TEHNICI DE CĂUTARE

Căutarea este o traversare *sistematică* a unui *spaţiu de soluţii* posibile ale unei probleme.

Un *spaţiu de căutare* este de obicei un graf (sau, mai exact, un arbore) în care un nod desemnează o soluţie parţială, iar o muchie reprezintă un pas în construirea unei soluţii. Scopul căutării poate fi acela de

- **a găsi *un drum* în graf de la o situaţie iniţială la una finală;**
- **a ajunge într-un nod care reprezintă situaţia finală.**

Programul

- **Programul reprezinta un agent inteligent.**

- **Agenţii cu care vom lucra vor adopta un *scop* şi vor urmări *satisfacerea* lui.**

Rezolvarea problemelor prin intermediul căutării

- **În procesul de rezolvare a problemelor, *formularea scopului*, bazată pe situaţia curentă, reprezintă primul pas.**
- **Vom considera un *scop* ca fiind o mulţime de stări ale universului, şi anume acele stări în care scopul este satisfăcut.**
- ***Acţiunile* pot fi privite ca generând tranziţii între stări ale universului.**
- **Agentul va trebui să afle care acţiuni îl vor conduce la o stare în care scopul este satisfăcut. Înainte de a face asta el trebuie să decidă ce tipuri de acţiuni şi de stări să ia în consideraţie.**
- **Procesul decizional cu privire la acţiunile şi stările ce trebuie luate în consideraţie reprezintă *formularea problemei*. Formularea problemei urmează după formularea scopului.**

- **Un agent care va avea la dispoziţie mai multe opţiuni imediate va decide ce să facă examinând mai întâi diferite *secvenţe de acţiuni* posibile, care conduc la stări de valori necunoscute, urmând ca, în urma acestei examinări, să o aleagă pe cea mai bună. Procesul de examinare a unei astfel de succesiuni de acţiuni se numeşte *căutare*.**

- **Un *algoritm de căutare* primeşte ca *input* o *problemă* şi întoarce ca *output* o *soluţie* sub forma unei succesiuni de acţiuni.**
  Odată cu găsirea unei soluţii, acţiunile recomandate de aceasta pot fi duse la

îndeplinire. Aceasta este *faza de execuţie*.

Prin urmare:

- ✔ **agentul *formulează*, *caută* şi *execută*.**

După formularea unui scop şi a unei probleme de rezolvat, agentul cheamă o procedură de căutare pentru a o rezolva. El foloseşte apoi soluţia pentru a-l ghida în acţiunile sale, executând ceea ce îi recomandă soluţia ca fiind următoarea acţiune de îndeplinit şi apoi înlătură acest pas din succesiunea de acţiuni. Odată ce soluţia a fost executată, agentul va găsi un nou scop.

Probleme şi soluţii corect definite

Probleme cu o singură stare

Elementele de bază ale definirii unei probleme sunt *stările* şi *acţiunile*. Pentru a descrie stările şi acţiunile, din punct de vedere formal, este nevoie de următoarele elemente:

- ***Starea iniţială* în care agentul ştie că se află.**
- **Mulţimea acţiunilor posibile disponibile agentului. Termenul de *operator* este folosit pentru a desemna descrierea unei acţiuni, prin specificarea stării în care se va ajunge ca urmare a îndeplinirii acţiunii respective, atunci când ne aflăm într-o anumită stare. (O formulare alternativă foloseşte o *funcţie succesor S*. Fiind dată o anumită stare *x*, *S(x)* întoarce mulţimea stărilor în care se poate ajunge din *x*, printr-o unică acţiune).**

- ***Spaţiul de stări*** **al unei probleme reprezintă mulţimea tuturor stărilor în care se poate ajunge plecând din starea iniţială, prin intermediul oricărei secvenţe de acţiuni.**
- **Un *drum* în spaţiul de stări este orice secvenţă de acţiuni care conduce de la o stare la alta.**
- ***Testul scop*** **este testul pe care un agent îl poate aplica unei singure descrieri de stare pentru a determina dacă ea este o *stare de tip scop*, adică o stare în care scopul este atins (sau realizat). Uneori există o mulţime explicită de stări scop posibile şi testul efectuat nu face decât să verifice dacă s-a ajuns în una dintre ele. Alteori, scopul este specificat printr-o proprietate abstractă şi nu prin enumerarea unei mulţimi de stări. De exemplu, în şah, scopul este să se ajungă la o stare numită "şah mat", în care regele adversarului poate fi capturat la următoarea mutare, orice ar face adversarul. S-ar putea întâmpla ca o soluţie să fie preferabilă alteia, chiar dacă amândouă ating scopul. Spre exemplu, pot fi preferate drumuri cu mai puţine acţiuni sau cu acţiuni mai puţin costisitoare.**

- ***Funcţia de cost a unui drum*** **este o funcţie care atribuie un cost unui drum. Ea este adeseori notată prin *g*. Vom considera costul unui drum ca fiind suma costurilor acţiunilor individuale care compun drumul.**

Împreună *starea iniţială*, *mulţimea operatorilor*, *testul scop* şi *funcţia de cost a unui drum* definesc o *problemă*.

Probleme cu stări multiple

Pentru definirea unei astfel de probleme trebuie specificate:

- **o *mulţime* de stări iniţiale;**
- **o mulţime de operatori care indică, în cazul fiecărei acţiuni, mulţimea stărilor în care se ajunge plecând de la orice stare dată;**
- **un test scop (la fel ca la problema cu o singură stare);**
- **funcţia de cost a unui drum (la fel ca la problema cu o singură stare).**

✔ **Un *operator* se aplică unei mulţimi de stări prin reunirea rezultatelor aplicării operatorului fiecărei stări din mulţime.**

✔ **Aici un *drum* leagă *mulţimi de stări*, iar o *soluţie* este un drum care conduce la o *mulţime de stări*, dintre care *toate sunt stări scop*.**

Spaţiul de stări este aici înlocuit de *spaţiul mulţimii de stări*.

Un exemplu: Problema misionarilor si a canibalilor

Definiţie formală a problemei:

- ***Stări*: o stare constă dintr-o secvenţă ordonată de trei numere reprezentând numărul de misionari, de canibali şi de bărci, care se află pe malul râului. Starea de pornire (iniţială) este (3,3,1).**
- ***Operatori*: din fiecare stare, posibilii operatori trebuie să ia fie un misionar, fie un canibal, fie doi misionari, fie doi canibali, fie câte unul din fiecare şi să îi transporte cu barca. Prin urmare, există *cel mult cinci operatori*, deşi majorităţii stărilor le corespund mai puţini operatori,**

intrucât trebuie evitate stările interzise. (*Observaţie*: Dacă am fi ales să distingem între indivizi, în loc de cinci operatori ar fi existat 27).

- ***Testul scop*: să se ajungă în starea (0,0,0).**
- ***Costul drumului*: este dat de numărul de traversări.**

Căutarea soluţiilor şi generarea secvenţelor de acţiuni

- **Rezolvarea unei probleme începe cu *starea iniţială*.**
- **Primul pas este acela de a testa dacă starea iniţială este o *stare scop*.**
- **Dacă nu, se iau în consideraţie şi alte stări. Acest lucru se realizează aplicând *operatorii* asupra stării curente şi, în consecinţă, generând o mulţime de stări. Procesul poartă denumirea de *extinderea stării*.**
- **Atunci când se generează mai multe posibilităţi, trebuie făcută o alegere relativ la cea care va fi luată în**

consideraţie în continuare, aceasta fiind esenţa căutării.

Alegerea referitoare la care dintre stări trebuie extinsă prima este determinată de strategia de căutare.

Procesul de căutare construieşte un *arbore de căutare*, a cărui rădăcină este un *nod de căutare* corespunzând stării iniţiale. La fiecare pas, algoritmul de căutare alege un nod-frunză pentru a-l extinde.

Observatie:

Este important să facem *distincţia între spaţiul stărilor şi arborele de căutare*. Spre exemplu, într-o problemă de căutare a unui drum pe o hartă, pot exista doar 20 de stări în spaţiul stărilor, câte una pentru fiecare oraş. Dar există un număr infinit de drumuri în acest spaţiu de stări. Prin urmare, arborele de căutare are un număr infinit de noduri. Evident, un bun algoritm de căutare trebuie să evite urmarea unor asemenea drumuri.

Observatie:

Este importantă distincţia între noduri şi stări:

- **Un nod este o structură de date folosită pentru a reprezenta arborele de căutare corespunzător unei anumite realizări a unei probleme, generată de un anumit algoritm.**
- **O stare reprezintă o configuraţie a lumii înconjurătoare.**

De aceea, nodurile au adâncimi şi părinţi, iar stările nu le au. Mai mult, este posibil ca două noduri diferite să conţină aceeaşi stare, dacă

acea stare este generată prin intermediul a două secvenţe de acţiuni diferite.

Reprezentarea nodurilor in program

Există numeroase moduri de a reprezenta nodurile. În general, se consideră că un nod este o structură de date cu cinci componente:

- starea din spaţiul de stări căreia îi corespunde nodul;
- nodul din arborele de căutare care a generat acest nod (nodul părinte);
- operatorul care a fost aplicat pentru a se genera nodul;
- numărul de noduri aflate pe drumul de la rădăcină la acest nod (*adâncimea* nodului);
- costul drumului de la starea iniţială la acest nod.

Reprezentarea colecţiei de noduri care aşteaptă pentru a fi extinse

Această colecţie de noduri poartă denumirea de *frontieră*. Cea mai simplă reprezentare ar fi aceea a unei mulţimi de noduri, iar strategia de căutare ar fi o funcţie care selectează, din această mulţime, următorul nod ce trebuie extins. Deşi din punct de vedere conceptual această cale este una directă, din punct de vedere computaţional ea poate fi foarte scumpă, pentru că funcţia strategie ar trebui să se "uite" la fiecare element al mulţimii pentru a-l alege pe cel mai bun. De aceea, *vom presupune că această colecţie de noduri este implementată ca o coadă.*

Evaluarea strategiilor de căutare

Strategiile de căutare se evaluează conform următoarelor patru criterii:

Completitudine: dacă, atunci când o soluţie există, strategia dată garantează găsirea acesteia;

Complexitate a timpului: durata de timp pentru găsirea unei soluţii;

Complexitate a spaţiului: necesităţile de memorie pentru efectuarea căutării;

Optimalitate: atunci când există mai multe soluţii, strategia dată să o găsească pe cea mai de calitate dintre ele.

Căutarea neinformată

Termenul de *căutare neinformată* desemnează faptul că o strategie de acest tip nu deţine nici o informaţie despre numărul de paşi sau despre costul drumului de la starea curentă la scop. Tot ceea ce se poate face este să se distingă o stare-scop de o stare care nu este scop. Căutarea neinformată se mai numeşte şi *căutarea oarbă*.

Căutarea informată

Să considerăm, de pildă, problema găsirii unui drum de la Arad la Bucureşti, având în faţă o hartă. De la starea iniţială, Arad, există trei acţiuni care conduc la trei noi stări: Sibiu, Timişoara şi Zerind. O căutare neinformată nu are nici o preferinţă între cele trei variante. Un agent mai inteligent va observa însă că scopul, Bucureşti, se află la sud-est de Arad şi că numai *Sibiu* este în această direcţie, care reprezintă, probabil, cea mai bună alegere. Strategiile care folosesc asemenea consideraţii se numesc *strategii de căutare informată* sau *strategii de căutare euristică*.

Căutarea informată

Căutarea informată **se mai numeşte şi** ***căutare euristică*****.**

Euristica este o metodă de studiu şi de cercetare bazată pe descoperirea de fapte noi. În acest tip de căutare vom folosi informaţia despre spaţiul de stări. Se folosesc cunoştinţe specifice problemei şi se rezolvă probleme de optim.

Căutarea de tip best-first

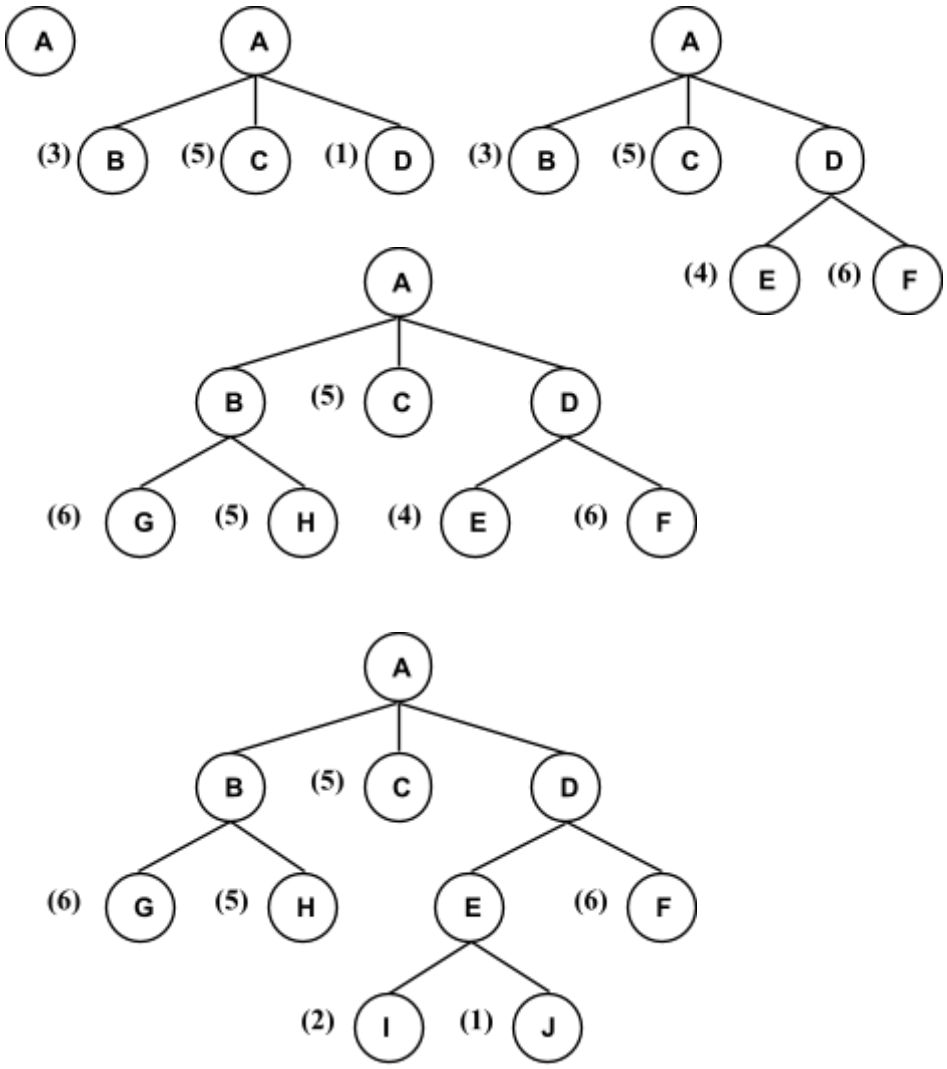
- **Procesul de căutare nu se desfăşoară în mod uniform plecând de la nodul iniţial. El înaintează în mod preferenţial de-a lungul unor noduri pe care informaţia euristică, specifică problemei, le indică ca aflându-se pe drumul cel mai bun către un scop. Un asemenea proces de căutare se numeşte *căutare euristică* sau *căutare de tip best-first*.**

- **Principiile pe care se bazează căutarea de tip best-first sunt următoarele:**

  1. Se presupune existenţa unei funcţii euristice de evaluare, $\hat{f}$, cu rolul de a ne ajuta să decidem care nod ar trebui extins la pasul următor. Se va adopta *convenţia* că valori mici ale lui $\hat{f}$ indică nodurile cele mai bune. Această funcţie se bazează pe informaţie specifică domeniului pentru care s-a formulat problema. Este o funcţie de descriere a stărilor, cu valori reale.

2. **Se extinde nodul cu cea mai mică valoare a lui $\hat{f}(n)$.** ***În cele ce urmează, se va presupune că extinderea unui nod va produce toţi succesorii acelui nod.***

Figura urmatoare ilustrează începutul unei căutări de tip best-first:



Aici există iniţial un singur nod, A, astfel încât acesta va fi extins.

Pentru a nu fi induşi în eroare de o euristică extrem de optimistă, este necesar să înclinăm căutarea în favoarea posibilităţii de a ne întoarce înapoi, cu scopul de a explora drumuri găsite mai devreme. De aceea, vom adăuga lui $\hat{f}$ ***un factor de adâncime*****,**

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n),$$

unde:

- $\hat{g}(n)$ **este** ***o estimaţie a adâncimii lui n în graf*****, adică reprezintă lungimea celui mai scurt drum de la nodul de start la** ***n*****;**
- $\hat{h}(n)$ **este** ***o evaluare euristică a nodului n*****.**

Prezentăm un algoritm de căutare generală bazat pe grafuri. Algoritmul include versiuni ale căutării de tip best-first ca reprezentând cazuri particulare.

Algoritm de căutare general bazat pe grafuri

Acest algoritm, pe care îl vom numi GraphSearch, este unul general, care permite orice tip de ordonare preferată de utilizator - euristică sau neinformată. Iată o *primă variantă* a definiţiei sale:

GraphSearch

1. **Creează un arbore de căutare, T_r, care constă numai din nodul de start n_0. Plasează pe n_0 într-o listă ordonată numită OPEN.**
2. **Creează o listă numită CLOSED, care iniţial este vidă.**
3. **Dacă lista OPEN este vidă, EXIT cu eşec.**
4. **Selectează primul nod din OPEN, înlătură-l din lista OPEN şi include-l în lista CLOSED. Numeşte acest nod *n*.**
5. **Dacă *n* este un nod scop, algoritmul se încheie cu succes, iar soluţia este cea obţinută prin urmarea în sens invers a unui drum de-a lungul arcelor din arborele T_r, de la *n* la n_0. (Arcele sunt create la pasul 6).**
6. **Extinde nodul *n*, generând o mulţime, *M*, de succesori. Include *M* ca succesori ai lui *n* în T_r, prin crearea de arce de la *n* la fiecare membru al mulţimii *M*.**
7. **Reordonează lista OPEN, fie în concordanţă cu un plan arbitrar, fie în mod euristic.**
8. **Mergi la pasul 3.**

- **<u>Observaţie</u>: Acest algoritm poate fi folosit pentru a efectua căutări de tip best-first, breadth-first sau depth-first. În cazul algoritmului *breadth-first* noile noduri sunt puse la sfârşitul listei OPEN (organizată ca o coadă), iar nodurile nu sunt reordonate. În cazul căutării de tip *depth-first* noile noduri sunt plasate la începutul listei OPEN (organizată ca o stivă). În cazul căutării de tip *best-first,* numită şi căutare euristică, lista OPEN este reordonată în funcţie de meritele euristice ale nodurilor.**