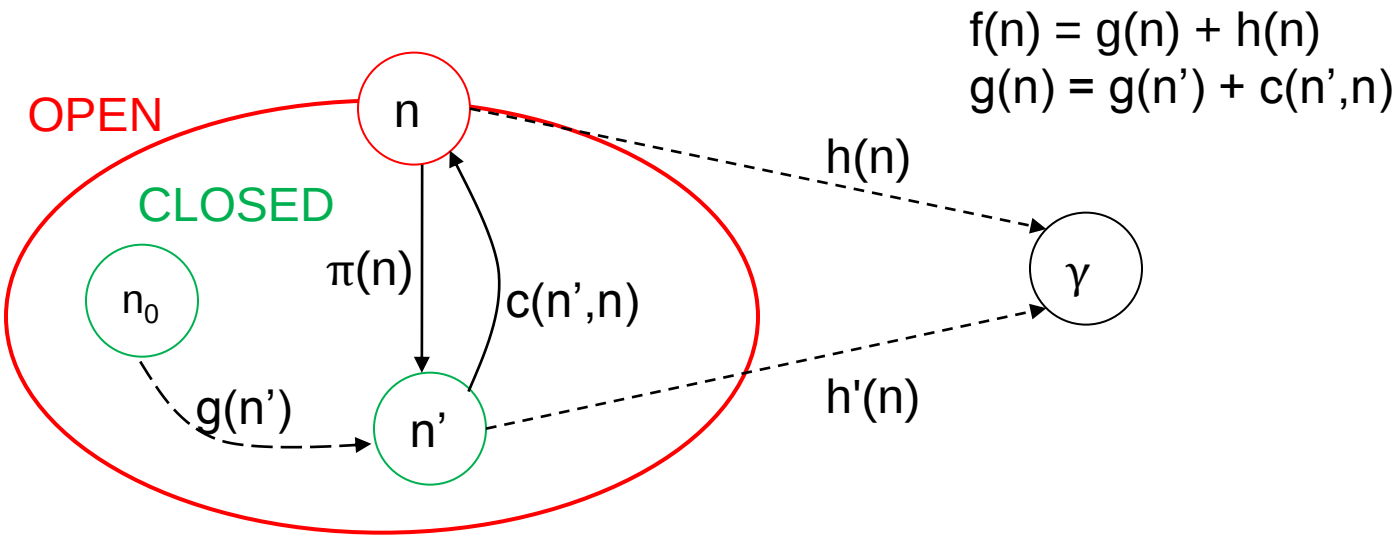
Notații (1)

- **S = (V,E)** – graful asociat spațiului stărilor problemei;
- **n_0** – nodul de start asociat stării inițiale a problemei;
- **$\Gamma \subseteq V$** – mulțimea nodurilor soluție. Un nod soluție se notează γ;
- **c(n,n') > 0** – costul arcului (n,n');
- **π(n)** – părintele lui n;
- **g(n)** – costul drumului n_0..n descoperit de algoritm la momentul curent de timp;
- **g_p(n)** – costul exact al porțiunii n_0..n din lungul unei căi date P;
- **g*(n)** – costul exact al unui drum optim n_0..n;

Notații (2)

- **$h(n) \geq 0$** – costul estimat al drumului optim de la nodul n la cel mai favorabil nod soluție $\gamma \in \Gamma$. În plus $h(\gamma) = 0$, pentru orice $\gamma \in \Gamma$;
- **$h^*(n)$** – costul exact al porțiunii de drum optim $n..\gamma$, pentru cel mai favorabil nod $\gamma \in \Gamma$ ($h^*(n) = \min \{cost(n..\gamma) | \gamma \in \Gamma\}$);
- **$f(n) = g(n) + h(n)$** – costul estimat al întregului drum $n_0..n..\gamma$, pentru cel mai favorabil nod $\gamma \in \Gamma$, unde porțiunea de drum $n_0..n$ este cea descoperita de algoritm la momentul curent de timp în cursul execuției;
- **$f^*(n) = g^*(n) + h^*(n)$** – costul exact al unui drum optim $n_0..n..\gamma$, pentru cel mai favorabil nod $\gamma \in \Gamma$;
- **$C = \min\{f^*(\gamma) | \gamma \in \Gamma\}$** – costul exact al unui drum optim $n_0..\gamma$, $\gamma \in \Gamma$. (C = costul soluției optime);

Funcția de evaluare A*



A* (1)

- A*(StInit, h, test_sol)
  - n_0 = constr_nod(StInit); // starea inițială

    Inițializări
  - $f(n_0) = h(n_0)$; $g(n_0) = 0$; $\pi(n_0) = null$; // euristici
  - OPEN = $\{n_0\}$; CLOSED = Ø; // și cozi
  - **Cât timp** (OPEN ≠ Ø) // mai am noduri de prelucrat
    - nod = selecție nod (OPEN); // f(nod) = min {f(n) | n ∈ OPEN}
    - **Dacă** (test_sol(nod)) **Întoarce** nod;

      Soluția
    - OPEN = OPEN \ {nod}; // updatez OPEN
    - CLOSED = CLOSED U {nod}; // și CLOSE
    - succs = expand(nod); // determin nodurile succesoare

      Continuarea căutării

A* (2)

- **Pentru fiecare** (succ ∈ succs) { // prelucrare succs

  Prelucrare succesori

  - g_succ = g(nod) + c(nod,succ); // calculez g
  - f_succ = g_succ + h(succ); // calculez f = g + h
  - **Dacă** (succ ∉ CLOSED U OPEN) **atunci** // nod nou descoperit →

    Nod nou

    - { OPEN = OPEN U {succ}; g(succ)= g_succ; f(succ)= f_succ; π(succ) = nod;} // il bag in OPEN
  - **altfel** // a mai fost prelucrat
    - **Dacă** (g_succ < g(succ)) { // verific daca noul g este mai mic decât // anteriorul

      Actualizări

      g(succ)= g_succ; f(succ)= f_succ; π(succ) = nod; // cale mai bună

      Dacă (succ ∈ CLOSED) // dacă era considerat expandat, îl redeschid

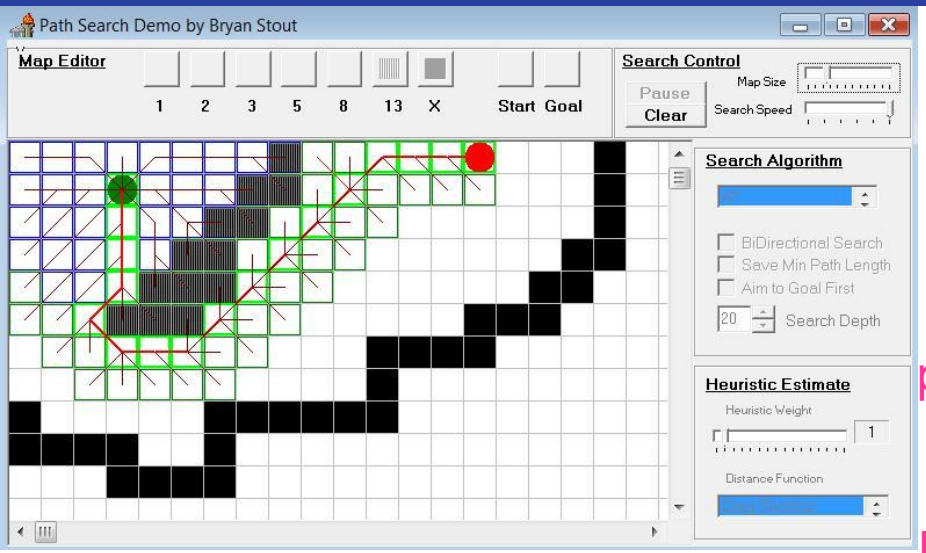
      Reprelucrare

      { CLOSED = CLOSED \ {succ'}; OPEN = OPEN U {succ'};
- **Întoarce** insucces;

  Insucces

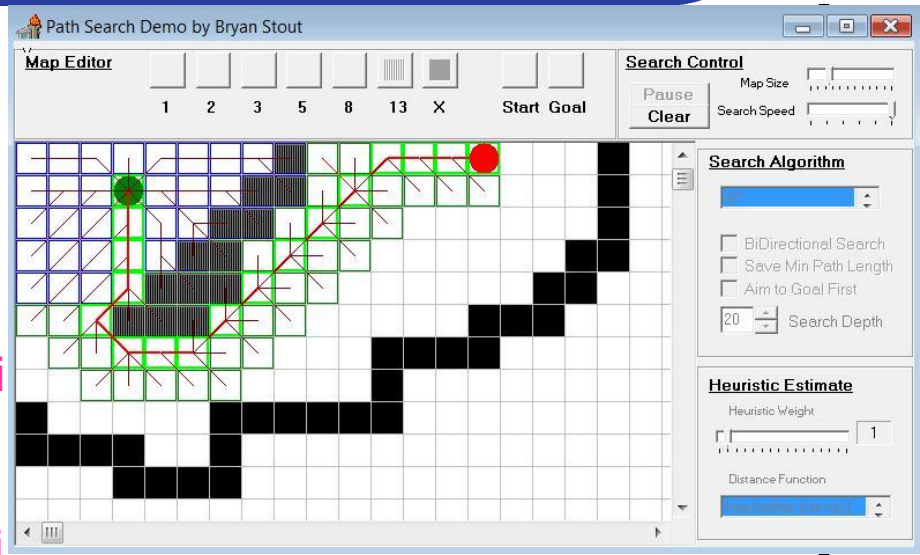
ex: A* cu diverse euristici

Exemple A* cu diverse euristici



A*

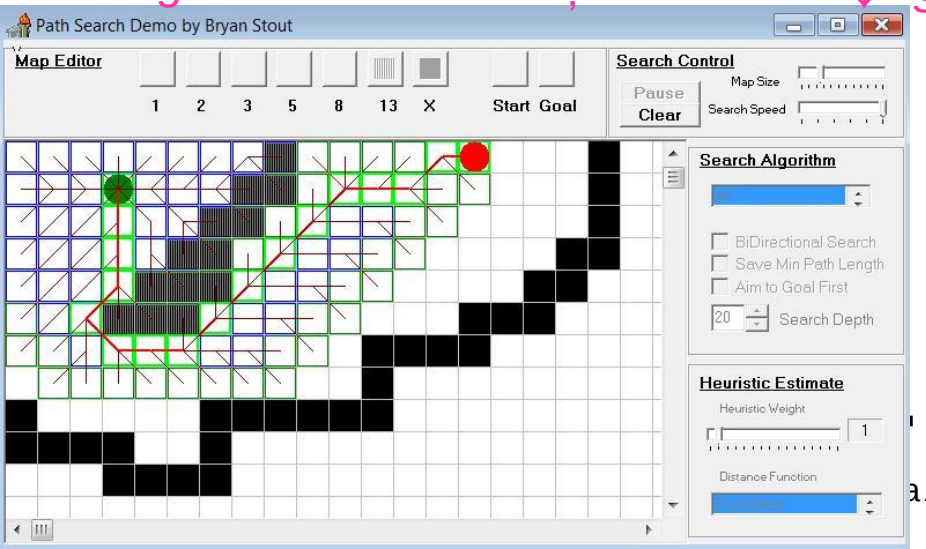
16 pași



Euristici:

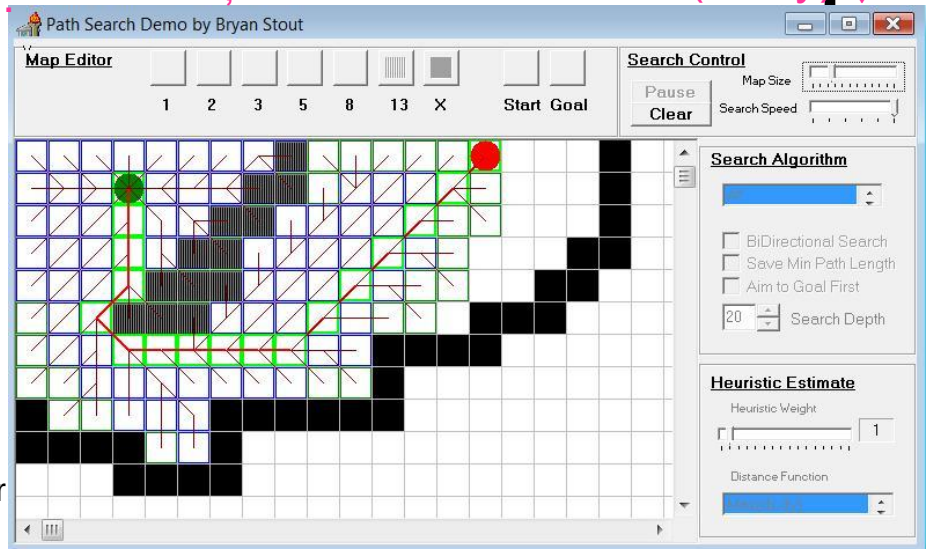
Diagonala↑

Distanța Euclidiana↓



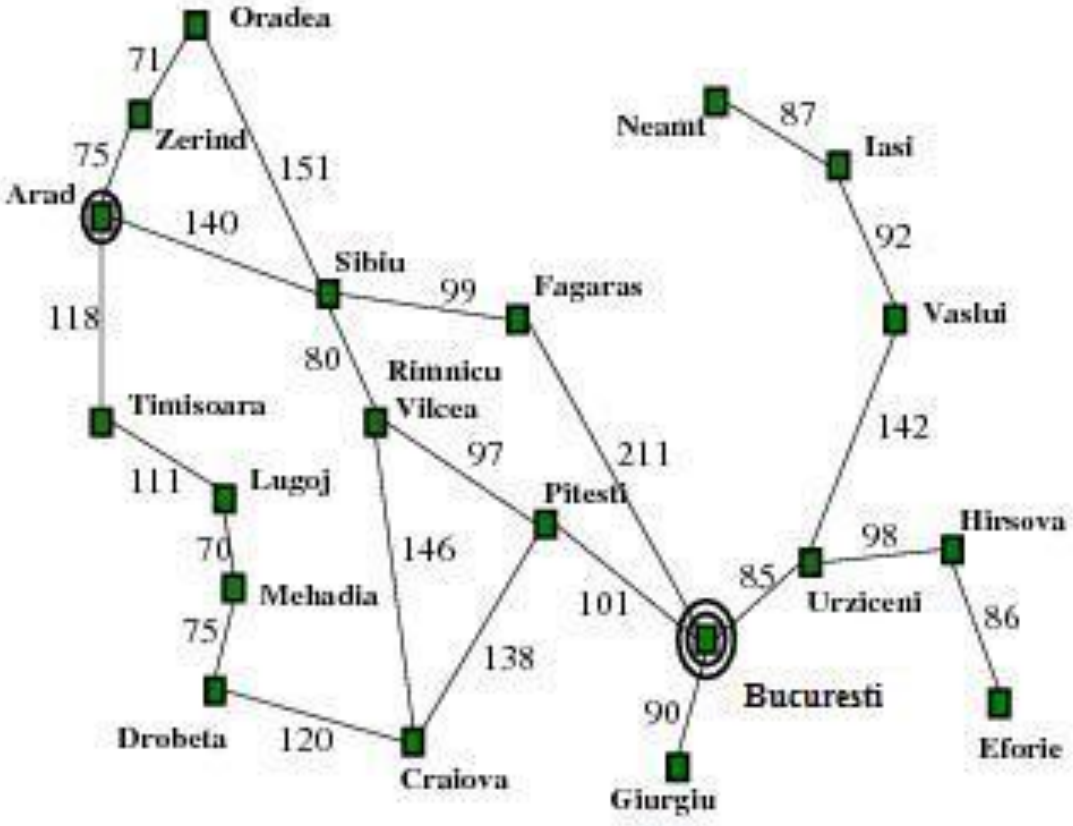
Distanța Manhattan ↑

Max(dx,dy) ↓



a Algor

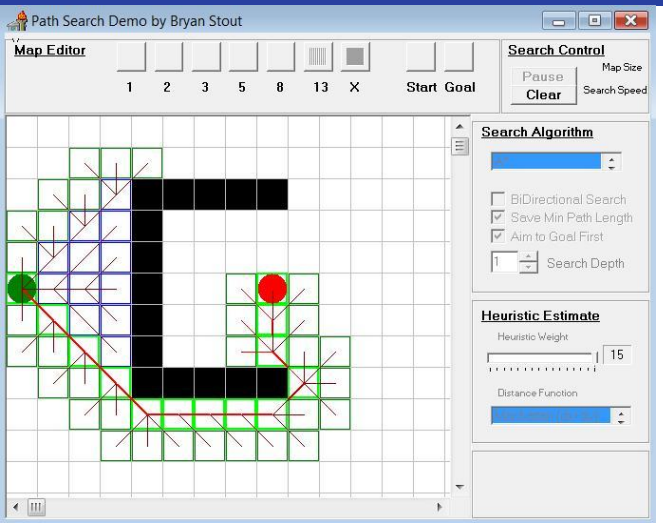
Aplicație A*



Distanta in linie dreapta pana la Bucuresti	
Arad	366
Craiova	160
Drobeta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

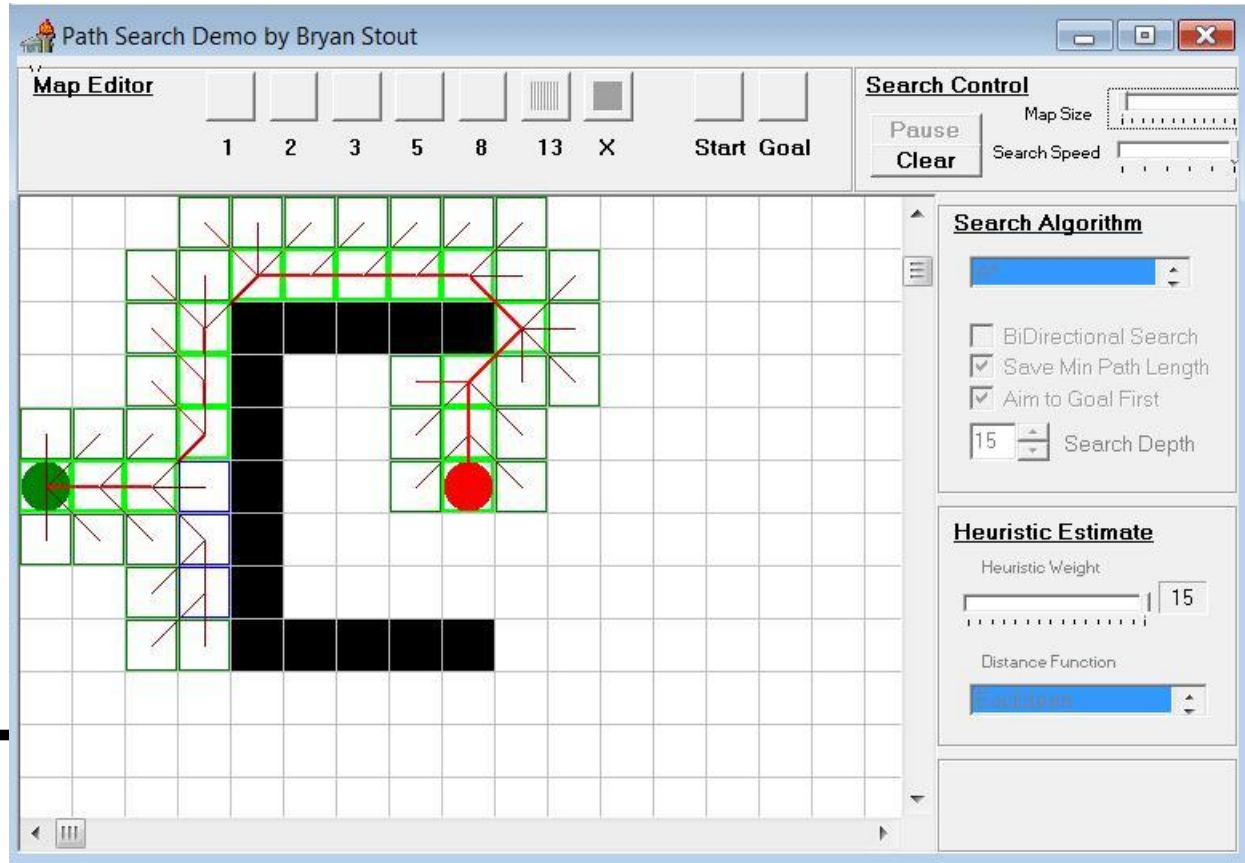
- Drumul optim Arad-București (h(n) = distanța in linie dreaptă pană la București, g(n) = distanța parcursă)

Problema



← A* – Distanța Manhattan – 12 pași

↓ A* – Distanța Euclidiana – 14 pași



Cum se explică???

Algoritmul A* - completitudine și optimalitate (1)

- Teorema 7.1: Algoritmul A* este complet chiar dacă graful explorat nu este finit.

- Lema 7.1: Fie $P = n_0,n_1,...,n_m$ un drum oarecare în graful explorat de A*, astfel încât la un moment T al explorării toate nodurile din P sunt în CLOSED. Atunci, la orice moment de timp egal sau superior lui T, există inegalitatea $g(n_i) \leq g_p(n_i)$, $i = 0,m$:
  - costul nodurilor din CLOSED poate să scadă, dar de fiecare dată când acest lucru se întâmplă, se pierde timp → scoaterea nodului din CLOSED, punerea în OPEN, prelucrarea acestuia încă o dată → trebuiesc evitate aceste situații → alegerea unei euristici cât mai bune care să minimizeze numărul acestor actualizări!

Algoritmul A* - completitudine și optimalitate (2)

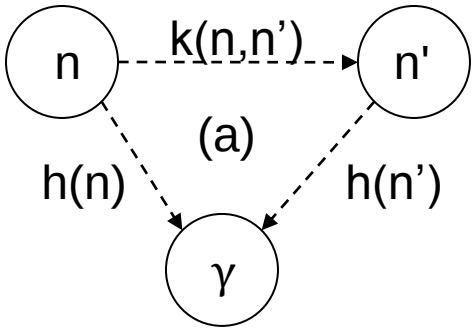
- Definiție 7.2: Funcția euristică h este admisibilă dacă pentru orice nod n din spațiul stărilor $h(n) \leq h^*(n)$. Cu alte cuvinte, o euristică admisibilă h este optimistă și $h(\gamma) = 0$ pentru orice nod $\gamma \in \Gamma$.

- Teorema 7.2: Algoritmul A* ghidat printr-o euristică admisibilă descoperă soluția optimă dacă există soluții.

Euristici – consistență si monotonie

- Definiție 7.4: O euristică h este consistentă dacă pentru oricare două noduri n și n' ale grafului explorat, astfel încât n' este accesibil din n, există inegalitatea: $h(n) \leq h(n') + k(n,n')$, unde $k(n,n')$ este costul unui drum optim de la n la n'.

- Definiție 7.5: O euristică h este monotonă dacă pentru oricare două noduri n și n' ale grafului explorat, astfel încât n' este succesorul lui n, există inegalitatea $h(n) \leq h(n') + c(n,n')$, unde $c(n,n')$ este costul arcului (n,n').

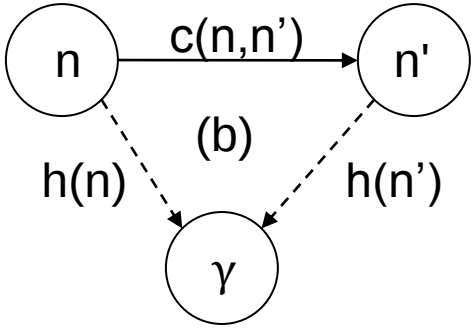


$h(n) \leq h(n') + k(n,n')$

Regula triunghiului pentru euristici:

← Consistență

→ Monotone



$h(n) \leq h(n') + c(n,n')$

Consistență = monotonie

- Teorema 7.5: O euristică este consistentă ⇔ este monotonă.
  - Demonstrație:
    - h – consistentă → h – monotonă. Alegem n' ∈ succs(n) → k(n,n') = c(n,n') → h(n) ≤ h(n') + c(n,n') → h – monotonă.
    - h – monotonă → h – consistentă. Fie n = $n_1,n_2,\ldots,n_q$ = n', un drum optim n..n' cu cost k(n,n'). → h(n) = $h(n_1) \leq h(n_2) + c(n_1,n_2) \leq h(n_3) + c(n_1,n_2) + c(n_2,n_3)\ldots \leq h(n_q) + c(n_1,n_2) + c(n_2,n_3) + \ldots c(n_{q-1},n_q) = h(n_q) + k(n_1,n_q)$ → h(n) ≤ h(n') + k(n,n') → h – consistentă.

Consistență → admisibilitate

- Teorema 7.6: O euristică consistentă este admisibilă.
  - Demonstrație:
    - Fie h o euristică consistentă → $h(n) \leq h(n') + k(n,n')$, $\forall$ n' accesibil din n. Fie $n' = \gamma \in \Gamma$ → $k(n, \gamma) = \min \{ k(n, \gamma') \mid \gamma' \in \Gamma\} = h^*(n)$ → $h(n) \leq h(\gamma) + h^*(n)$, dar $h(\gamma) = 0$ → $h(n) \leq h^*(n)$ → euristică admisibilă.
- Corolar 7.2: O euristică monotonă este admisibilă.

Dominanță - Definiții

- Definiție 7.6: Fie h_1 si h_2 două euristici admisibile.
  - 1. Spunem că h_1 este mai informată decât h_2 dacă $h_2(n) < h_1(n)$ pentru orice nod $n \notin \Gamma$ din graful spațiului de stare explorat.
  - 2. Spunem că h_1 este aproximativ mai bine informată decât h_2 dacă $h_2(n) \leq h_1(n)$ pentru orice nod $n \notin \Gamma$ din graful spațiului de stare explorat.

- Definiție 7.7: Un algoritm A_1^* domină un algoritm A_2^* dacă orice nod expandat de A_1^* este expandat si de A_2^*. (eventual, A_2^* expandează noduri suplimentare față de A_1^*, deci A_1^* poate fi mai rapid ca A_2^*.)

- Definiție 7.8: Un algoritm A_1^* domină aproximativ un algoritm A_2^* dacă orice nod expandat de A_1^* este expandat si de A_2^* cu eventuala excepție a unor noduri care satisfac condiția $h_1(n) = h_2(n) = C - g^*(n)$.

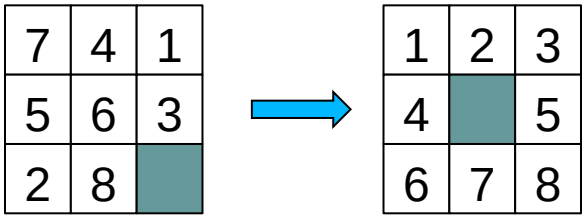
Dominanța - Teoreme

- Teorema 7.11: Dacă o euristică monotonă h_1 este mai informată decât o euristică monotonă h_2, atunci un algoritm A_1^* condus de h_1 domină un algoritm A_2^* condus de h_2.

- Teorema 7.12: Dacă o euristică monotonă h_1 este aproximativ mai bine informată decât o euristică monotonă h_2, atunci un algoritm A_1^* condus de h_1 domină aproximativ un algoritm A_2^* condus de h_2.

Dominanța - Exemplu

- Considerăm jocul 8-pătrățele care trebuie aranjat pornind de la forma inițială prin mutarea locului ‘liber’ astfel încât să ajungem la forma finală:



- Două euristici posibile:
  - h_1 = numărul pătrățelelor a căror poziție curentă diferă de poziția finală;
  - $h_1 = \Sigma_{p \in piese}(\delta_p)$, unde $\delta_p = 0$ dacă poziția curentă coincide cu cea finală și $\delta_p = 1$, altfel
  - h_2 = distanța Manhattan = suma distanțelor pe verticală si orizontală între pozițiile curente ale pătrățelelor și pozițiile lor finale
  - $h_2 = \Sigma_{p \in piese}(dist\_h_p + dist\_v_p)$

Admisibilitate? Monotonie? Dominanță? Care euristică va fi aleasă pentru A*?

Complexitate A*

- Liniară dacă $|h(n) - h^*(n)| \leq \delta$, unde $\delta \geq 0$ este o constantă.
- Subexponențială, dacă $|h(n) - h^*(n)| \leq O(\log(h^*(n)))$.
- Exponențială, altfel, (dar mult mai bună decât a căutărilor neinformate).
- Mai multe explicații găsiți in Giumale 7.4.4