UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

POLITEHNICA DIN BUCUREȘTI
UNIVERSITATEA
1818

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

Proiectarea Algoritmilor

27. Căutări informate - comparare căutări neinformate

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

Cuprins

- Explorarea spațiului stărilor problemei
- Explorare informată irevocabilă
- Explorări tentative informate
  - Explorare lacomă
  - Explorare tentativă completă
  - Explorare A*

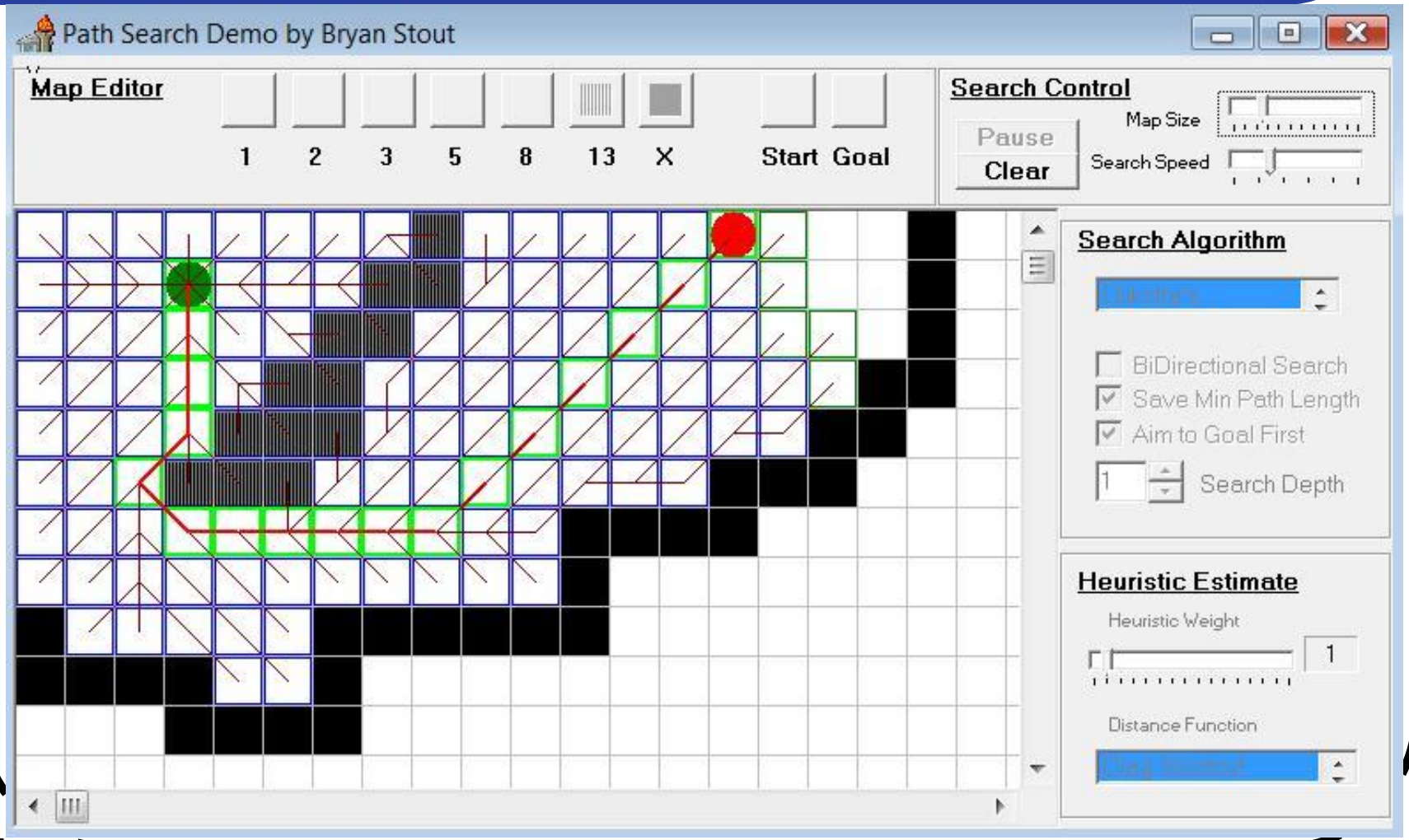
Probleme cu căutările neinformate

- Modelul unor probleme este prea complicat → variantele de rezolvare se bazează pe explorarea spațiului stărilor.

- Probleme:
  - Deseori se calculează prea mult (ex: drumul optim între 2 puncte folosind Dijkstra) - ex: Dijkstra.
  - În cazul grafurilor infinite sau nedescoperite încă, algoritmii clasici fie sunt ineficienți, fie nu garantează găsirea soluției.

- Soluție:
  - Rezolvarea să nu se mai bazeze numai pe calculele exacte ci și pe experiența anterioară (eurisitici) → direcționarea căutării.

Exemplu Dijkstra



Explorarea spațiului stărilor problemei

Spațiul stărilor unei probleme

- Definiție: Stare a problemei = abstractizare a unei configurații valide a universului problemei, configurație ce determină univoc comportarea locală a fenomenului descris de problemă.

- Definiție: Spațiul stărilor = graf în care nodurile corespund stărilor problemei, iar arcele desemnează tranzițiile valide intre stări.
  - Caracteristică importantă: nu este cunoscut apriori, ci este descoperit pe măsura explorării!
  - Descriere
    - Nodul de start (starea inițiala);
    - Funcție de expandare a nodurilor (produce lista nodurilor asociate stărilor valide în care se poate ajunge din starea curentă);
    - Predicat de testare dacă un nod corespunde unei stări soluție.

Obiectivele navigării prin spațiul stărilor

- Cartografierea sistematică a spațiului stărilor.
- Asamblarea soluțiilor parțiale care în final conduc la soluția finală. Această soluție finală poate fi:
  - Identificarea stărilor soluție (poziționarea a n regine pe tabla de șah fără să se atace);
  - Drumul străbătut de la starea inițiala spre o stare soluție (acoperirea tablei de șah cu un cal);
  - Strategia de rezolvare = arbore multicăi în care rădăcina este starea inițială, iar frunzele sunt stări soluție. În acest arbore, unele noduri corespund unor evenimente neprevăzute care influențează calea de urmat în rezolvare (identificarea monedei false dintr-un grup de 3 monede).

Căutări informate/neinformate; Algoritmi tentativi/irevocabili

- Definiție: Dacă explorarea este ‘la întâmplare’ → algoritm neinformat.
- Definiție: Dacă explorarea se bazează pe informația acumulată în cursul explorării, informație prelucrată euristic (costuri)→ algoritm informat.
- Definiție: Dacă algoritmul de explorare are posibilitatea să abandoneze calea curentă de rezolvare și să revină la o cale anterioară → algoritmi tentativi.
- Definiție: Altfel (algoritmul avansează pe o singură direcție) → algoritmi irevocabili.

Căutări informate vs neinformate

- Căutările informate beneficiază de informații suplimentare pe care le colectează și le utilizează în încercarea de a ghici direcția în care trebuie explorat spațiul stărilor pentru a găsi soluția.
- Aceste informații sunt stocate:
  - În nodurile din spațiul stărilor:
    - Starea problemei reprezentată de nod;
    - Părintele nodului curent;
    - Copii nodului curent (obținuți prin expandarea acestuia);
    - Costul asociat nodului curent care estimează calitatea nodului f(n);
    - Adâncimea de explorare.
  - În structuri auxiliare pentru diferențierea nodurilor în raport cu gradul de prelucrare:
    - Expandat (închis) – toți succesorii nodului sunt cunoscuți;
    - Explorat (deschis) – nodul e cunoscut, dar succesorii săi nu;
    - Neexplorat – nodul nu e cunoscut.

Listele CLOSED și OPEN

- OPEN = mulțimea (lista) nodurilor explorate (frontiera între zona cunoscută și cea necunoscută).
- CLOSED = mulțimea (lista) nodurilor expandate (regiunea cunoscută în totalitate).
- Explorarea zonelor necunoscute se face prin alegerea și expandarea unui nod din OPEN. După expandare, nodul respectiv e trecut în CLOSED.
- Majoritatea algoritmilor tentativi folosesc lista OPEN, dar doar o parte folosesc lista CLOSED.

Completitudine si optimalitate

- Definiție: Algoritm complet = algoritm de explorare care garantează descoperirea unei soluții, dacă problema acceptă soluție.
  - Algoritmii irevocabili sunt mai rapizi și consumă mai puține resurse decât cei tentativi, dar nu sunt compleți pentru că pierd informație.

- Definiție: Algoritm optimal = algoritm de explorare care descoperă soluția optimă a problemei.

Algoritm generic de explorare

- Explorare(StInit, test_sol)
  - OPEN = {constr_nod(StInit)}; // starea inițiala
  - **Cât timp** (OPEN ≠ Ø)
    // mai am noduri de prelucrat
    - nod = selecție_nod(OPEN); // aleg un nod
    - **Dacă** (test_sol(nod)) **Întoarce** nod;
      // am găsit o soluție
    - OPEN = OPEN \ {nod} U expandare{nod};
      // extind căutarea
  - **Întoarce** insucces; // nu s-a găsit nicio soluție

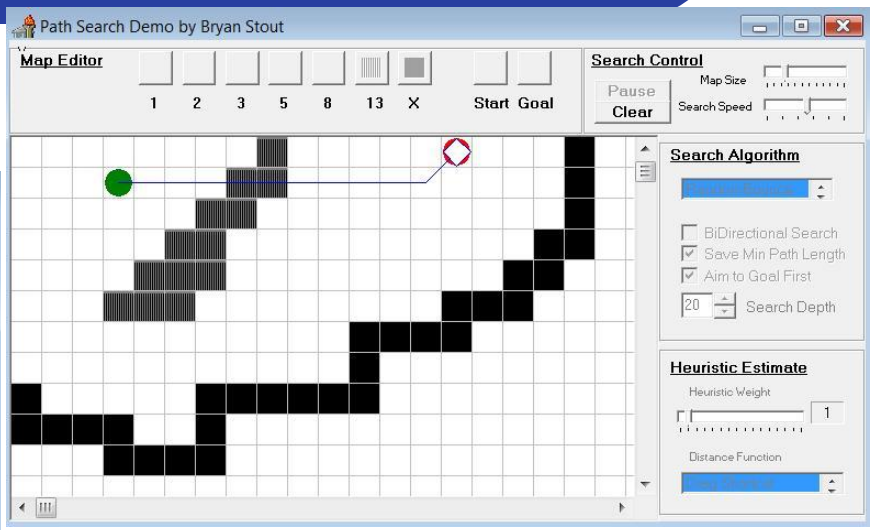
Discuție pe baza algoritmului

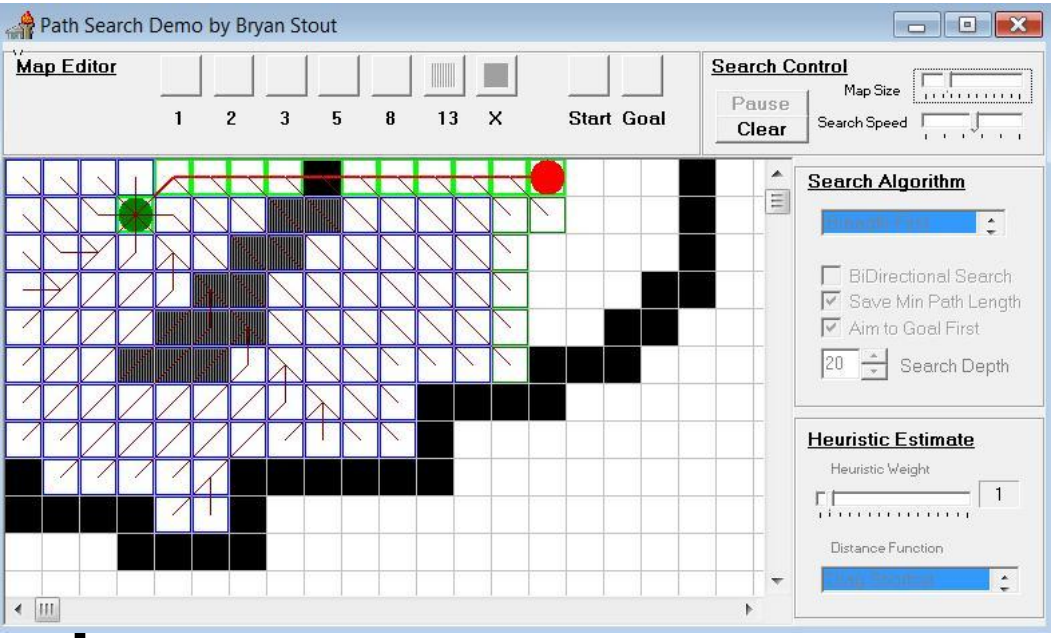
- Dacă **selecție_nod** se realizează independent de costul nodurilor din graful stărilor → căutare neinformată:
  - Dacă e de tip “random” → algoritm aleator - ex: RandomBounce
  - Dacă e de tip “primul venit, primul servit” → OPEN e coadă → BFS – ex: Breadth-first
  - Dacă e de tip “ultimul venit, primul servit” → OPEN e stivă → DFS – ex: Depth-first limitat / IDDFS

- Dacă **selecție_nod** se bazează pe un cost exact sau estimat (euristic) al stărilor problemei → căutare informată:
  - Estimarea costului și folosirea sa în procesul de selecție → esențiale pentru completitudinea, optimalitatea și complexitatea algoritmilor de explorare!

Exemplu de căutări neinformate

RandomBounce →





BFS ↑

IDDFS →