

Inteligența Artificială - Tema 1

Strategii de căutare informată

Teodor-Stefan Dutu

Universitatea Politehnică București
Facultatea de Automatică și Calculatoare
Grupa 341C3

Abstract. Se vor compara performanțele, ca timp de execuție, memorie utilizată și număr stări explorate, ale unor algoritmi de căutare informată în spațiul stărilor.

Cuprins

1	Introdúcere	3
2	Cerinta 1 - Depth First Iterative Deepening	4
3	Algoritmi euristici	6
3.1	Cerinta 4 - Euristica propusa	6
3.2	Cerinta 2 - Iterative Deepening A*	7
3.3	Cerinta 3 - Learning Real-Time A*	10
3.4	Bonus - Branch and Bound	14
4	Concluzii	18

1 Introducere

Scopul meu e sa evaluez performantele unor algoritmi de cautare informata ce folosesc functii euristice, prin comparatie cu un algoritm 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 materie de timp si de memorie inregistrate de algoritmul `DFID` sunt cel prezentate in Figura 1 si in Tabelul 1.

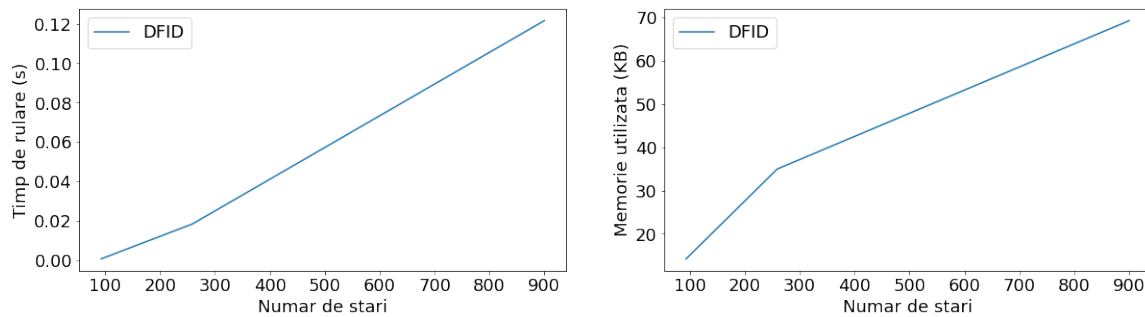
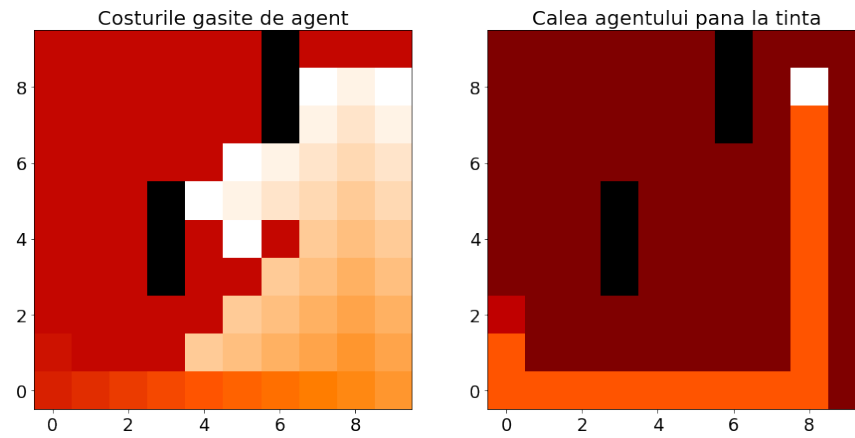


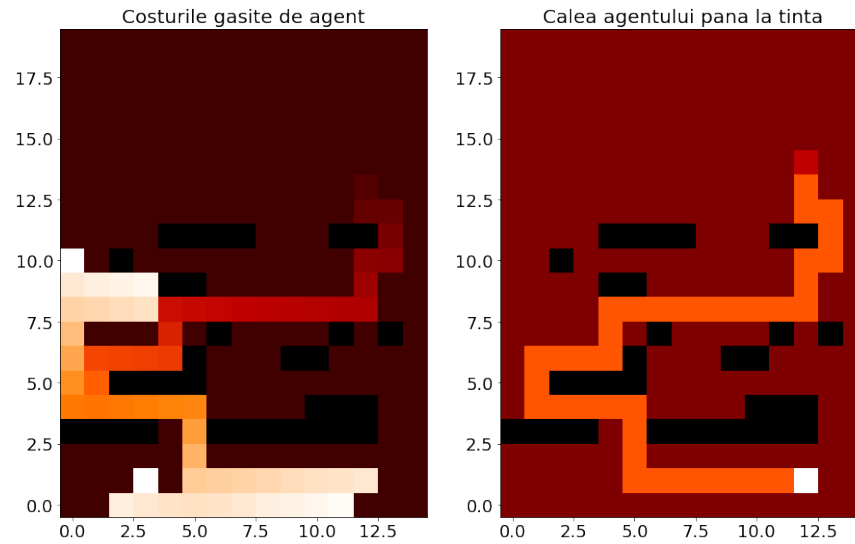
Fig. 1: Performantele algoritmului DFID

	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

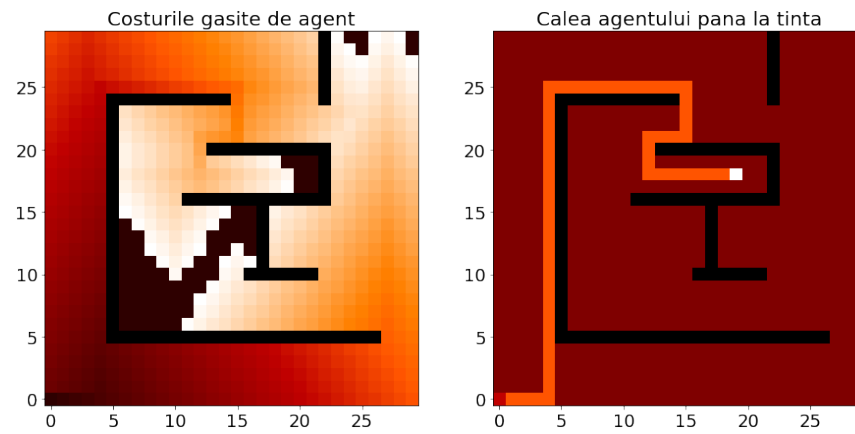
Table 1: Performantele algoritmului DFID



(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 - Euristică propusă

Se impune să scriu despre cerința 4 aici, înainte să prezint rezultatele algoritmilor care folosesc funcții euristice, deoarece fiecare dintre acești algoritmi este analizat folosind și euristica propusă în această secțiune.

În primul rând, trebuie spus că distanța euclidiană, numită în continuare **Euclid** este una slabă, deoarece mediul este discret, nu continuu, iar mișcarea agentului se poate face doar paralel cu axele de coordonate, nu și diagonal, iar **Euclid** se pretează unei libertăți mai mare de mișcare. O euristica bine cunoscută potrivită pentru mișcarea paralelă cu Ox și Oy este distanța **Manhattan**. Dar aceasta poate fi îmbunătățită, păstrându-i complexitatea constantă. Mai exact, euristica **Manhattan** consideră că fiecare tranziție are costul 1, ceea ce nu este adevărat în cazul hărții din fișierul **input3.txt**. Pentru o aproximare mai fidelă a distanței reale din oricare nod la cel țintă, h^* , putem înmulți distanța **Manhattan** cu **costul minim al oricărui arc din graful care modelează mediul**. Fie A mulțimea arcelor și N cea a nodurilor din acest graf. Astfel ia naștere euristica **Weighthattan** (*weight + manhattan*), definită formal astfel:

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

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

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

unde x_s și y_s sunt coordonatele carteziene ale stării s , iar x_f și y_f ale stării finale

Lema 1: Euristica **Weighthattan** este **consistentă**.

Demonstratie: Ca **Weighthattan** să fie consistentă, trebuie ca $\forall s \in N$:

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

Ultima ramură este evidentă, din definiția euristicii, deci ne rămâne de demonstrat că

$$h(s) \leq cost(s, s') + h(s') \quad \forall s' \in succ(s) \quad (3)$$

$$(2) + (3) \Rightarrow m(|y_f - y_s| + |x_f - x_s|) \leq cost(s, s') + m(|y_f - y_{s'}| + |x_f - x_{s'}|) \quad \forall s' \in succ(s) \quad (4)$$

unde cu $x_{s'}$ și $y_{s'}$ am notat poziția stării s' .

Din definiția lui $m \Rightarrow m \leq cost(s, s')$, deci pentru a demonstra ecuația (4) este suficient să demonstrăm că:

$$m(|y_f - y_s| + |x_f - x_s|) \leq m + m(|y_f - y_{s'}| + |x_f - x_{s'}|) \quad \forall s' \in succ(s) \quad (5)$$

$$(5) \Leftrightarrow |y_f - y_s| + |x_f - x_s| \leq 1 + |y_f - y_{s'}| + |x_f - x_{s'}| \quad \forall s' \in succ(s) \quad (6)$$

Relația (6) reprezintă relația de consistență pentru euristica **Manhattan**, care este consistentă, deci și **Weighthattan** este consistentă (Q.E.D). Fiind consistentă, euristica **Weighthattan** este și **admisibilă**.

Observație 1: Dacă $\exists s_i, s_j \in N$ a.i. $arc(s_i, s_j) \in N$ și $cost(s_i, s_j) = 1$, atunci $weighthattan(s) = manhattan(s) \quad \forall s \in N$.

Observație 2: Euristica **Weighthattan** o domină pe cea **Manhattan**.

Din *Observațiile 1, 2* și din *Lema 1*, ne așteptăm ca performanțele algoritmilor care folosesc euristici să fie identice pentru euristicele **Manhattan** și **Weighthattan**, la rularea pe hărțile din fișierele **input1.txt** și **input2.txt**, unde costurile minime sunt 1, și ne așteptăm să se îmbunătățească la rularea algoritmilor pe harta din fișierul **input3.txt**, unde costul minim este supraunitar, și anume 2. Graficele din secțiunile următoare confirmă aceste așteptări.

3.2 Cerinta 2 - Iterative Deepening A*

Asa cum am mentionat in Sectiunea 1, euristicele 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 semnificatie 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.

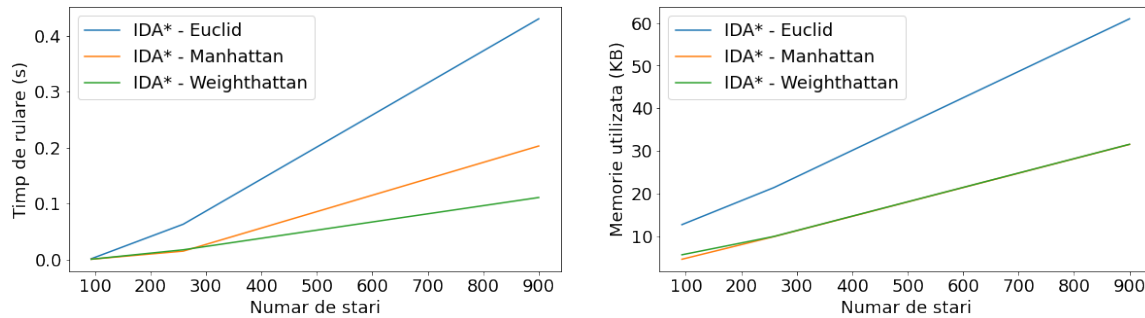


Fig. 3: Performantele algoritmului IDA*

	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*

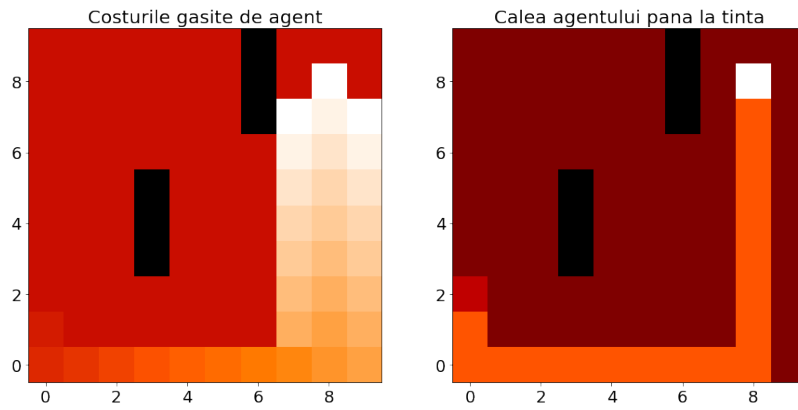
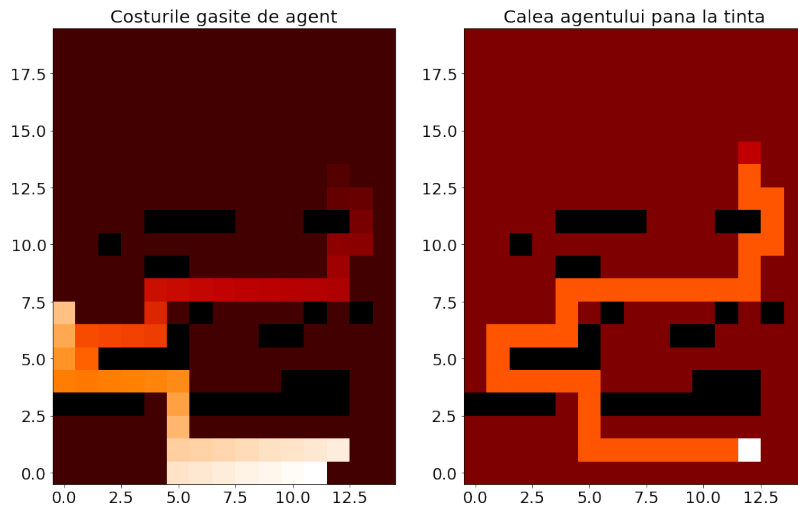
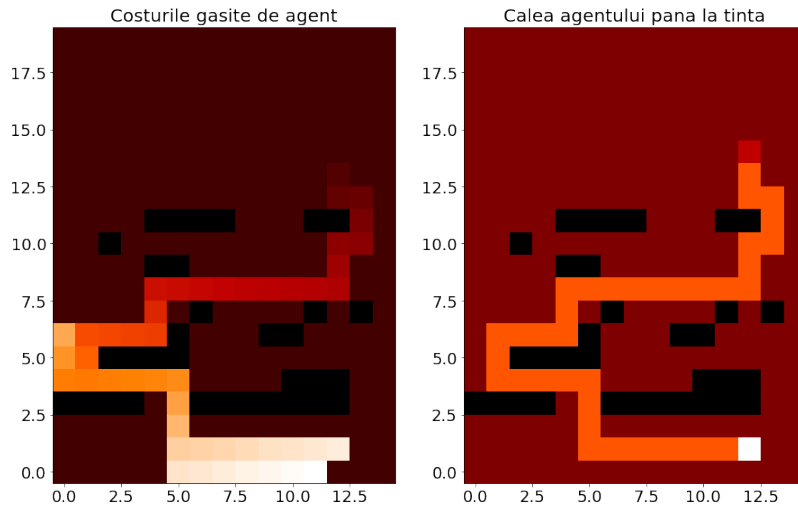
(a) Comportamentul agentului pe harta din `input1.txt` cu orice euristica(b) Comportamentul agentului pe harta din `input2.txt` cu euristica Euclid(c) Comportamentul agentului pe harta din `input2.txt` cu euristicile Manhattan si Weighthattan

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

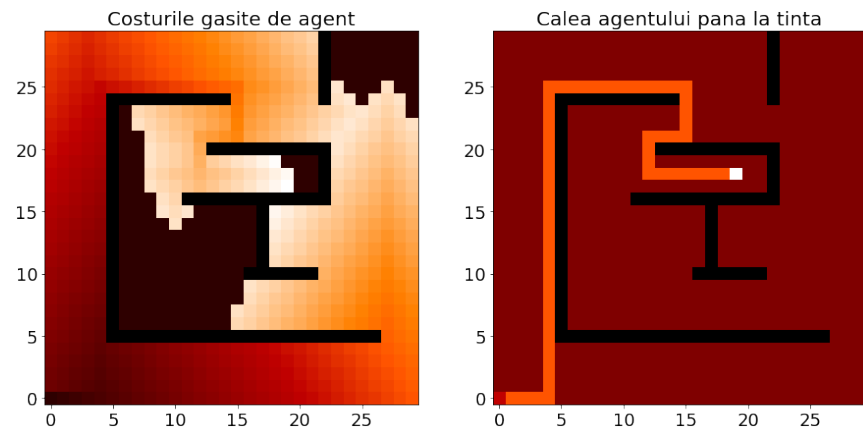
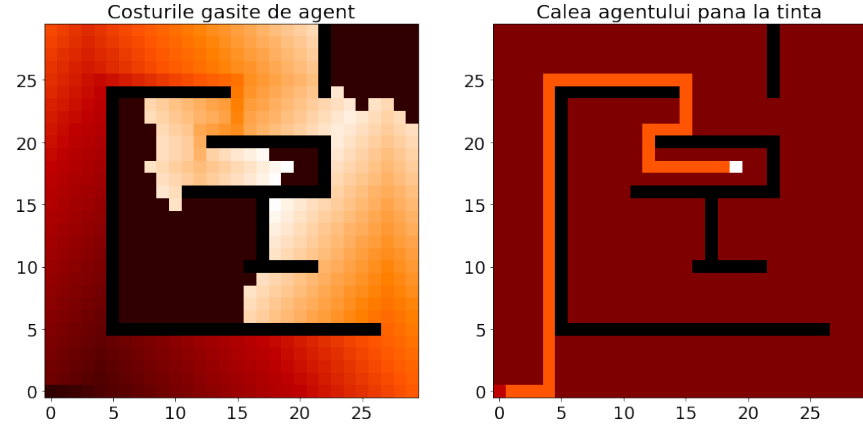
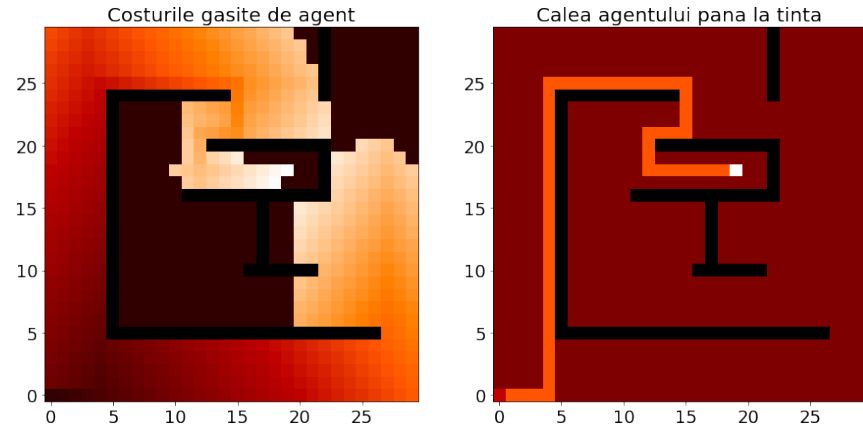
(a) Comportamentul agentului pe harta din `input3.txt` cu euristica Euclid(b) Comportamentul agentului pe harta din `input3.txt` cu euristica Manhattan(c) Comportamentul agentului pe harta din `input3.txt` cu euristica Weighthattan

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 euclidiană, 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 rulara 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 euristicele 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 rulara sa pe fisierul `input3.txt`, ale carei rezultate pot fi analizate in Figura 9. Astfel, folosind euristica Weighthattan, timpul de rulare este aproape de 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.

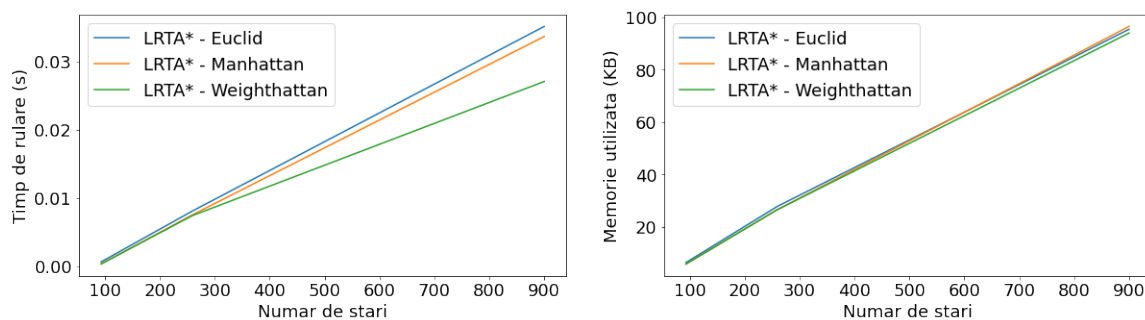
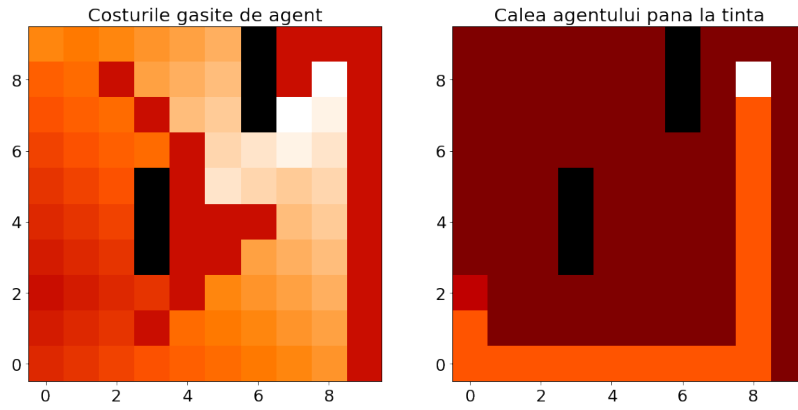


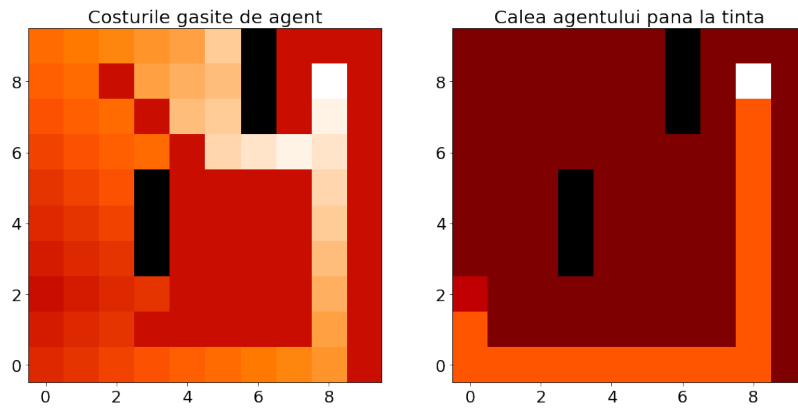
Fig. 6: Performantele algoritmului LRTA*

	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)

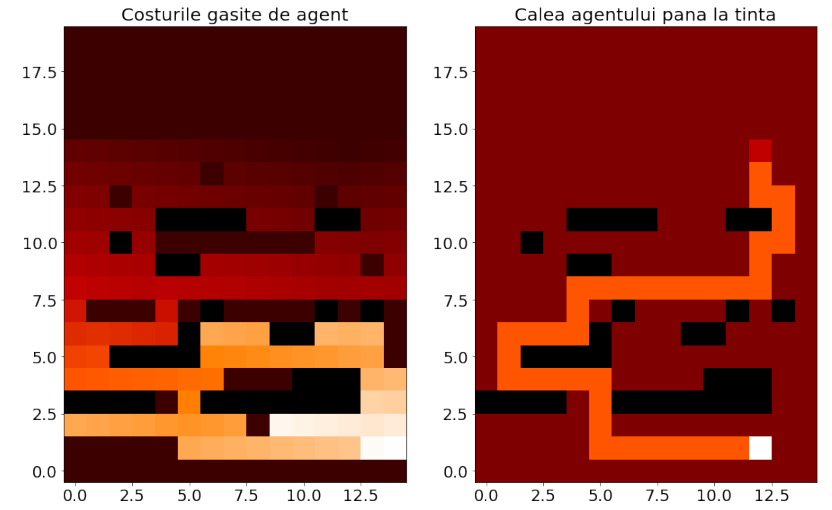
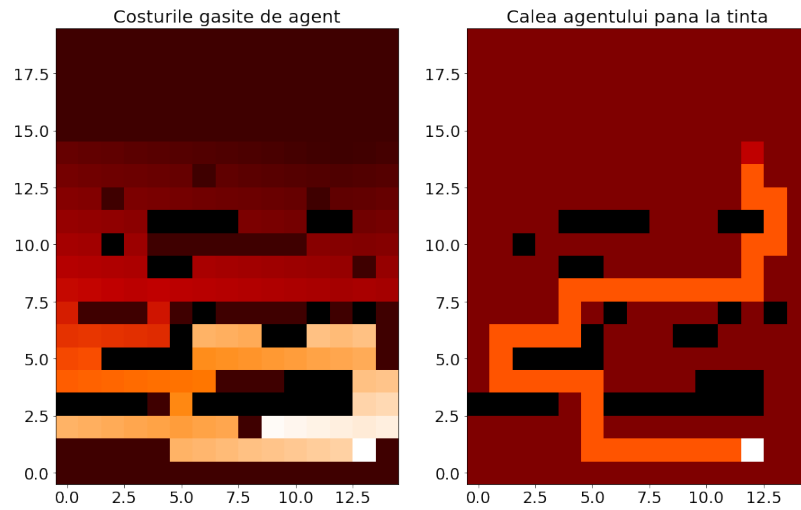
(a) Comportamentul agentului pe harta din `input2.txt` cu euristica Euclid(b) Comportamentul agentului pe harta din `input2.txt` cu euristicile Manhattan si Weighthattan

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

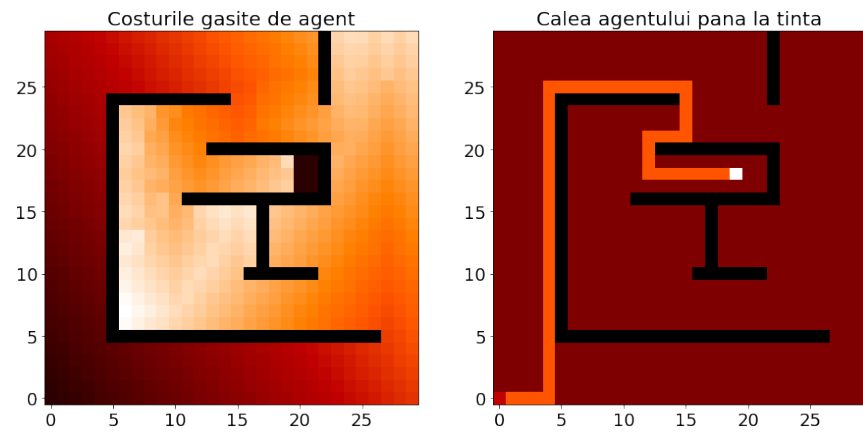
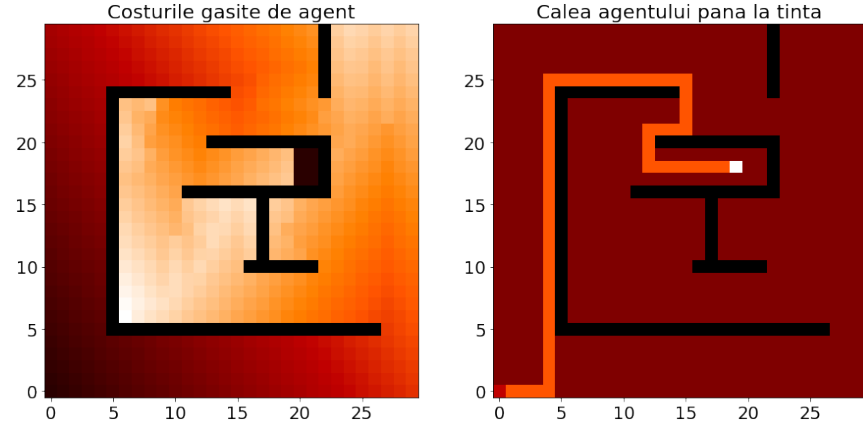
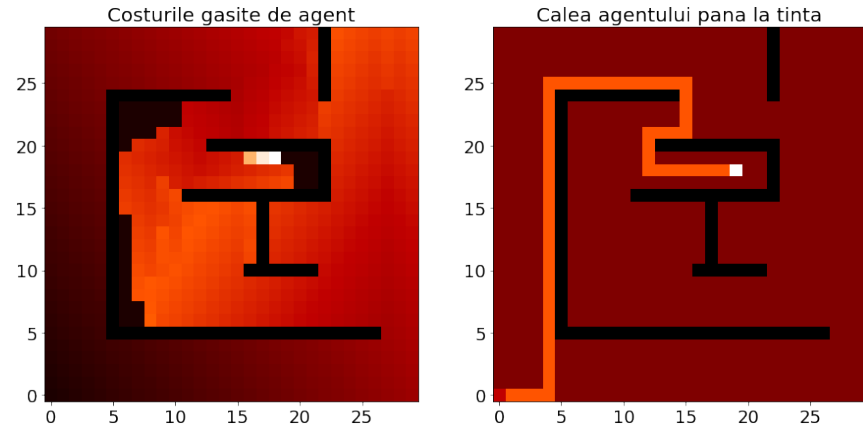
(a) Comportamentul agentului pe harta din `input3.txt` cu euristica Euclid(b) Comportamentul agentului pe harta din `input3.txt` cu euristica Manhattan(c) Comportamentul agentului pe harta din `input3.txt` cu euristica Weighthattan

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 aceasta este $g(s)$, avem:

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

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

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

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, euristicele pot fi comparate in Figura 10 si in Tabelul 4, iar costurile gasite de acest algoritim, 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.

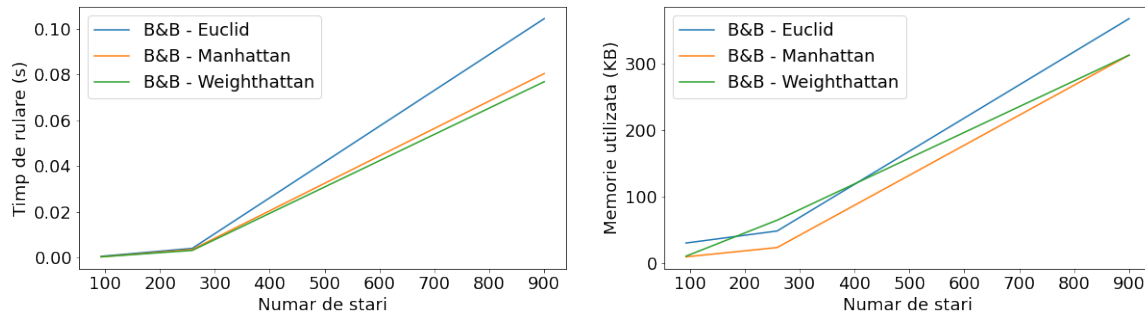
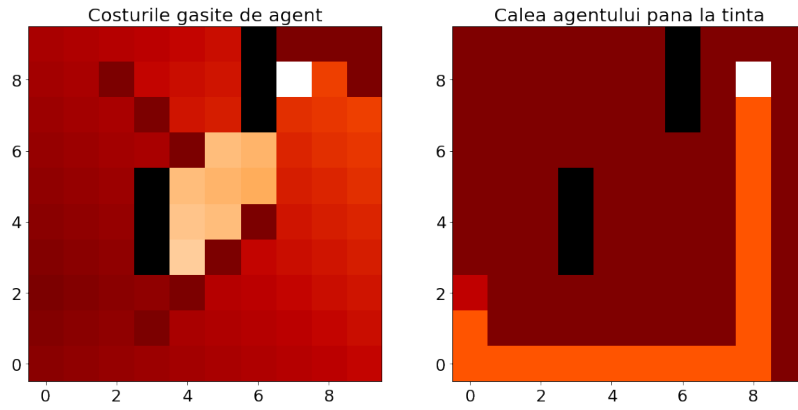
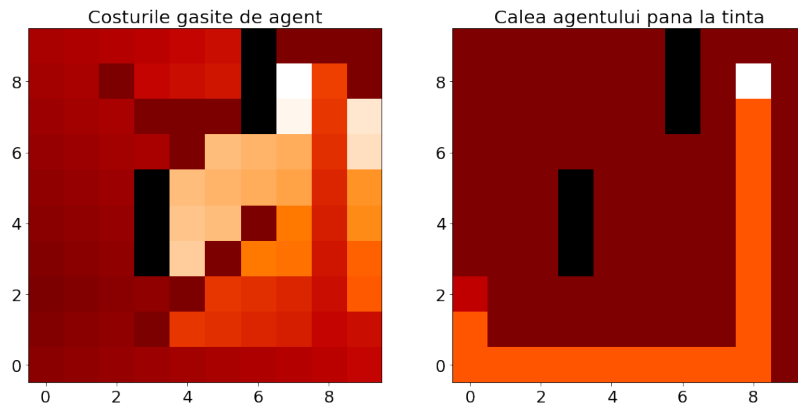


Fig. 10: Performantele algoritmului BnB

	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	395.75 KB
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)

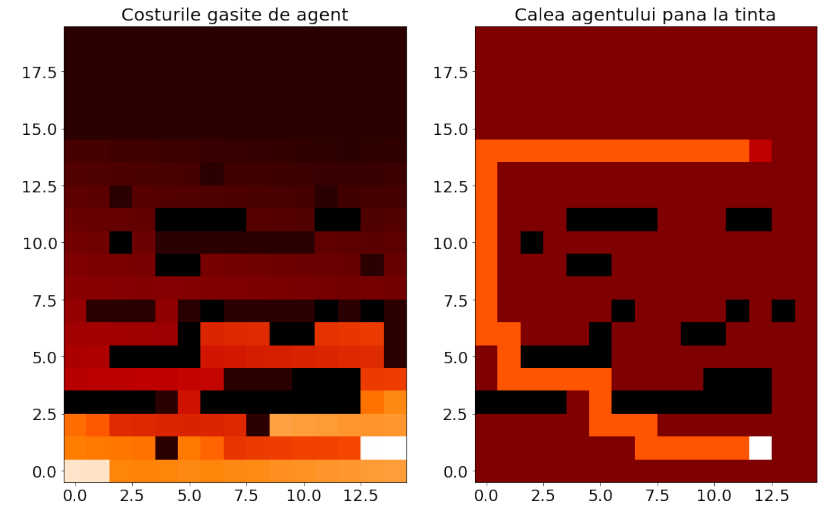
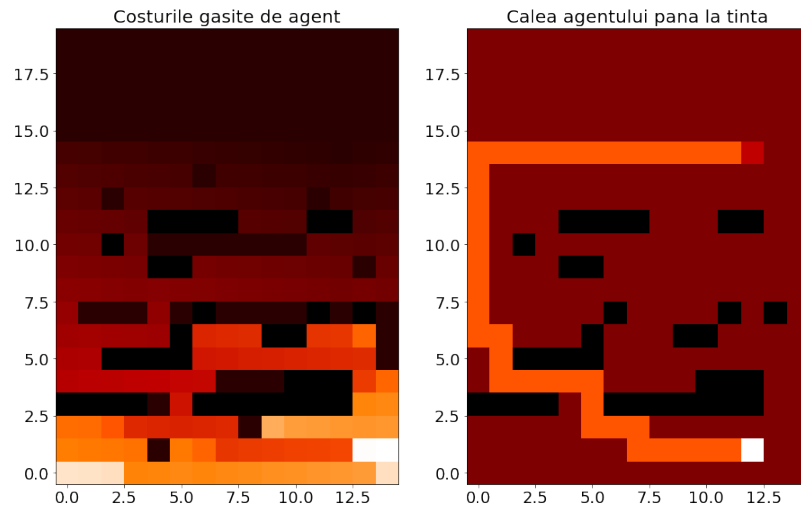
(a) Comportamentul agentului pe harta din `input2.txt` cu euristica Euclid(b) Comportamentul agentului pe harta din `input2.txt` cu euristicile Manhattan si Weighthattan

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

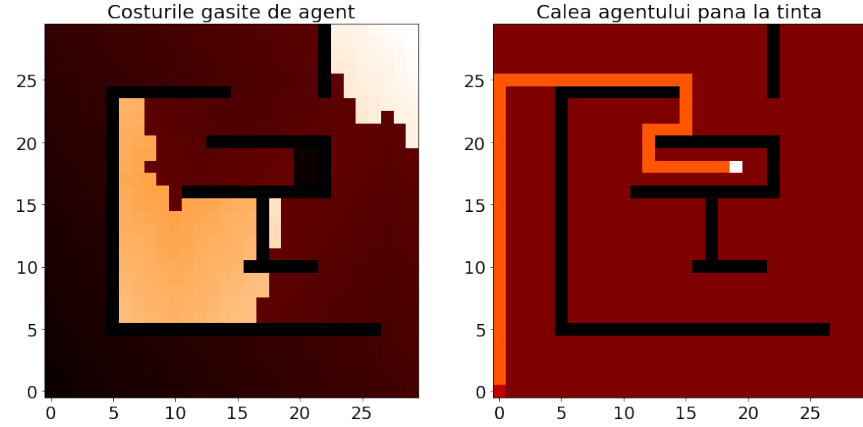
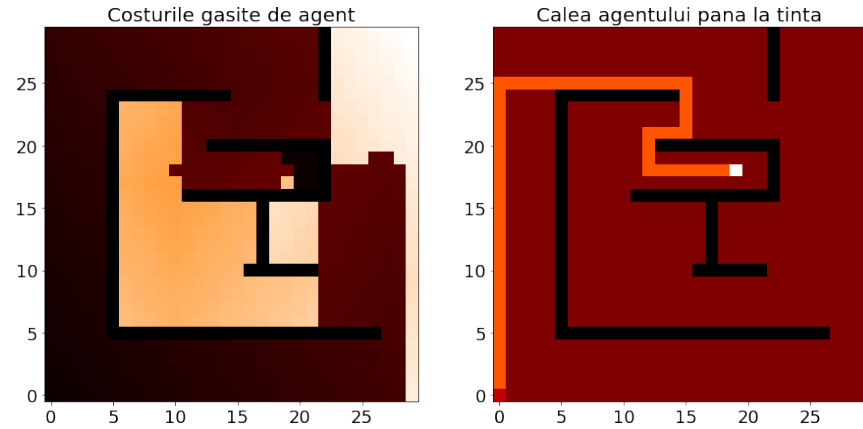
(a) Comportamentul agentului pe harta din `input3.txt` cu euristica Euclid(b) Comportamentul agentului pe harta din `input3.txt` cu euristica Manhattan(c) Comportamentul agentului pe harta din `input3.txt` cu euristica Weighthattan

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 utilizarea 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.

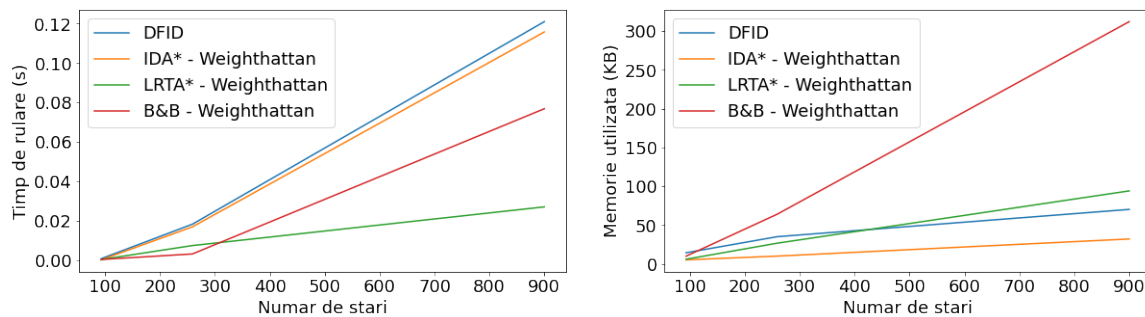


Fig. 14: Performantele tuturor algoritmilor, folosind euristica **Weighthattan**

Bibliografie

1. *Cursul de Inteligenta Artificiala: 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 Artificiala: 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