Tema 1 – Strategii de căutare

Deadline: 7.11.2021, 23:59

Mihai Nan

Inteligență artificială (Anul universitar 2021-2022)

1 Obiective

Scopul acestei teme este rezolvarea unei probleme în două variante:

- modelată ca o problemă de căutare în spațiul stărilor, prin implementarea unor strategii de căutare;
- modelată ca o problemă de satisfacere a constrângerilor.

Aspectele urmărite sunt:

- alegerea unei metode de reprezentare a datelor problemei;
- găsirea unor funcții euristice pentru evaluarea stărilor problemei;
- analizarea particularităților diverselor strategii de căutare;
- abstractizarea procesului de rezolvare;
- realizarea unei analize comparative între proprietățile strategiilor implementate.

2 Problema propusă

2.1 Enunțul problemei

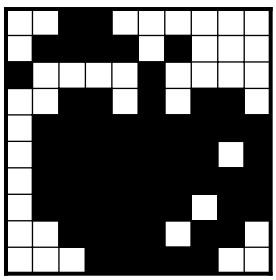
Nonogramele sunt puzzle-uri logice care în urma rezolvării dezvăluie diferite imagini. Soluția este dată de către colorarea sau nu a unor celule dintr-o grilă pe baza interpretării logice a numerelor prezente la începutul fiecărui rând și coloane.

Pentru a rezolva un astfel de puzzle, pornim de la următoarele:

Definiția problemei

Se dă o grilă formată din pătrate care trebuie să fie înnegrite sau lăsate libere. Lângă fiecare linie se află lungimile șirurilor de pătrate negre de pe linia respectivă. Deasupra fiecărei coloane se află lungimile șirurilor de pătrate negre de pe acea coloană.

	1	1 4	2 6	2 7	1 6	8	1 4 1	4 2	2 3	4
2										
4 1										
1 1										
2 1 2										
9										
7 1										
9										
6 2										
4 2										
5										



2.2 Date de intrare

Datele de intrare ale problemei vor fi furnizate într-un fișier `json`, având următoarele câmpuri:

- `name` – numele testului;
- `height` – înălțimea grilei (ca număr de celule);
- `width` – lățimea grilei (ca număr de celule);
- `rows` – vom avea o listă ce conține ca elemente liste cu valorile numerice corespunzătoarea fiecărei linii;
- `columns` – vom avea o listă ce conține ca elemente liste cu valorile numerice corespunzătoarea fiecărei coloane.

```
{
    "name": "test1",
    "height": 10,
    "width": 10,
    "rows": [[2], [4, 1], [1, 1], [2, 1, 2], [9], [7, 1], [9], [6, 2], [4, 2], [5]],
    "columns": [[1], [1, 4], [2, 6], [2, 7], [1, 6], [8], [1, 4, 1], [4, 2], [2, 3], [4]]
}
```

2.3 Date de ieșire

Datele de ieșire vor fi furnizate tot în cadrul unui fișier `json`, având următoarele câmpuri:

- `strategy` – strategia folosită pentru determinarea soluției;
- `nodes generated` – numărul de noduri generate în timpul căutării soluției;
- `nodes expanded` – numărul de noduri expandate complet în timpul căutării soluției;
- `time` – timpul necesar determinării soluției;
- `solution` – soluția problemei reprezentată sub forma unei liste de liste (celule colorate vor avea valoarea 1, iar cele albe vor avea valoarea 0).

3 Cerințe

3.1 Problema de căutare în spațiul stărilor

3.1.1 Cerința 1 – 1 punct

Pentru această cerință, trebuie să citiți informațiile furnizate, să vă alegeți o reprezentare pe care să o folosiți pentru starea problemei propuse și să implementați funcțiile de care aveți nevoie pentru evaluarea unei stări, tranziția între stări, determinarea acțiunilor posibile care se pot aplica într-o stare și tot ceea ce mai considerați că ar fi necesar pentru reprezentarea și prelucrarea datelor problemei.

3.1.2 Cerința 2 – 3 puncte

Implementați următoarele strategii de căutare neinformată: **Breadth First Search**, **Depth First Search** și **Iterative deepening search**. Aplicați aceste strategii pentru rezolvarea problemei date.

3.1.3 Cerința 3 – 2 puncte

Implementați strategia de căutare informată **A* Search** împreună cu două funcții euristice adecvate. Aplicați această strategie pentru rezolvarea problemei date.

3.2 Problema satisfacerii restricțiilor

Probleme de satisfacere a restricțiilor sunt probleme cu o mare aplicabilitate practică, deoarece multe situații din viața reală pot fi modelate folosind acest tip.

3.2.1 Cerința 4 – 3 puncte

Modelați puzzle-ul logic ca pe o problemă de satisfacere a restricțiilor și determinați o soluție pentru această problemă folosind algoritmul **Maintaining Arc Consistency (MAC)** [1]. Problemele de acest tip se pot rezolva folosind tehnica Backtracking, acesta fiind un algoritm complet care găsește soluțiile problemei, dacă acestea există. Deoarece, în general, spațiul de căutare este unul foarte mare, ne vom propune să optimizăm algoritmul Backtracking prin reducerea acestui spațiu de căutare chiar în timpul rulării algoritmului. În cazul algoritmului **MAC**, acest lucru se realizează prin impunerea și menținerea arc-consistenței la fiecare pas al algoritmului (atât timp cât există variabile neinstanțiate).

Înainte de începerea căutării se aplică un algoritm pentru obținerea arc-consistenței verificându-se toate hiperarcele posibile. Apoi, la fiecare nod al arborelui de căutare, se impune arc-consistența pentru acele restricții corespunzătoare variabilei instanțiate în acel nod. Mai precis, dacă variabila x este cea insanțiată la pasul curent, se va impune arc-consistența pentru toate hiperarcele (y, c), unde $c \in \mathbf{C}$ este o constrângere cu $x \in \mathbf{Vars}(c)$, iar $y \in \mathbf{Vars}(c) \setminus \{x\}$.

Pentru reducerea spațiului de căutare (a domeniilor variabilelor) la rularea algoritmului Backtracking se pot impune restricții locale de consistență mai puternice decât arc-consistența, dar, în general, **MAC** reprezintă un compromis bun între costul propagării restricțiilor și dimensiunea spațiului efectiv explorat.

3.3 Analiză comparativă – 1 punct

Realizați o analiză comparativă între strategiile implementate, specificând **succint** pentru fiecare algoritm în parte care sunt avantajele identificate și limitările sale.

Pentru rezolvarea acestei cerințe veți realiza un fișier `README`, în format `pdf`, în care veți prezenta în mod explicit:

- avantajele și dezavantajele identificate pentru fiecare strategie în parte;
- descrierea celor două funcții euristice considerate pentru algoritmul **A* Search**;
- tabel cu rezultatele obținute pentru fiecare strategie în parte pentru testele propuse și analizarea acestora. În acest tabel trebuie incluse informațiile legate de numărul de noduri generate, numărul de noduri expandate complet și timpul necesar determinării soluției.

3.4 Bonus – 2 puncte

Se acordă maximum 2 puncte bonus pentru:

- implementarea strategiei de căutare **Learning Real-Time A*** [2], aplicarea algoritmului pentru problema propusă și compararea acesteia cu strategia **A* Search** (**1 punct**);
- utilizarea euristicilor în rezolvarea problemei satisfacerii restricțiilor: ordonarea variabilelor în vederea atribuirii sau ordonarea valorilor în vederea atribuirii. (**1 punct**).

Referințe

[1] Daniel Sabin and Eugene C Freuder. Contradicting conventional wisdom in constraint satisfaction. In *International Workshop on Principles and Practice of Constraint Programming*, pages 10–20. Springer, 1994.

[2] Richard E Korf. Real-time heuristic search. *Artificial intelligence*, 42(2-3):189–211, 1990. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.1955&rep=rep1&type=pdf`.