Inteligenta Artificiala - Tema 1
Strategii de cautare informata

Teodor-Stefan Dutu

Universitatea Politehnica Bucuresti
Facultatea de Automatica si Calculatoare
Grupa 341C3

Abstract. Se vor compara performantele, ca timp de executie, memorie utilizata si numar stari explorate, ale unor algoritmi de cautare informata in spatiul starilor.

Cuprins

1 Introducere

Scopul meu e sa evaluez performantele unor algoritmi de cautare informata ce folosesc functii euristice prin comparatie cu unul ce nu se bazeaza pe o functie euristica, si anume `DFID`, analizat in Sectiunea 2. Pentru fiecare dintre algoritmii euristici voi compara 3 astfel de functii: distanta euclidiana (indicata in enuntul temei), distanta `Manhattan` (pentru comparatie cu celelalte 2) si cea conceputa pentru rezolvarea cerintei 4, distanta **Weighthattan**, detaliata in Sectiunea 3.1. Fiecare algoritm va fi rulat, cu fiecare euristica, pe fiecare dintre hartile din cele 3 fisiere de input date `input1.txt`, `input2.txt` si `input3.txt`.

Analiza fiecarui algoritm cuprinde grafice care ilustreaza timpii de rulare ai acestuia si memoria consumata in functie de numarul de stari (cu mentiunea ca sunt 93 de noduri (stari) in `input1.txt`, 259 in `input2.txt` si 900 in `input3.txt`) si de euristica. Aceasta analiza este urmata de o serie de harti (una pentru fiecare combinatie de euristica si fisier de intrare), care prezinta pe de o parte costurile gasite de algoritm pentru fiecare stare explorata, cat si drumurile gasite de la starea initiala la cea finala.

2 Cerinta 1 - Depth First Iterative Deepening

Intrucat nu foloseste nicio euristica, acest algoritm va fi folosit drept etalon pentru ceilalti algoritmi implementati. Rezultatele obtinute de acesta sunt exemplificate in Figura 2. Pentru hartile de cost, culorile mai deschise corespund unor costuri mai mari. In figurile continand calea de la pozitia de start la cea finala este figurata cu portocaliu, starea initiala cu rosu, iar cea finala cu alb. In ambele tipuri de imagini, obstacolele sunt figurate cu **negru**.

Se observa ca sunt explorate un numar mare de stari in cazul hartilor din fisierele `input1.txt` si `input3.txt`. Numarul mic de stari explorate in rularea algoritmului `DFID`[1] pe harta din fisierul `input2.txt` este datorat numarului mare si distributiei obstacolelor pe aceasta, obstacole care limiteaza drumul agentului intr-o masura mai mare decat in celelalte harti. Prin urmare, performantele in de timp si memorie inregistrate de algoritmul `DFID` sunt cel prezentate in Figura 1 si in Tabelul 1.

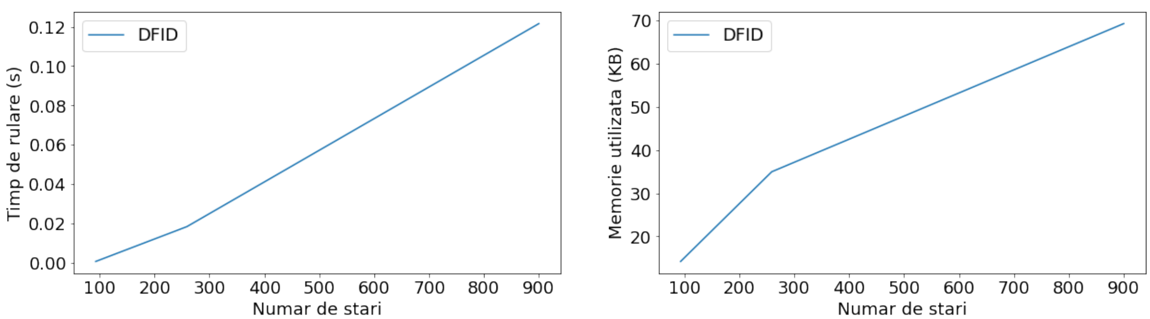
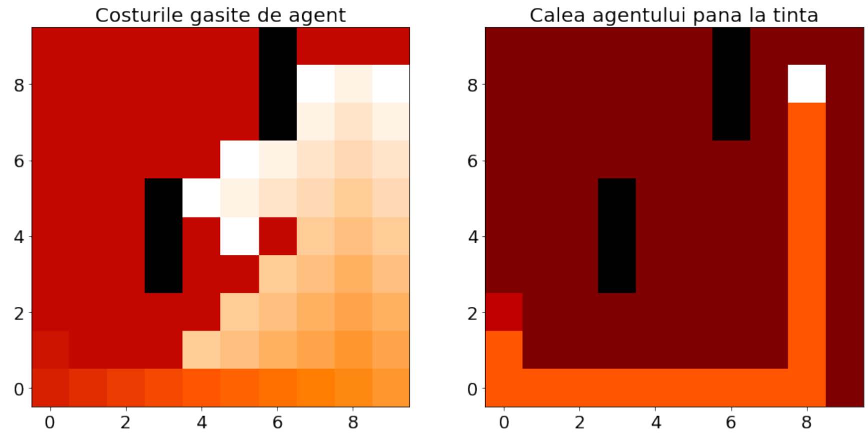


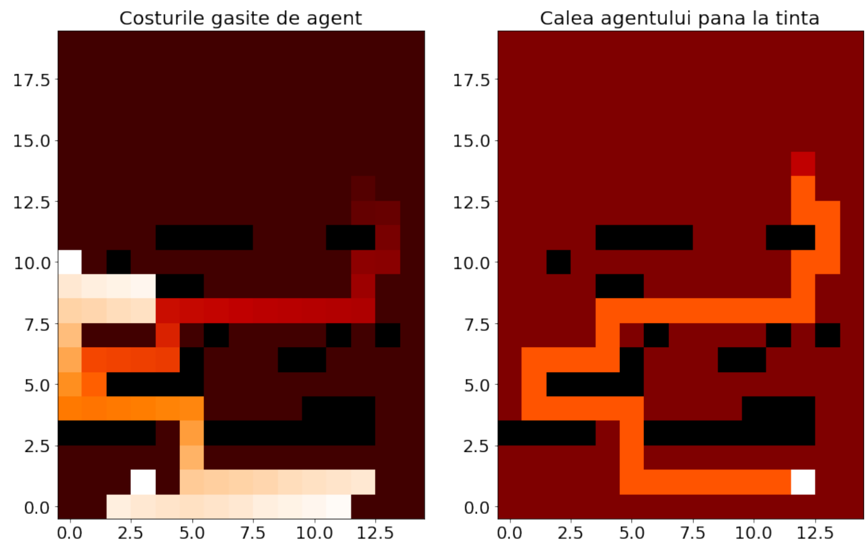
Fig. 1: Performantele algoritmului `DFID`

	input1.txt	**input2.txt**	**input3.txt**
Timp de executie (s)	0.00076	0.01845	0.12158
Memorie utilizata (KB)	14.21	34.96	70.12

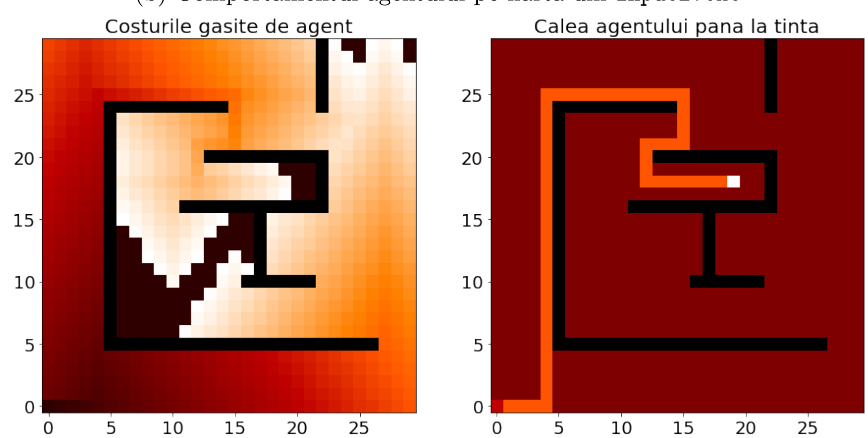
Table 1: Performantele algoritmului `DFID`



(a) Comportamentul agentului pe harta din `input1.txt`



(b) Comportamentul agentului pe harta din `input2.txt`



(c) Comportamentul agentului pe harta din `input3.txt`

Fig. 2: Costurile si caile catre tinta gasite cu algoritmul `DFID`

3 Algoritmi euristici

3.1 Cerinta 4 - Euristica propusa

Se impune sa scriu despre ceerinta 4 aici, inainte sa prezint rezultatele algoritmilor care folosesc functii euristice, deoarece fiecare dintre acesti algoritmi este analizat folosind si euristica propusa in aceasta sectiune.

In primul rand, trebuie spus ca distanta euclidiana, numita in continuare `Euclid` este una slaba, deoarece mediul este discret, nu continuu, iar miscarea agentului se poate face doar paralel cu axele de coordonate, nu si diagonal, iar `Euclid` se preteaza unei libertati mai mare de miscare. O euristica bine cunoscuta potrivita pentru miscarea paralela cu Ox si Oy este distanta `Manhattan`. Dar aceasta poate fi imbunatatita, pastrandu-i complexitatea constanta. Mai exact, euristica `Manhattan` considera ca fiecare tranzitie are costul 1, ceea ce nu este adevarat in cazul hartii din fisierul `input3.txt`. Pentru o aproximare mai fidela a distantei reale din oricare nod la cel tinta, h^*, putem inmulti distanta `Manhattan` cu **costul minim al oricarui arc din graful care modeleaza mediul**. Fie A multimea arcelor si N cea a nodurilor din acest graf. Astfel ia nastere euristica `Weighthattan` (*weight* + *manhattan*), definita formal astfel:

$$weighthattan(s) = manhattan(s) \cdot min_{s_i,s_j \in N}(\{cost(s_i, s_j) \mid arc(s_i, s_j) \in A\}) \tag{1}$$

Fie $m = min_{s_i,s_j \in N}(\{cost(s_i, s_j) \mid arc(s_i, s_j) \Rightarrow$

$$(1) \Leftrightarrow weighthattan(s) = m(|y_f - y_s| + |x_f - x_s|) \tag{2}$$

unde x_s si y_s sunt coordonatele carteziene ale starii s, iar x_f si y_f ale starii finale

Lema 1: Euristica `Weighthattan` este **consistenta**.
Demonstratie: Ca `Weighthattan` sa fie consistenta, trebuie ca $\forall s \in N$:

$$\begin{cases} h(s) \leq cost(s, s') + h(s') \ \forall s' \in succesori(s) \\ SAU \\ h(s) = 0, \text{ daca } s = s_f \end{cases}$$

Ultima ramura este evidenta, din definitia euristicii, deci ne ramane de demonstrat ca

$$h(s) \leq cost(s, s') + h(s') \ \forall s' \in succ(s) \tag{3}$$

$$(2) + (3) \Rightarrow m(|y_f - y_s| + |x_f - x_s|) \leq cost(s, s') + m(|y_f - y'_s| + |x_f - x'_s|) \ \forall s' \in succ(s) \tag{4}$$

unde cu x'_s si y'_s am notat pozitia starii s'.

Din definitia lui $m \Rightarrow m \leq cost(s, s')$, deci pentru a demonstra ecuatia (4) este suficient sa demonstram ca:

$$m(|y_f - y_s| + |x_f - x_s|) \leq m + m(|y_f - y'_s| + |x_f - x'_s|) \ \forall s' \in succ(s) \tag{5}$$

$$(5) \Leftrightarrow |y_f - y_s| + |x_f - x_s| \leq 1 + |y_f - y'_s| + |x_f - x'_s| \ \forall s' \in succ(s) \tag{6}$$

Relatia (6) reprezinta relatia de consistenta pentru euristica `Manhattan`, care este consistenta, deci si `Weighthattan` este consistenta (Q.E.D). Fiind consistenta, euristica `Weighthattan` este si **admisibila**.

Observatie 1: Daca $\exists\ s_i, s_j \in N$ a.i. $arc(s_i, s_j) \in N$ si $cost(s_i, s_j) = 1$, atunci $weighthattan(s) = manhattan(s)\ \forall s \in N$.
Observatie 2: Euristica `Weighthattan` o domina pe cea `Manhattan`.

Din *Observatiile 1, 2* si din *Lema 1*, ne asteptam ca performantele algoritmilor care folosesc euristici sa fie identice pentru euristicile `Manhattan` si `Weighthattan`, la rularea pe hartile din fisierele `input1.txt` si `input2.txt`, unde costurile minime sunt 1, si ne asteptam sa se imbunatateasca la rularea algoritmilor pe harta din fisierul `input3.txt`, unde costul minim este supraunitar, si anume 2. Graficele din sectiunile urmatoare confirma aceste asteptari.

3.2 Cerinta 2 - Iterative Deepening A*

Asa cum am mentionat in Sectiunea 1, euristicile comparate sunt `Euclid`, `Manhattan` si `Weighthattan`.

O prima observatie este ca indiferent de euristica folosita, `IDA*`[1] exploreaza mai putine stari in comparatie cu `DFID`, deoarece acum agentul incearca sa se indrepte mereu spre solutie. Din acest motiv, cantitatea de memorie utilizata de `IDA*` este mai mica decat cea folosita de `DFID`, ceea ce se poate observa in Figurile 4, unde culorile au aceeasi semificatie ca in Figura 2. Aceasta cantitate mica de memorie este datorata folosirii unor euristici care ii permit agentului sa exploreze spatiul doar inspre tinta si nu in toate directiile.

Cu toate acestea, timpii de rulare ai `IDA*` sunt mai mari pentru ca euristica `Euclid` este una slaba, care subestimeaza cu mult costul real $h^*(s)$ al oricarei stari, in afara de cea finala. Performantele se imbunatatesc atunci cand se folosesc distantele Manhattan sau Weighthattan, din motivele explicate in Sectiunea 3.1. Numarul de stari explorate creste si el, dupa cum se poate vedea in Figurile 4 si 5. Pe o harta cu costuri mai mari decat 1, cum este cea din `input3.txt`, se observa imbunatatirea adusa de euristica `Weighthattan` chiar si fata de `Manhattan`. Aceasta imbunatatire nu este vizibila doar la nivelul numarului de stari explorate, ci si la nivelul timpului de rulare a algoritmului, asa cum se poate observa in Figura 3 si in Tabelul 2.

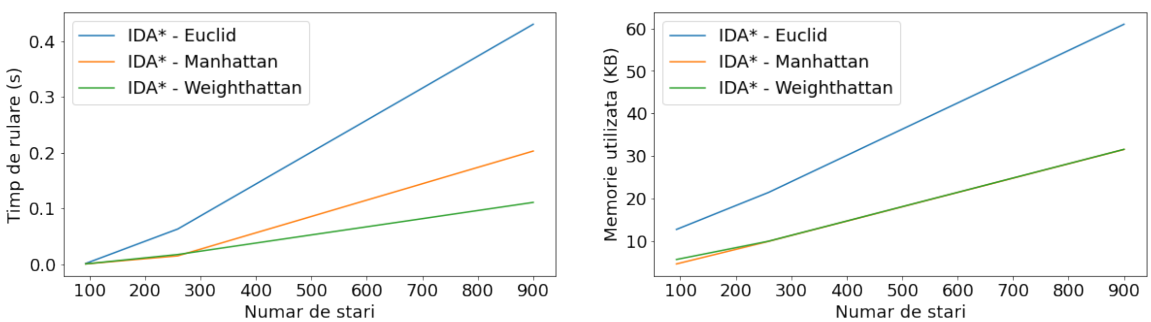
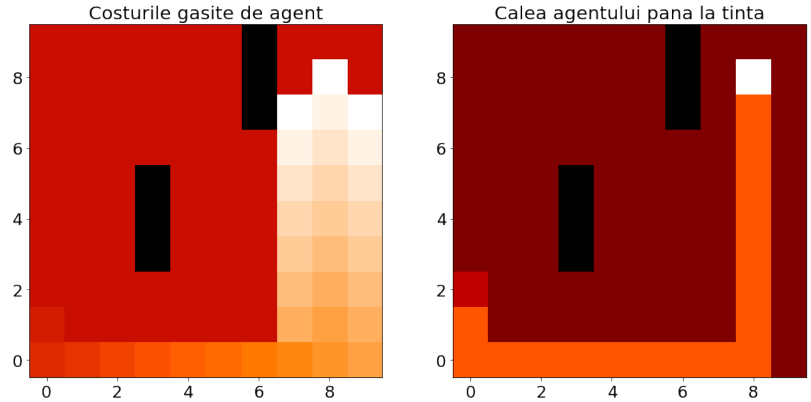


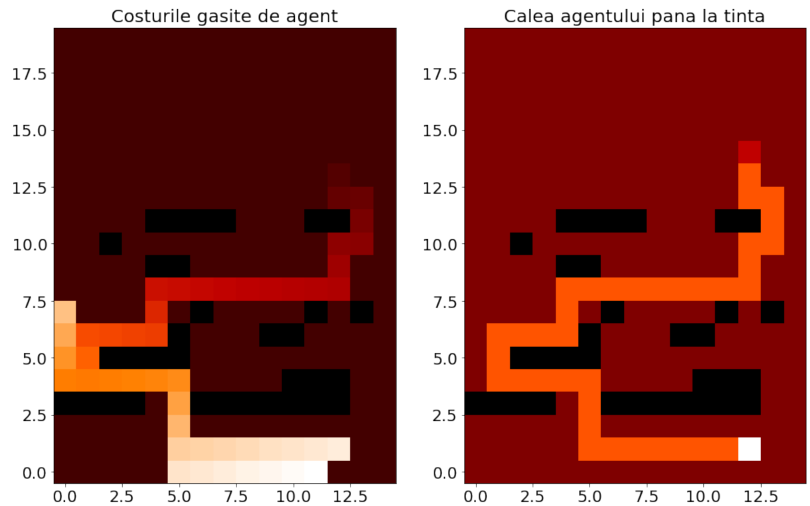
Fig. 3: Performantele algoritmului `IDA*`

	input1.txt	input2.txt	input3.txt
Timp de executie - Euclid (s)	0.00162	0.06274	0.43001
Timp de executie - Manhattan (s)	0.00013	0.01460	0.20262
Timp de executie - Weighthattan (s)	0.00014	0.01705	0.11052
Memorie utilizata - Euclid (KB)	12.75	21.61	59.71
Memorie utilizata - Manhattan (KB)	4.58	9.98	31.45
Memorie utilizata - Weighthattan (KB)	5.10	9.93	32.01

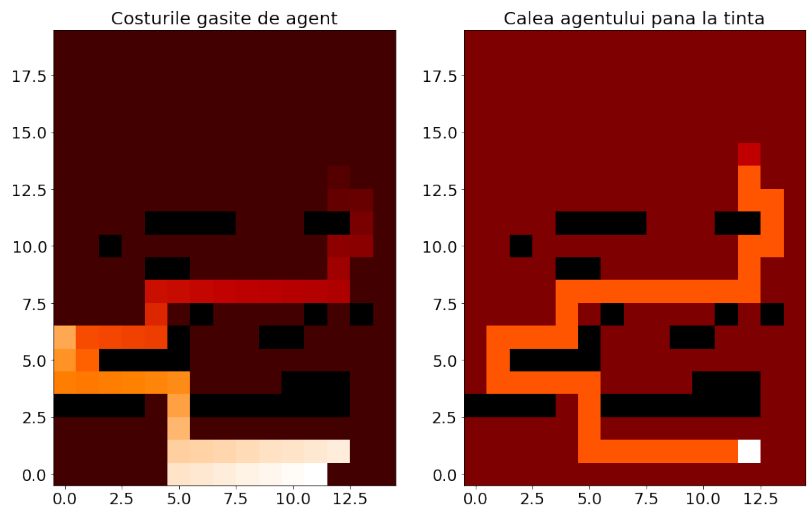
Table 2: Performantele algoritmului `IDA*`



(a) Comportamentul agentului pe harta din `input1.txt` cu orice euristica

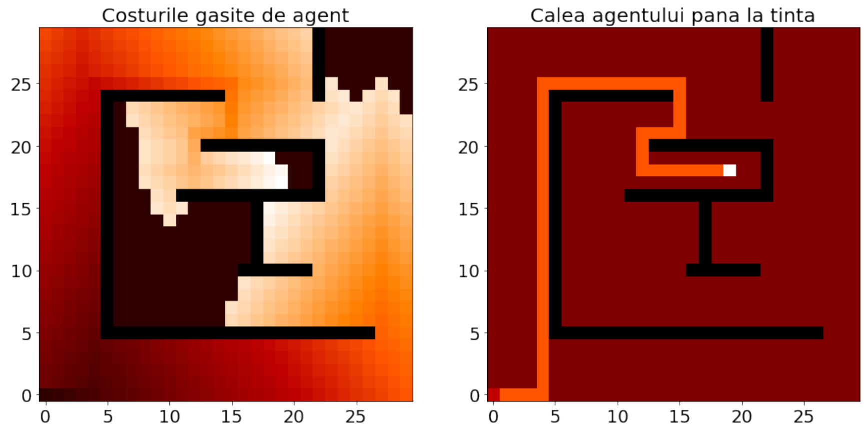


(b) Comportamentul agentului pe harta din `input2.txt` cu euristica `Euclid`

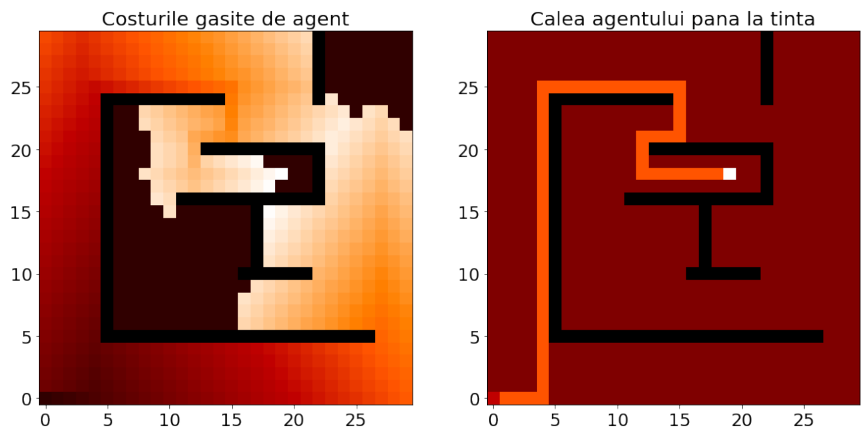


(c) Comportamentul agentului pe harta din `input2.txt` cu euristicile `Manhattan` si `Weighthattan`

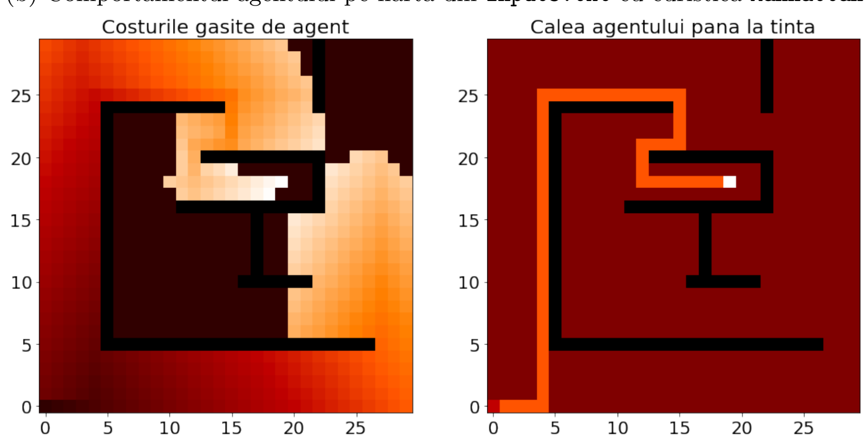
Fig. 4: Costurile si caile catre tinta gasite cu algoritmul `IDA*` (1)



(a) Comportamentul agentului pe harta din `input3.txt` cu euristica `Euclid`



(b) Comportamentul agentului pe harta din `input3.txt` cu euristica `Manhattan`



(c) Comportamentul agentului pe harta din `input3.txt` cu euristica `Weighthattan`

Fig. 5: Costurile si caile catre tinta gasite cu algoritmul `IDA*` (2)

3.3 Cerinta 3 - Learning Real-Time A*

Se compara aceleasi 3 euristici ca la `IDA*`: distanta euclidiana, distanta Manhattan si distanta Weighthattan. Agentul este lasat sa caute cai pana la starea tinta si sa-si actualizeze tabela de costuri estimate (H) pana cand doua rulari succesive intorc aceeasi cale catre tinta, ceea ce indica faptul ca agentul a invatat toate costurile relevante pentru traseul catre tinta. Acest fenomen este reliefat la rularea algoritmului pe harta din `input3.txt`, in Figura 9. Din figura se deduce cantitatea mult mai mare de informatie pe care `LRTA*`[2] ajunge sa o posede fata de ceilalti algoritmi analizati (`LRTA*` calculand costurile pentru toate starile de pe harta), ceea ce implica timpi de rulare cu mult mai mici atunci cand se foloseste euristica `Weighthattan`.

Acest algoritm exploreaza mai putine stari atunci cand foloseste euristicile `Manhattan` si `Weighthattan` chiar si atunci cand ruleaza pe harta din fisierul `input1.txt`. Calea si costurile gasite de `LRTA*` pot fi analizate in Figura 7. Performantele acestui algoritm sunt cel mai vizibile la rularea sa pe fisierul `input3.txt`, ale carei rezultate pot fi analizate in Figura 9. Astfel, folosind euristica `Weighthattan`, timpul de rulare este aproape 6 ori mai mic decat cel inregistrat de algoritmul etalon, `DFID`.

Cum "totul in viata se plateste", informarea foarte buna a algoritmului vine cu pretul folosirii unei cantitati mari de memorie (necesare pentru a retine tabela H si pentru a rula algoritmul pana la invatarea acesteia), care nu se imbunatateste semnificativ folosind euristici mai bune decat `Euclid`, cum ar fi `Manhattan` sau `Weighthattan`. Acest lucru se poate vedea in Figura 6 si in Tabelul 3.

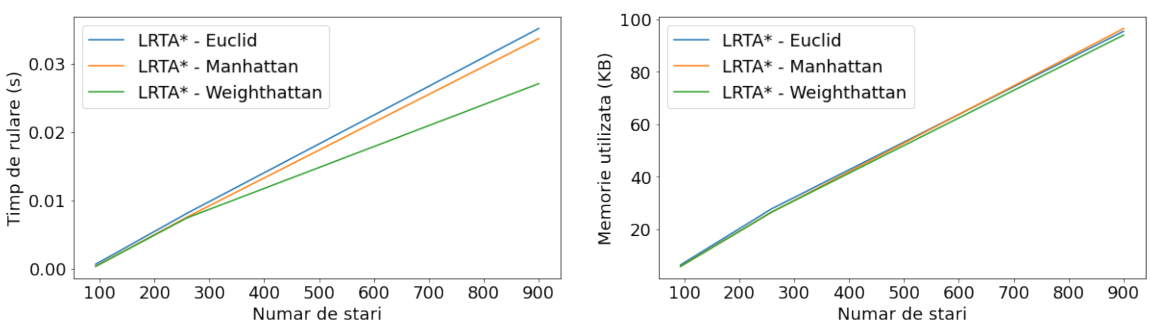
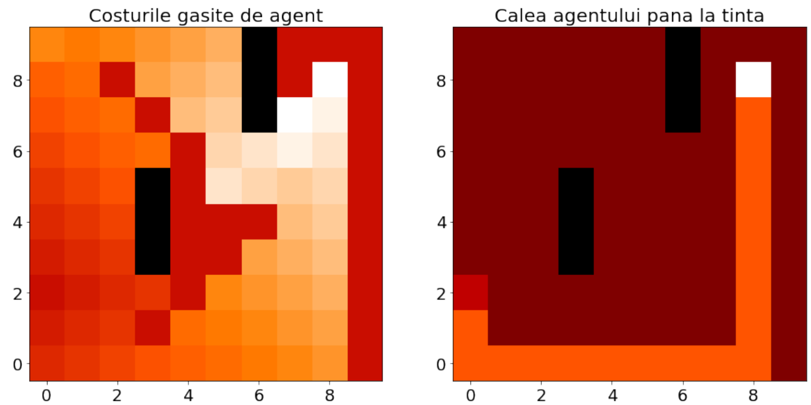


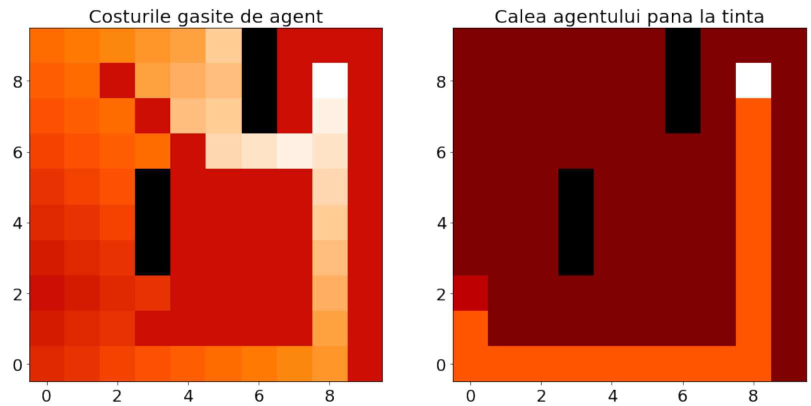
Fig. 6: Performantele algoritmului `LRTA*`

	input1.txt	input2.txt	input3.txt
Timp de executie - Euclid (s)	0.00064	0.00807	0.03509
Timp de executie - Manhattan (s)	0.000332	0.00752	0.03361
Timp de executie - Weighthattan (s)	0.00034	0.00741	0.02703
Memorie utilizata - Euclid (KB)	6.46	27.82	95.42
Memorie utilizata - Manhattan (KB)	5.89	26.58	96.48
Memorie utilizata - Weighthattan (KB)	5.898	26.58	93.98

Table 3: Performantele algoritmului `LRTA*`

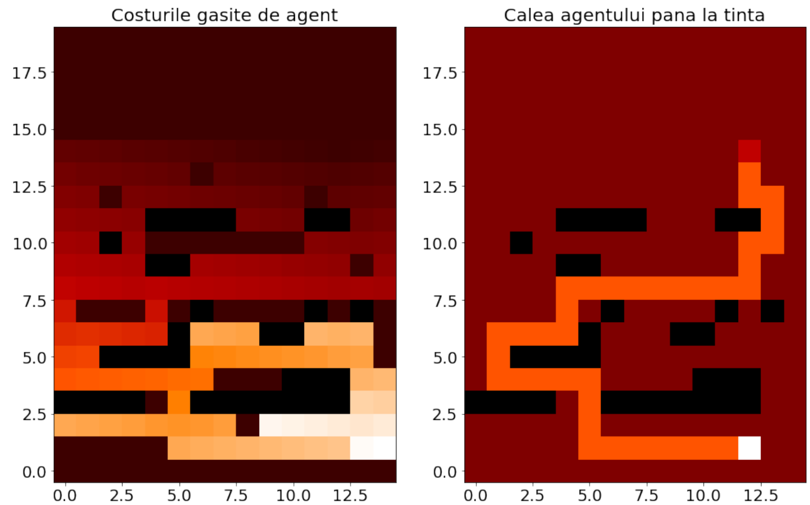


(a) Comportamentul agentului cu euristica `Euclid`

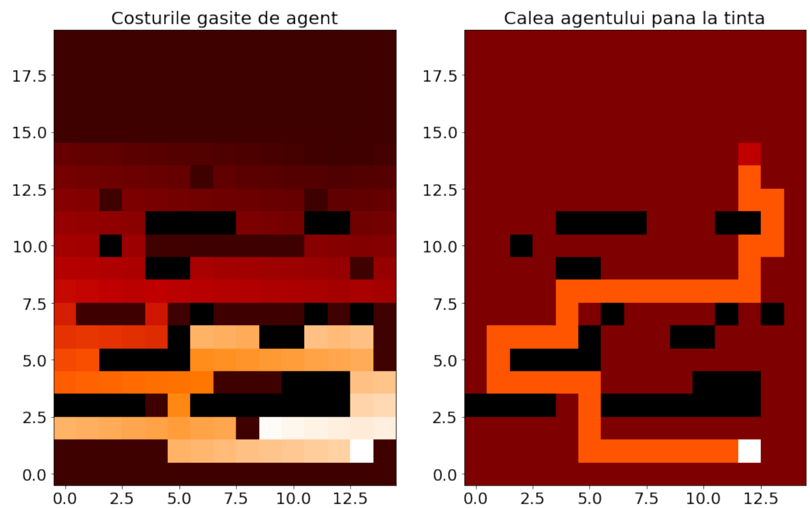


(b) Comportamentul agentului cu euristicile `Manhattan` si `Weighthattan`

Fig. 7: Costurile si caile catre tinta gasite cu algoritmul `LRTA*` (1)

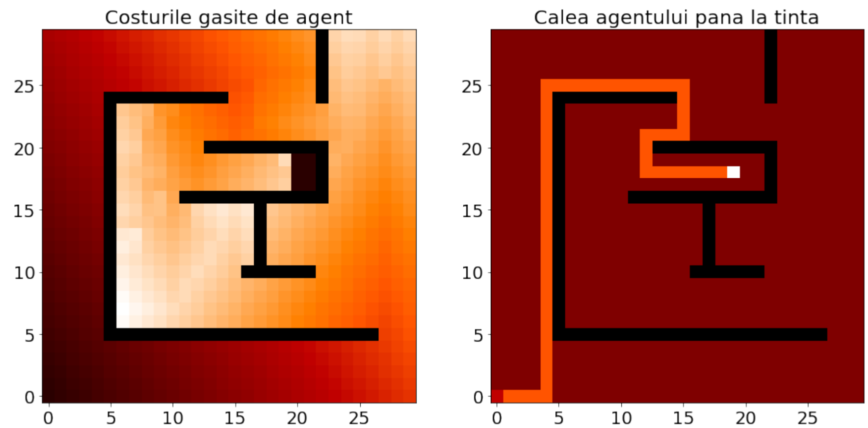


(a) Comportamentul agentului pe harta din `input2.txt` cu euristica `Euclid`

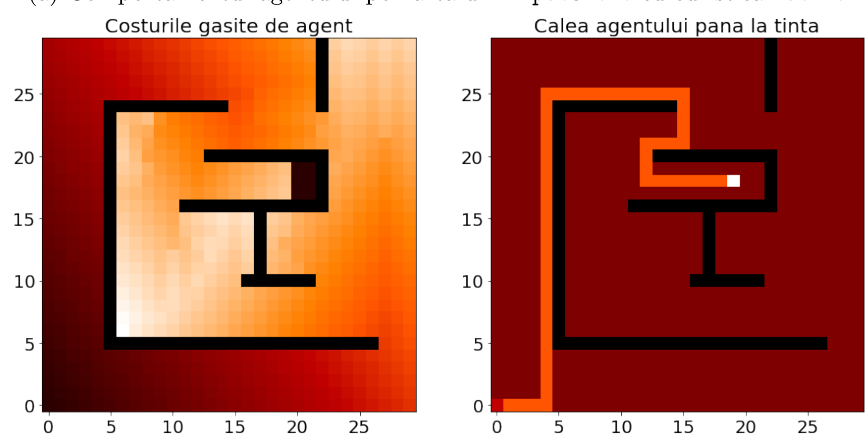


(b) Comportamentul agentului pe harta din `input2.txt` cu euristicile `Manhattan` si `Weighthattan`

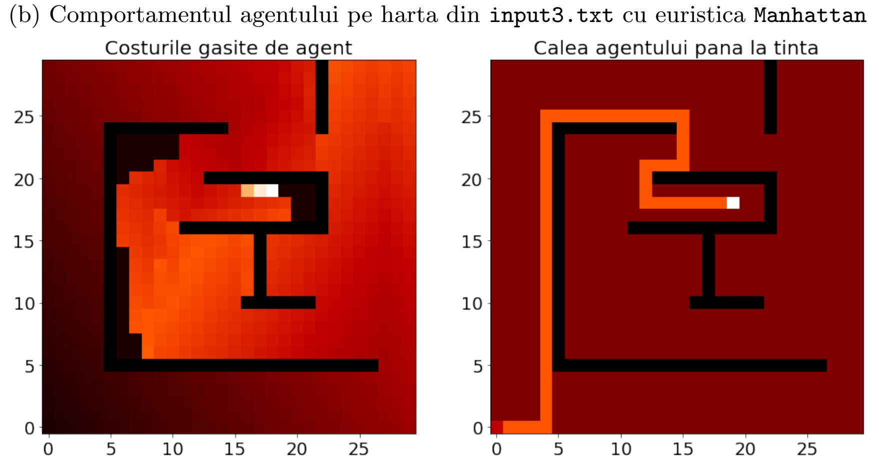
Fig. 8: Costurile si caile catre tinta gasite cu algoritmul `LRTA*` (2)



(a) Comportamentul agentului pe harta din `input3.txt` cu euristica `Euclid`



(b) Comportamentul agentului pe harta din `input3.txt` cu euristica `Manhattan`



(c) Comportamentul agentului pe harta din `input3.txt` cu euristica `Weighthattan`

Fig. 9: Costurile si caile catre tinta gasite cu algoritmul `LRTA*` (3)

3.4 Bonus - Branch and Bound

Algoritmul[3] foloseste un cost-limita, fie el L, care reprezinta cel mai mare cost gasit de agent pana la starea finala pana la un moment dat. Acest cost este initializat cu ∞, si este updatat de fiecare data cand agentul descopera un drum catre starea finala. Scopul acestei limite este de a opri explorarea nodurilor prin care costul estimat (folosind aceleasi euristici de pana acum) pana la tinta este mai mare decat aceasta limita, intrucat, pentru orice euristica admisibila h si orice stare s pentru care costul real de la sursa pana la aceaste este $g(s)$ avem:

$$h(s) \leq h^*(s), \text{ deci daca}$$

$$L \leq g(s) + h(s), \text{ atunci}$$

$$L \leq g(s) + h^*(s) \Leftrightarrow L \leq g(s_f), \text{ unde } s_f \text{ este starea fnala}$$

Acest lucru inseamna ca deja am gasit un drum cel putin la fel de bun pana la s_f, si anume drumul pentru care am obtinut costul L. Din acest motiv, de fiecare data cand algoritmul descopera starea finala, o va face cu un cost mai mic decat cel precedent.

In figuri si in cod, algoritmul a fost prescurtat `BnB`. Ca in cazul celorlalti algoritmi, euristicile pot fi comparate in Figura 10 si in Tabelul 4, iar costurile gasite de acest algoritm, precum si caile de la starea initiala la cea finala, se pot vizualiza in Figurile 11, 12 si 13. Folosind `Weighthattan`, algoritmul acesta ruleaza intr-un timp aproape de 2 ori mai mic decat `DFID`. Pretul este, insa, unul foarte mare, `BnB` utilizand de pana 5 ori mai multa memorie decat `DFID`, deoarece `BnB` nu foloseste niciun mecanism care sa-i limiteze adancimea cautarii, iar pana se gaseste prima data starea finala, se exploreaza un numar mare de stari.

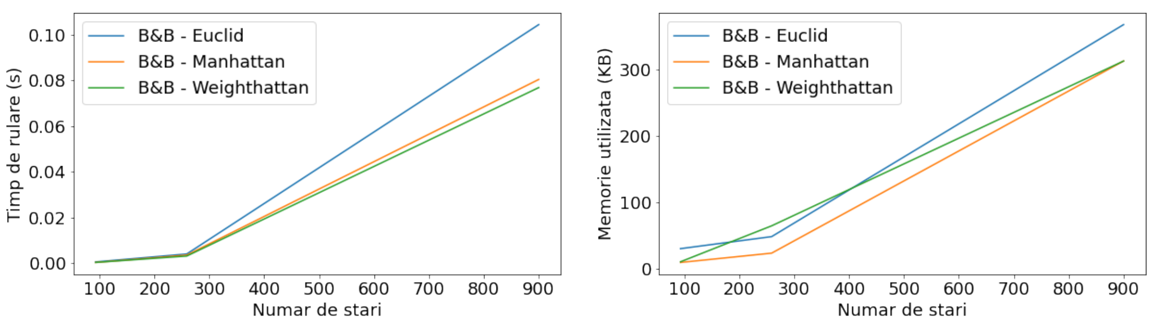
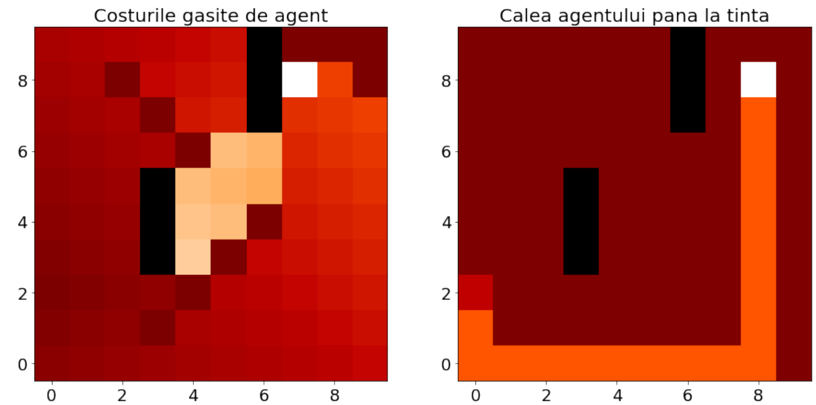


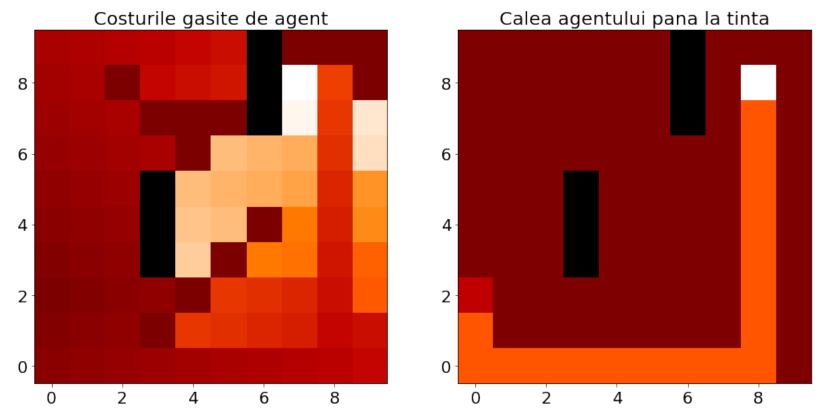
Fig. 10: Performantele algoritmului `BnB`

	input1.txt	input2.txt	input3.txt
Timp de executie - Euclid (s)	0.00048	0.00343	0.10184
Timp de executie - Manhattan (s)	0.00021	0.00261	0.08294
Timp de executie - Weighthattan (s)	0.00023	0.00281	0.07436
Memorie utilizata - Euclid (KB)	26.35	61.42	395.75KB
Memorie utilizata - Manhattan (KB)	9.14	35.83	339.53
Memorie utilizata - Weighthattan (KB)	8.15	73.89	339.53

Table 4: Performantele algoritmului `BnB`

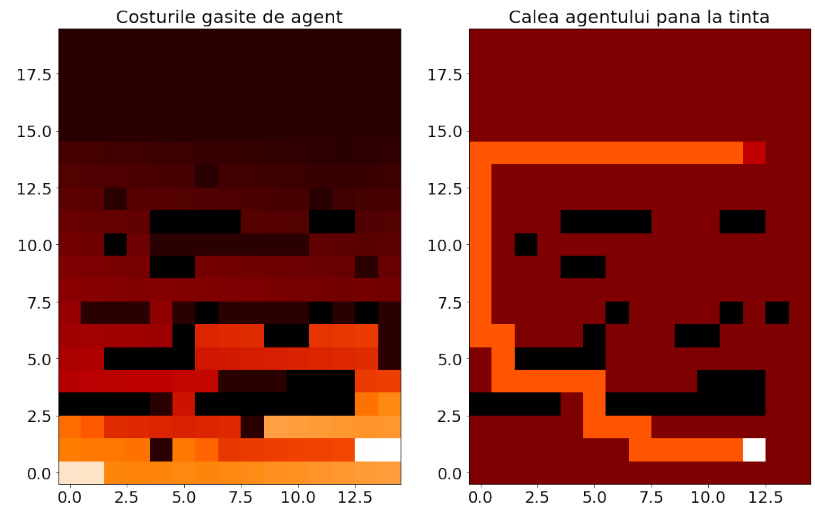


(a) Comportamentul agentului cu euristica `Euclid`

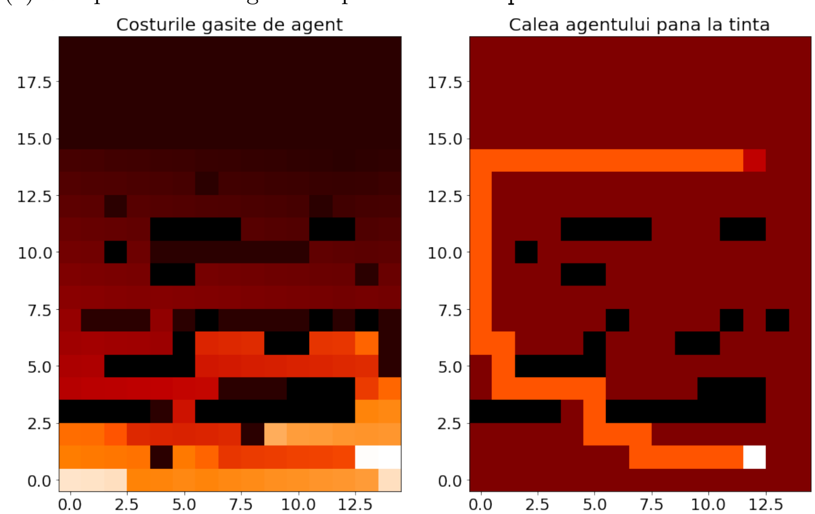


(b) Comportamentul agentului cu euristicile `Manhattan` si `Weighthattan`

Fig. 11: Costurile si caile catre tinta gasite cu algoritmul `BnB` (1)

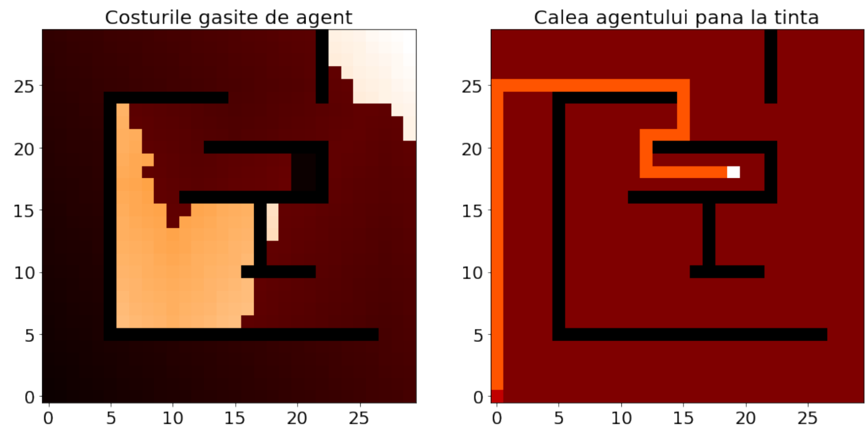


(a) Comportamentul agentului pe harta din `input2.txt` cu euristica `Euclid`

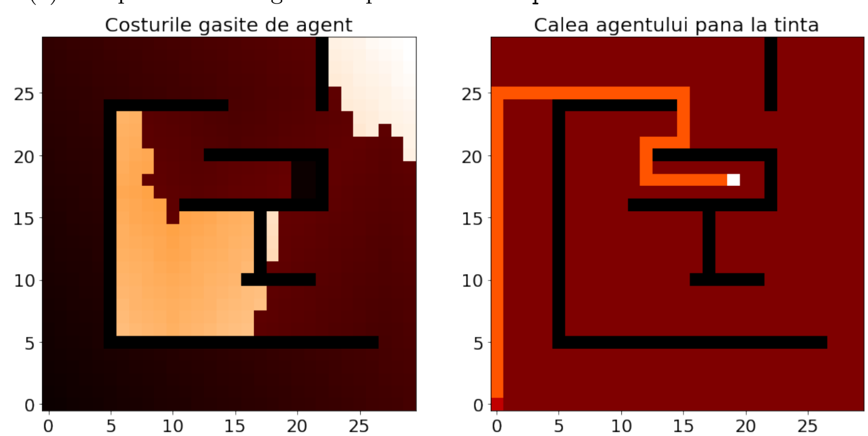


(b) Comportamentul agentului pe harta din `input2.txt` cu euristicile `Manhattan` si `Weighthattan`

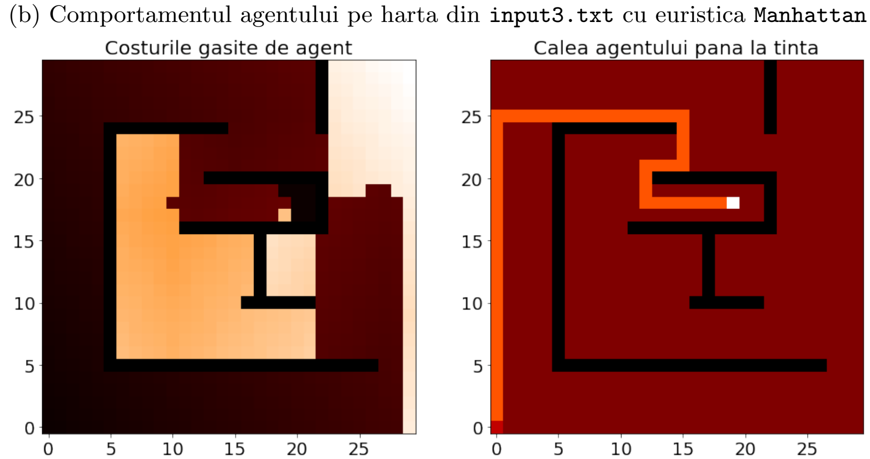
Fig. 12: Costurile si caile catre tinta gasite cu algoritmul `BnB` (2)



(a) Comportamentul agentului pe harta din `input3.txt` cu euristica `Euclid`



(b) Comportamentul agentului pe harta din `input3.txt` cu euristica `Manhattan`



(c) Comportamentul agentului pe harta din `input3.txt` cu euristica `Weighthattan`

Fig. 13: Costurile si caile catre tinta gasite cu algoritmul `BnB` (3)

4 Concluzii

Graficele din Figura 14 rezuma performantele algoritmilor analizati. Astfel, algoritmul care se comporta cel mai bine ca timp de executie este `LRTA*`. O performanta temporala buna are si `BnB`. Acesta insa, are un cost de memorie mai mult decat triplu chiar si fata de `LRTA*`, despre care spuneam ca are nevoie de multa memorie pentru tabela H. Cel mai eficient din punctul de vedere al memoriei este algoritmul `IDA*`.

Cu alte cuvinte, daca sistemul pe care se ruleaza cautarea informata dispune de o cantitate mica de memorie, se recomanda utilizara algoritmului `IDA*`, chiar daca acesta nu aduce performante importante in materie de timp de executie. Daca ne permitem un consum de aproximativ 3 ori mai mare decat cel al lui `IDA*`, atunci algoritmul `LRTA*` poate fi folosit pentru diminuarea de aproape 6 ori a timpilor de executie fata de `DFID` si `IDA*`. `BnB` este de evitat, deoarece nu primeaza nici ca timp de executie, si cu siguranta nu este performant din punctul de vedere al memoriei utilizate.

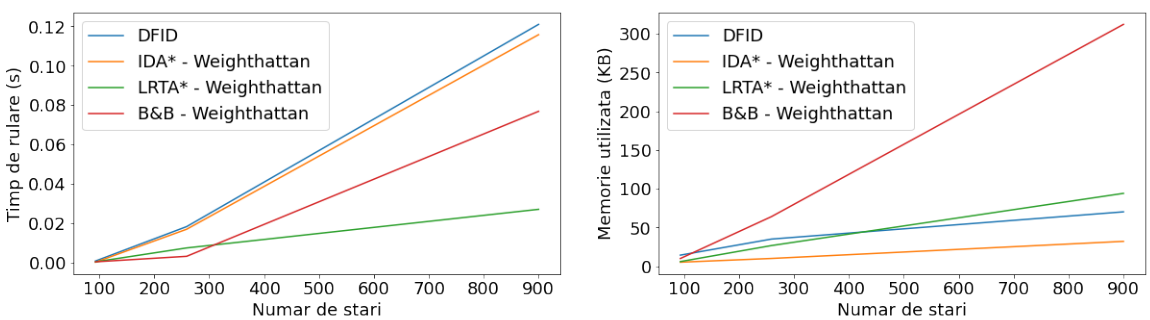


Fig. 14: Performantele tuturor algoritmilor, folosind euristica `Weighthattan`

Bibliografie

1. *Cursul de Inteligenta Artficiala: Cursul 2 - Strategii de cautare*
   https://curs.upb.ro/pluginfile.php/364789/mod_resource/content/1/IA_Lect_2_Strategii_Cautare.pdf
   Data ultimei accesari: 7 Nov 2020
2. *Cursul de Inteligenta Artficiala: Cursul 3 - Cautari online*
   https://curs.upb.ro/pluginfile.php/380310/mod_resource/content/1/IA_Lect_3_Online_CSP.pdf
   Data ultimei accesari: 9 Nov 2020
3. *Algoritmul Branch and Bound*
   https://artint.info/html/ArtInt_63.html
   Data ultimei accesari: 10 Nov 2020