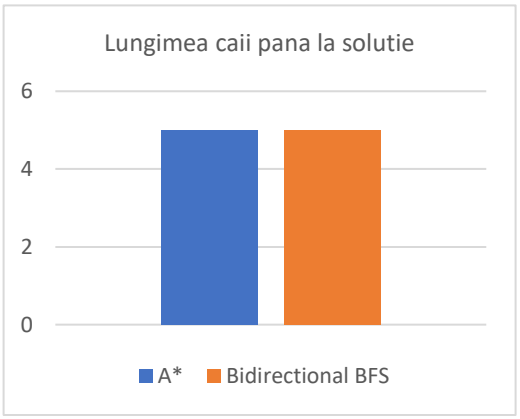
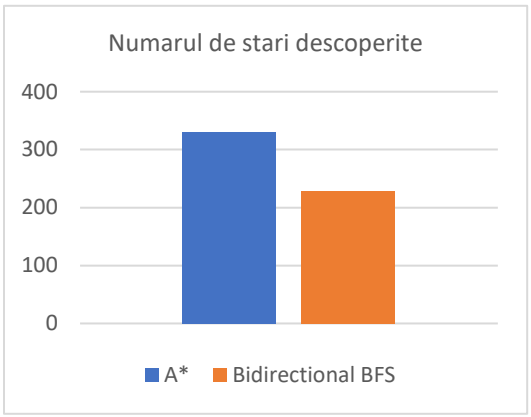
Rubik Cube

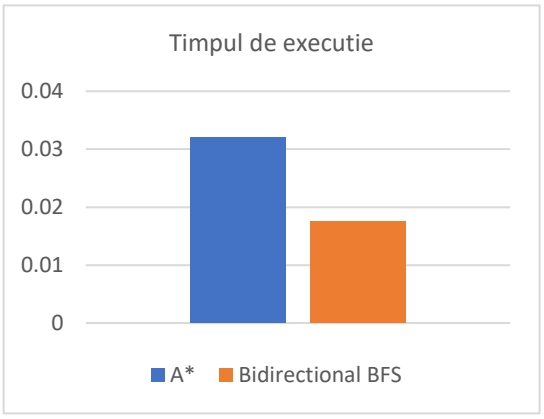
3.1. A* si Bidirectional BFS

Pentru A* am definit o euristica, h1, ce numara toate patratelele diferite de cubul rezolvat, iar apoi le imparte la 8 pentru a fi admisibila(se schimba cate 8 patratele la fiecare mutare).

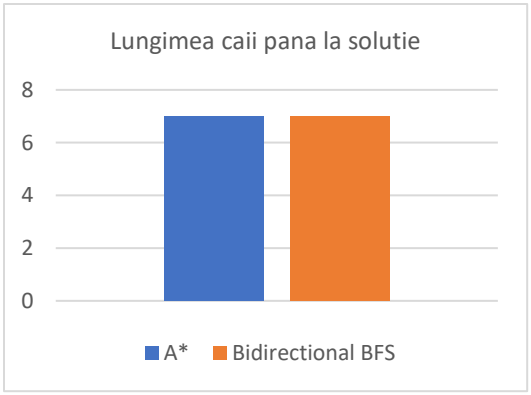
- Cube 1

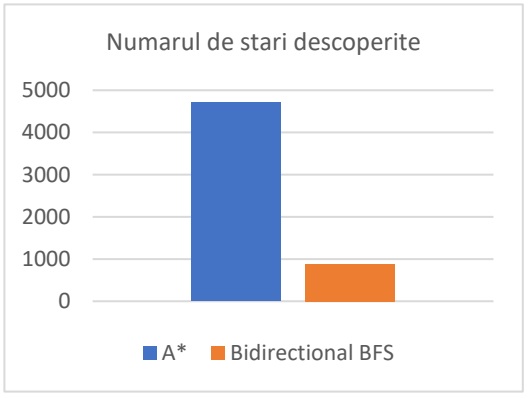


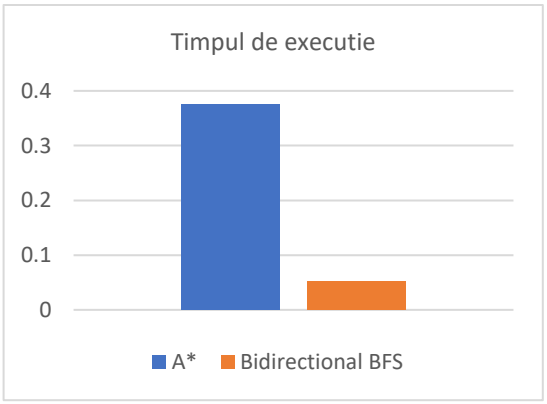




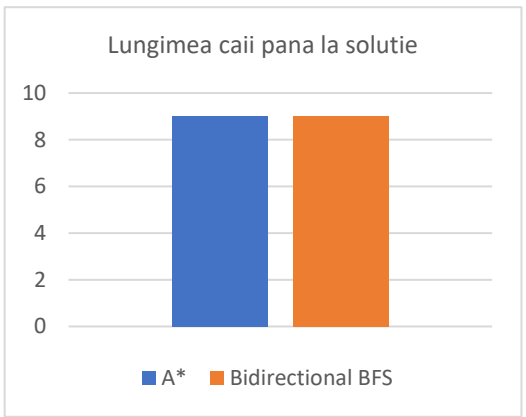
- Cube 2

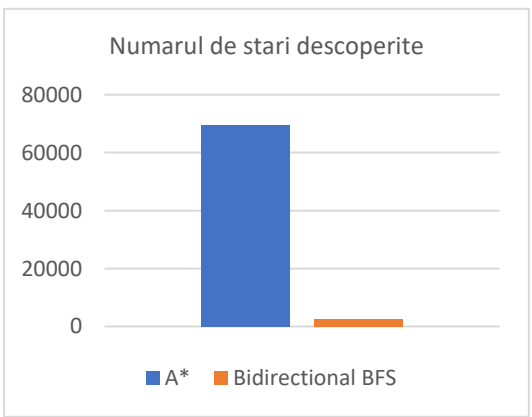


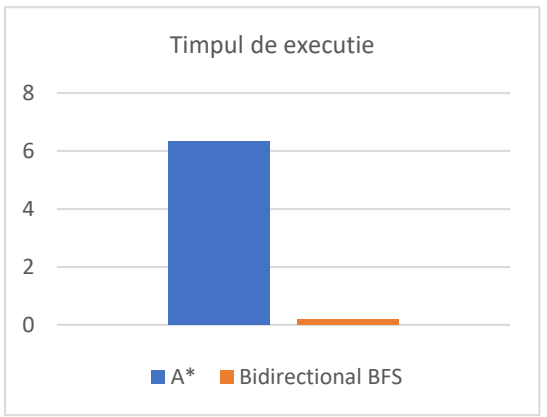




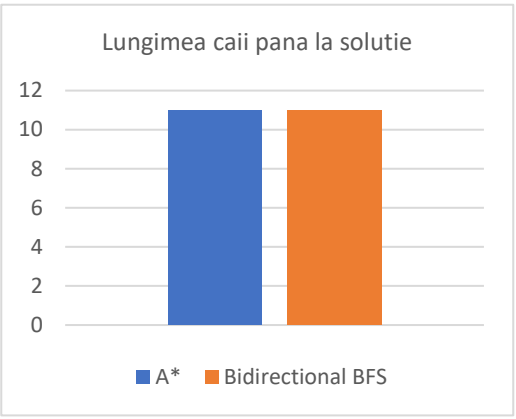
- Cube 3

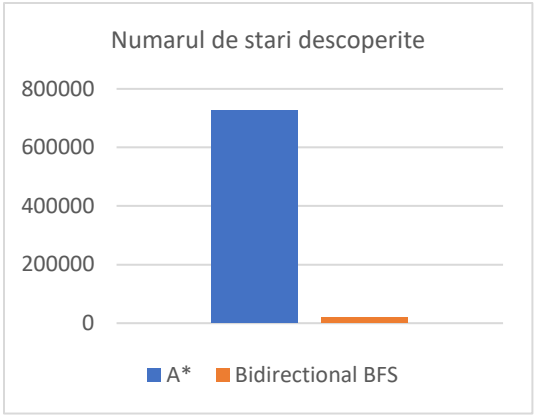


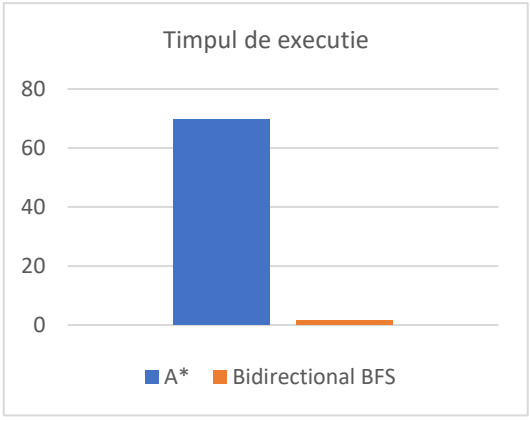




- Cube 4







In urma comaratiei celor 2 algoritmi pe baza cazurilor de test date pot trage urmatoarele concluzii:

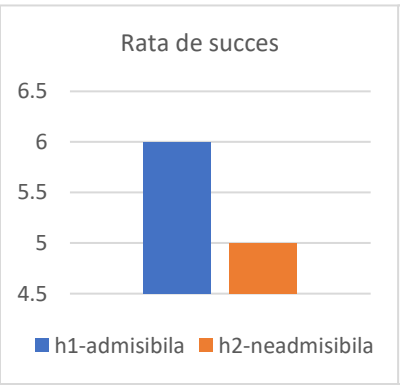
- Ambii algoritmi ofera solutii corecte(cel mai rapid path pentru rezolvarea cubului rubic)
- Cu cat complexitatea cubului creste, cu atat creste si diferenta de performante dintre cei 2 algoritmi. Si numarul starilor descoperite si timpul de executie sunt mult mai bune in cazul BFS ului bidirectional(testele 2, 3, 4) si doar putin main bune in cazul unui cub rubic cu rezolvare rapida(pana in 5 mutari).

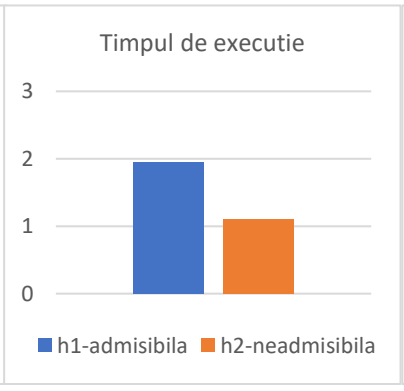
3.2. Monte Carlo Tree Search

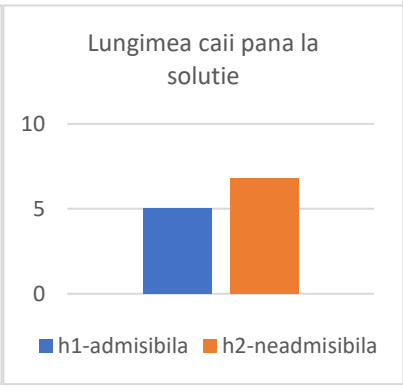
Pentru cea de a doua euristica am ales sa numar patratelele alaturate care nu au aceeasi culoare, fara a mai imparti la 8. Astfel aceasta este una neadmisibila.

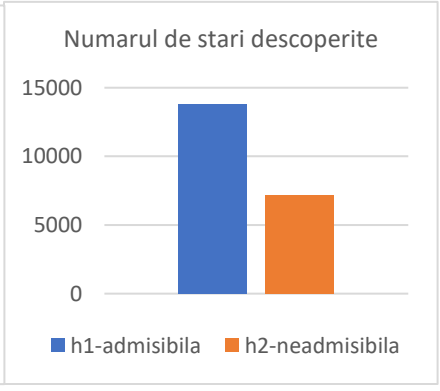
Pentru comparatia performantelor algoritmilor pentru diverse cuburi, bugete, C am ales sa compar doar cazurile in care se ajunge la un rezultat bun, astfel realizand si o comparatie pentru success rate / 20.

- Cube 1, budget=1000, C=0.5

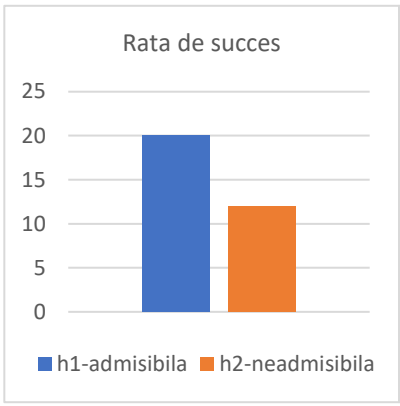


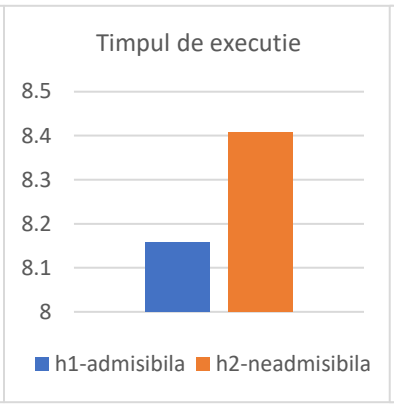


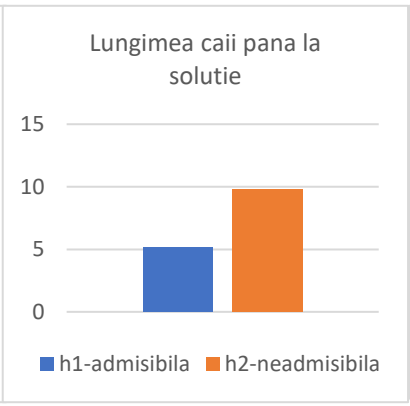


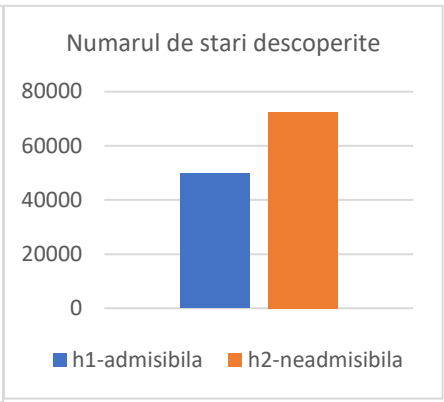


- Cube 1, budget=5000, C=0.5

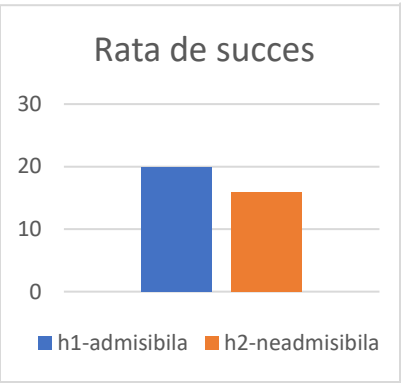


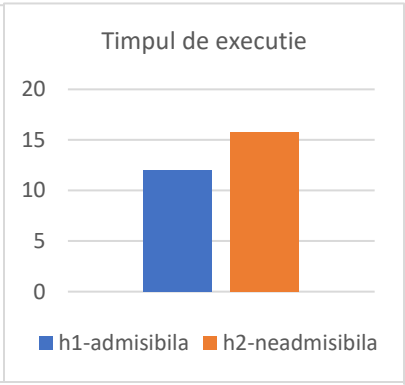


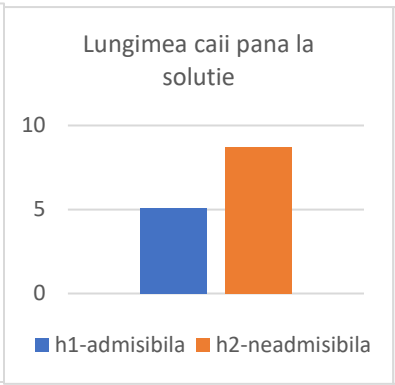


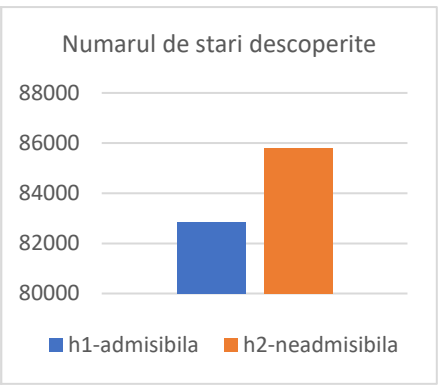


- Cube 1, budget=10000, C=0.5

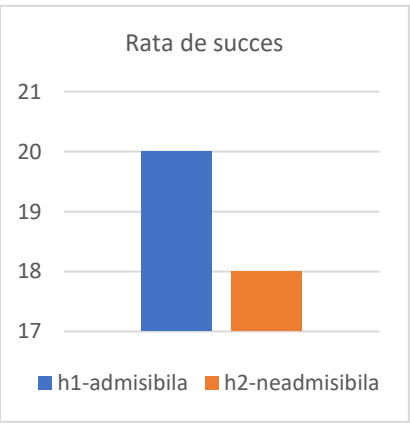


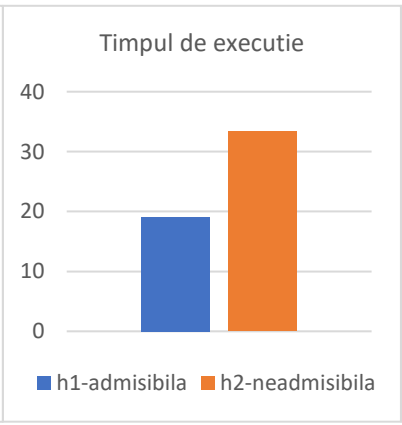


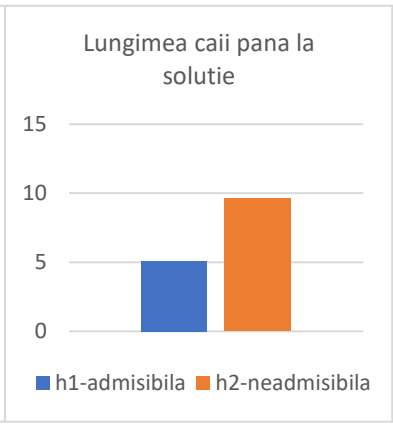


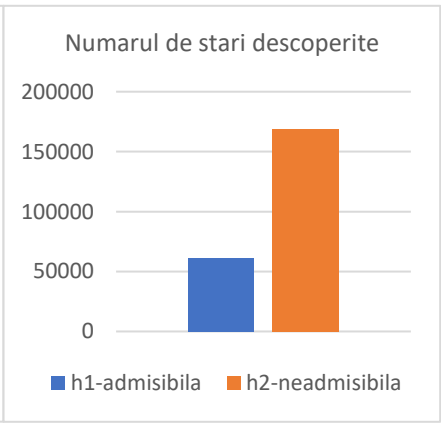


- Cube1, budget=20000, C=0.5

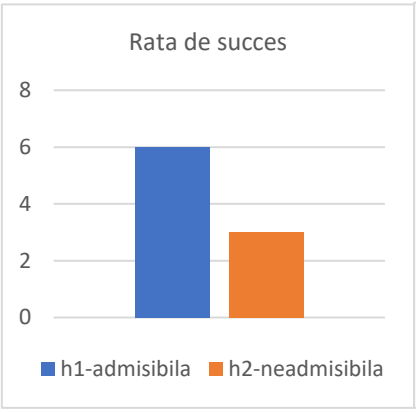


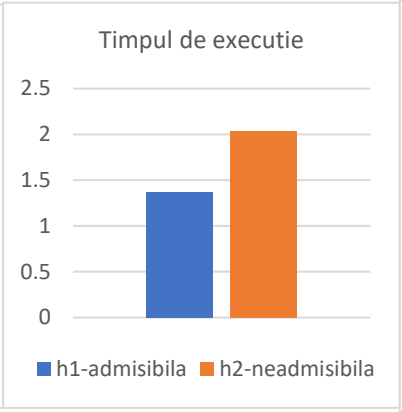


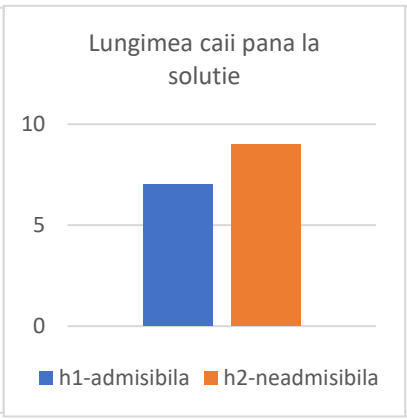


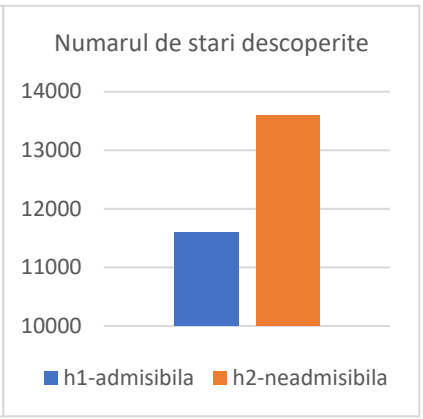


- Cube1, budget=1000, C=0.1

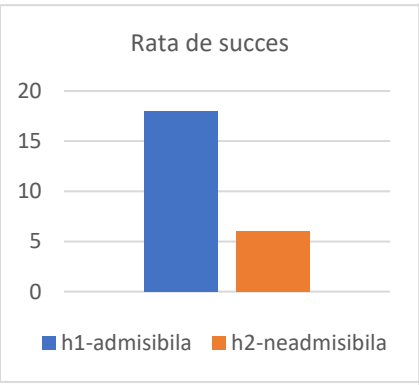


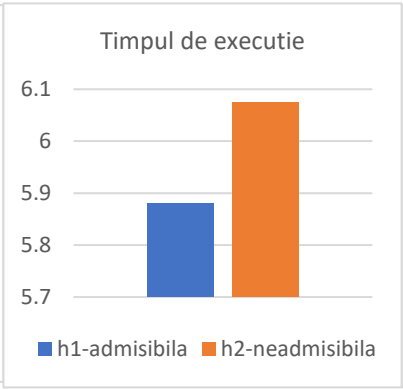


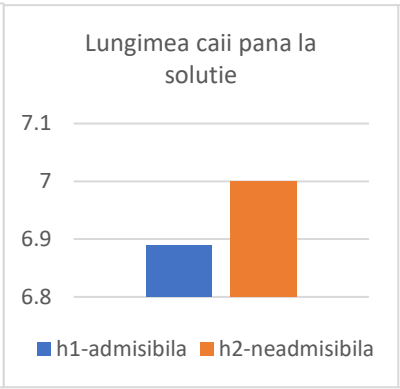


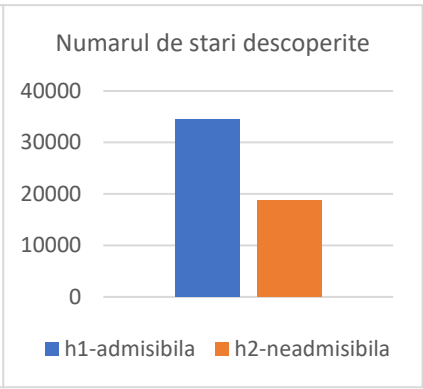


- Cube1, budget=5000, C=0.1

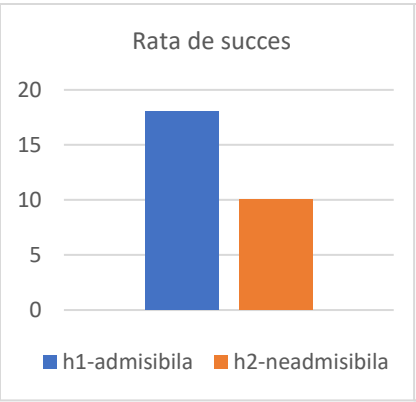


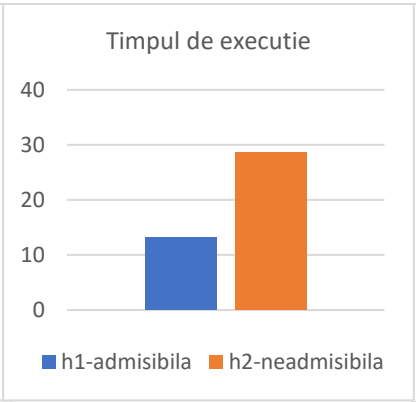


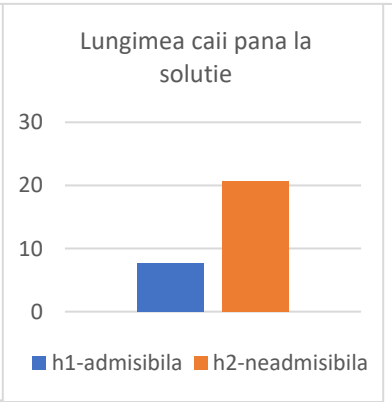


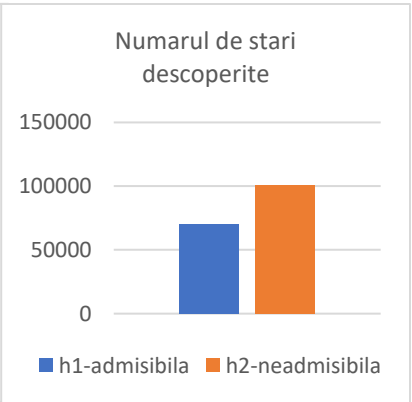


- Cube1, budget=10000, C=0.1

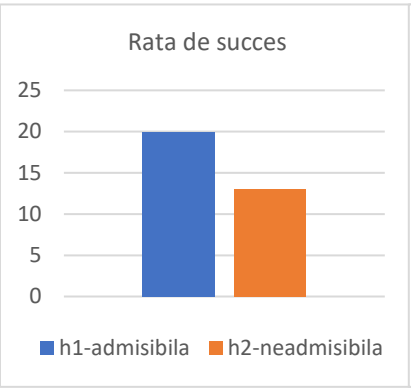


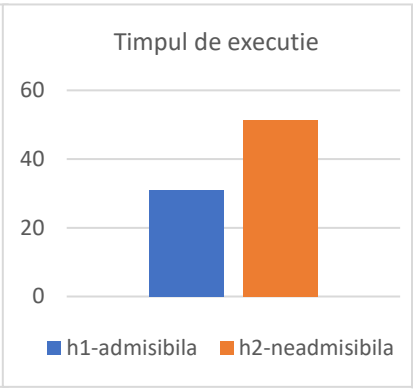


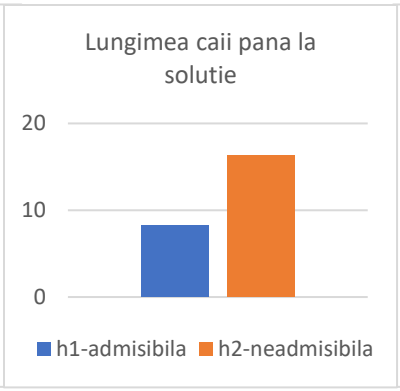


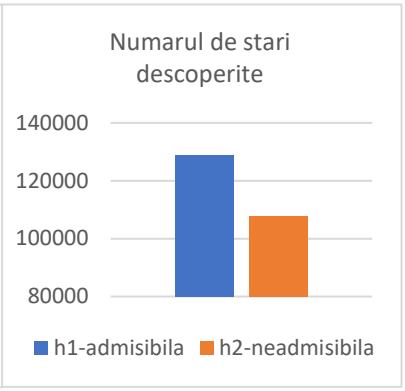


- Cube1, budget=20000, C=0.1

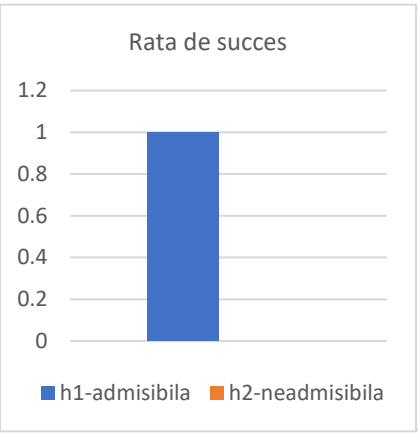


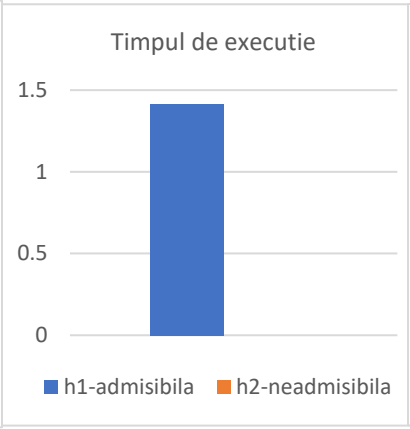


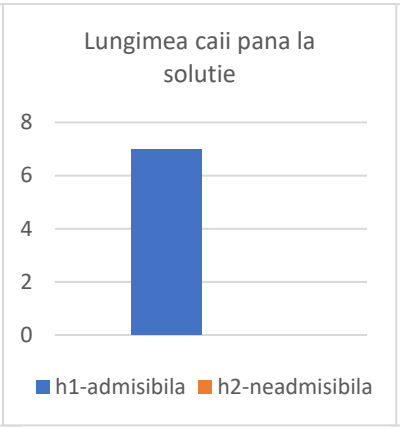


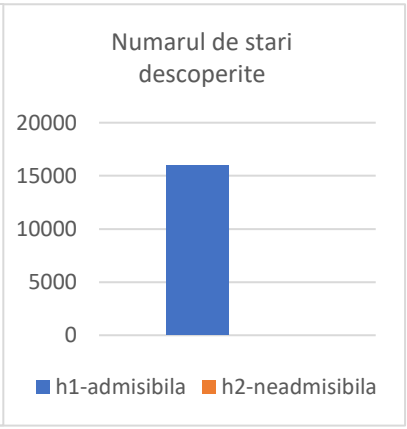


- Cube2, budget=1000, C=0.5

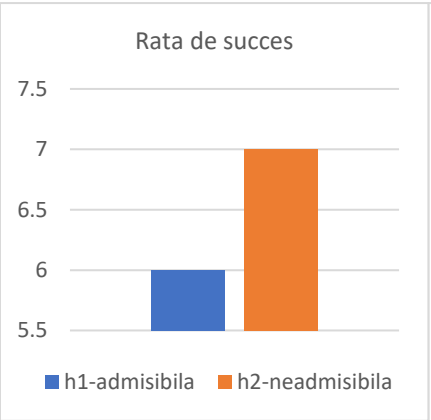


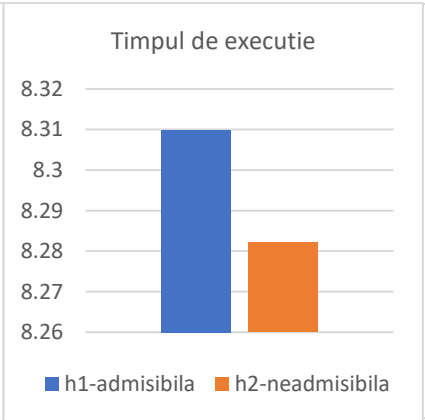


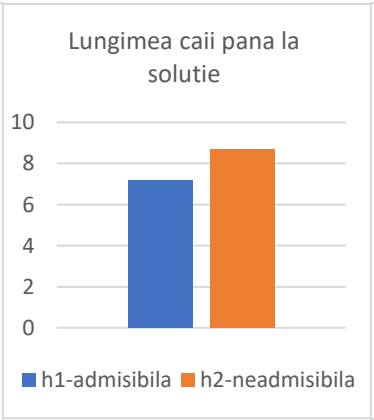


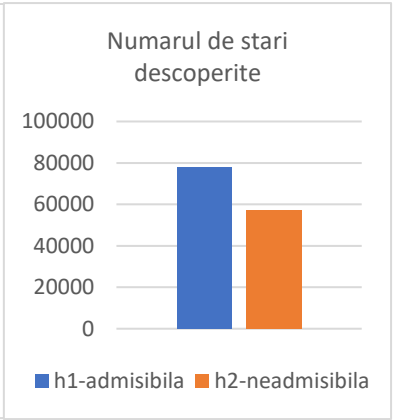


- Cube2, budget=5000, C=0.5

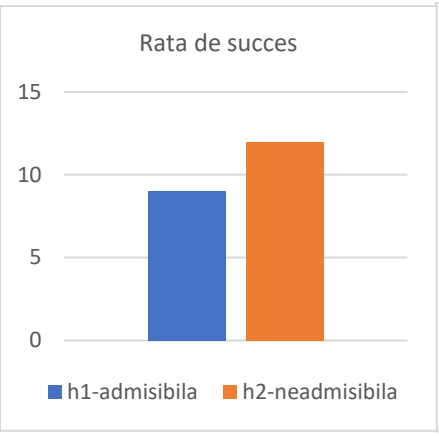


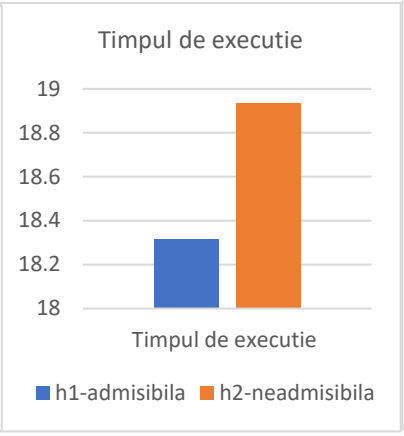


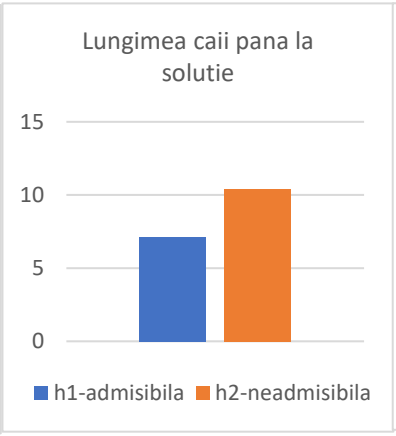


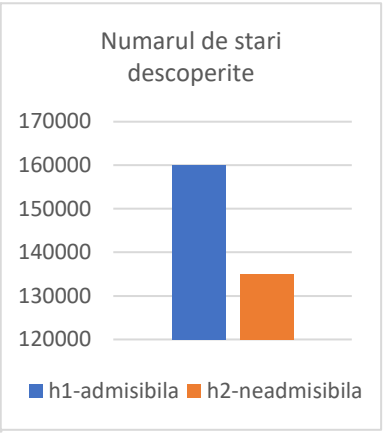


- Cube2, budget=10000, C=0.5

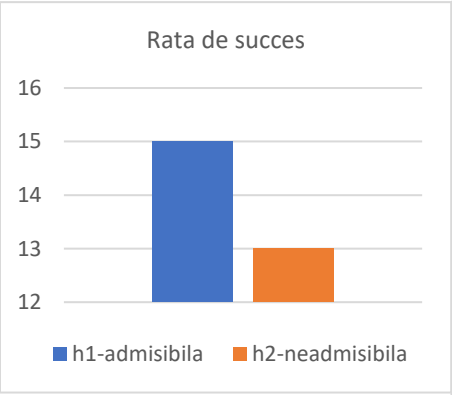


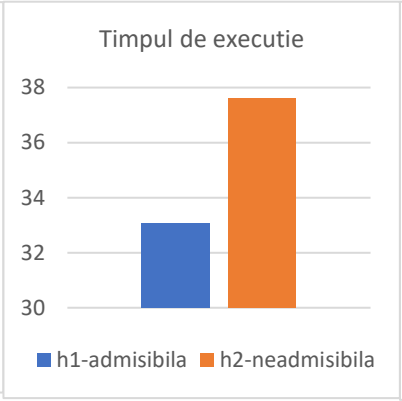


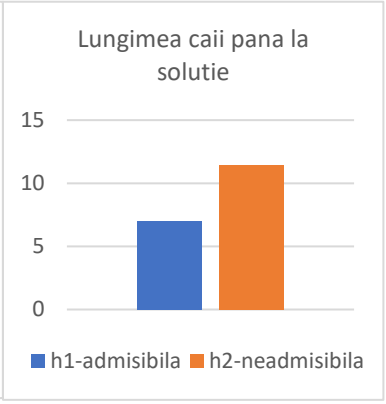


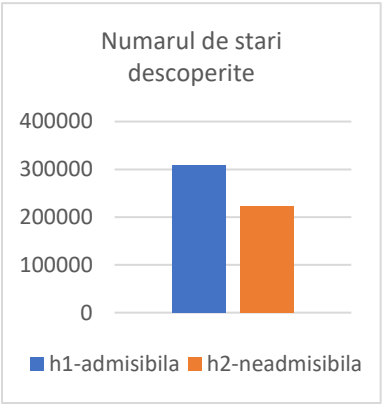


- Cube2, budget=20000, C=0.5

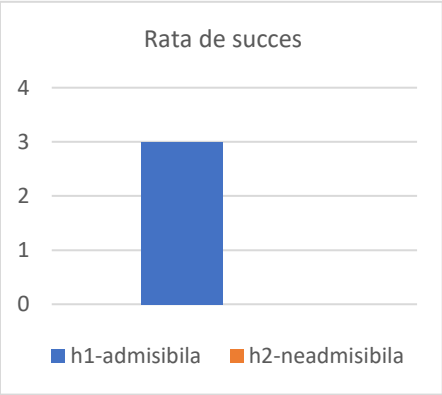


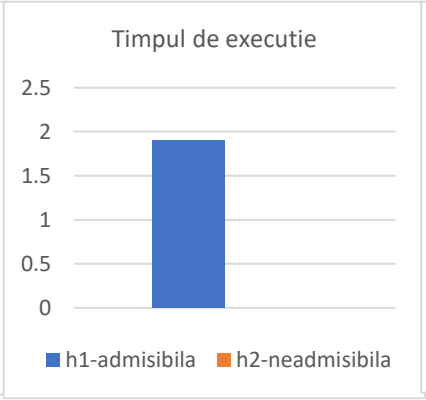


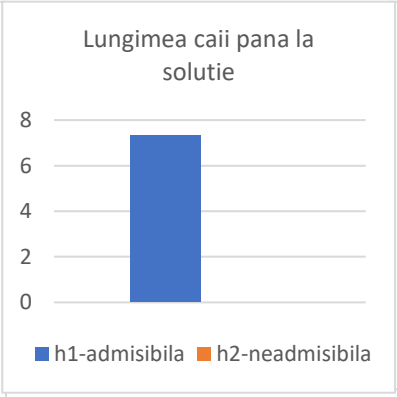


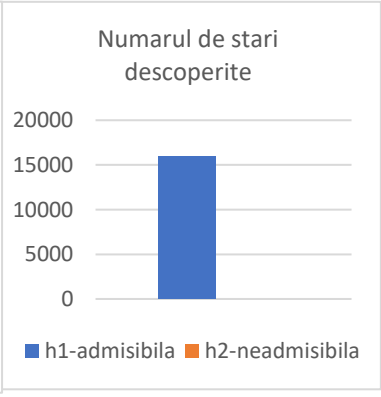


- Cube2, budget=1000, C=0.1

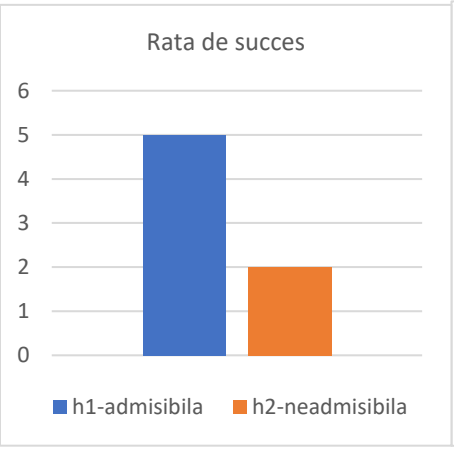


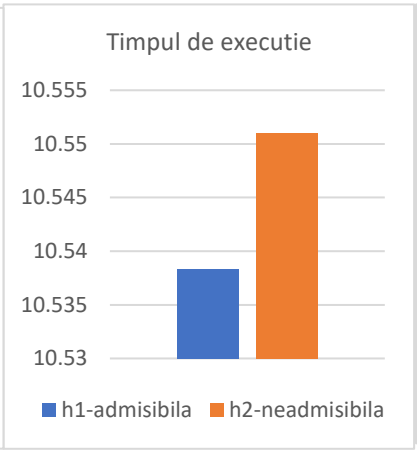


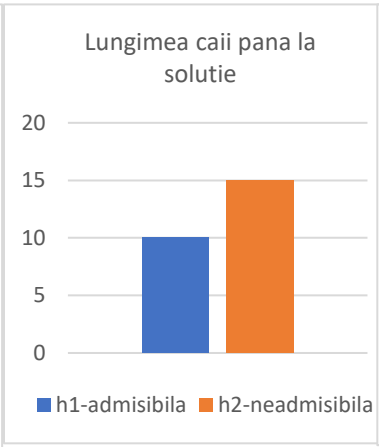


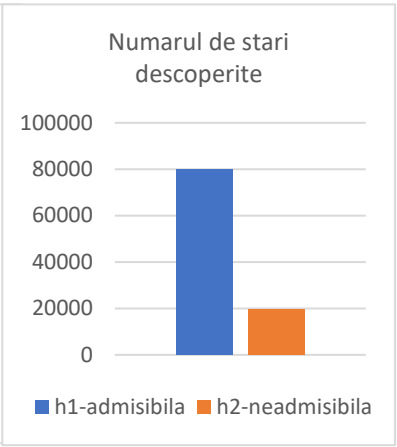


- Cube2, budget=5000, C=0.1

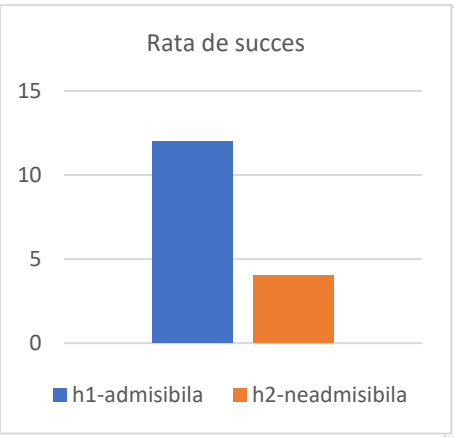


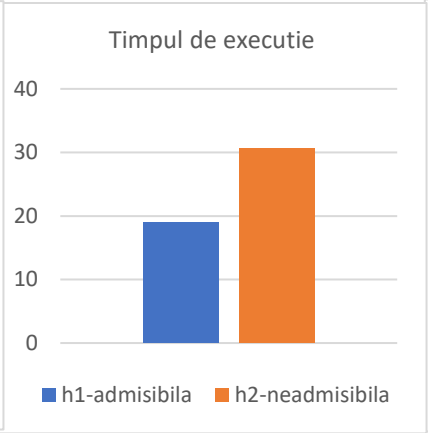


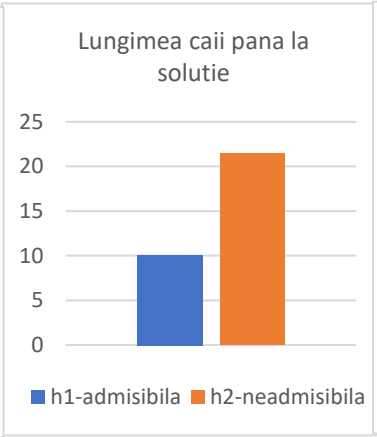


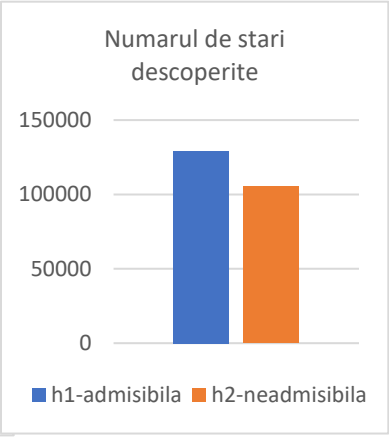


- Cube2, budget=10000, C=0.1

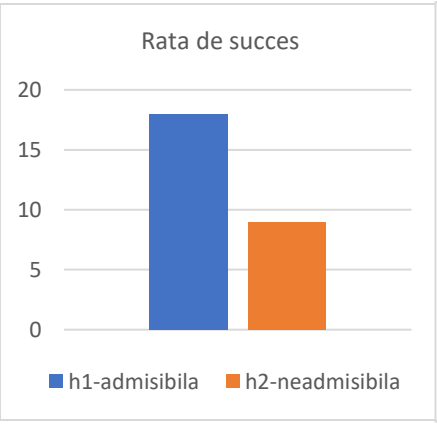


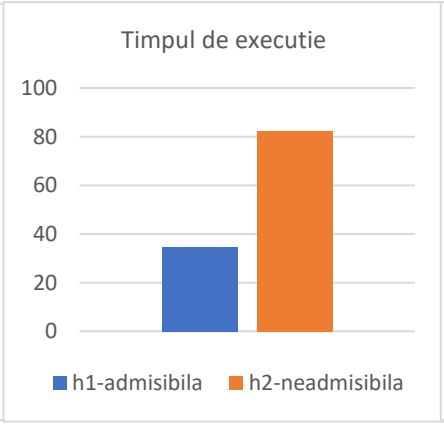


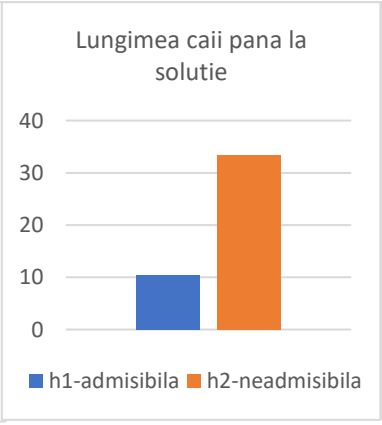


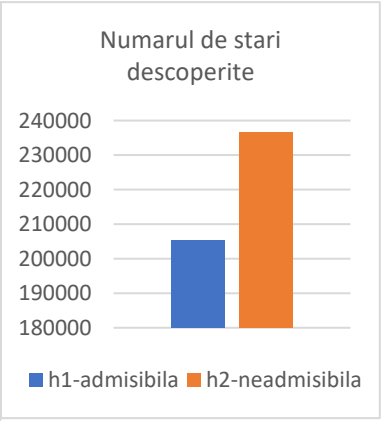


- Cube2, budget=20000, C=0.1

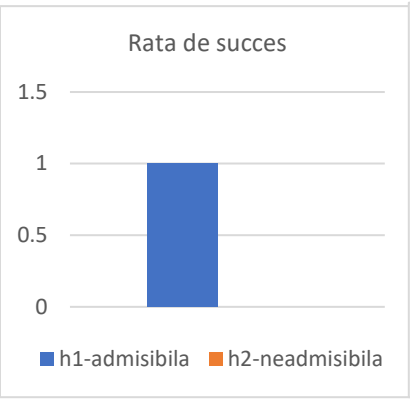


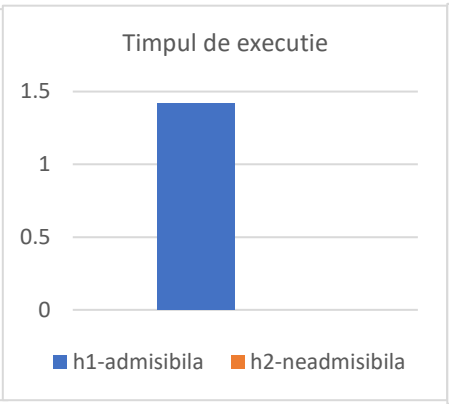


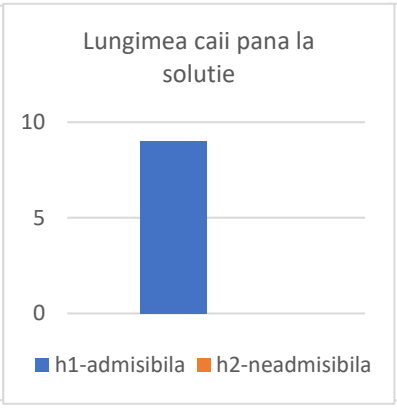


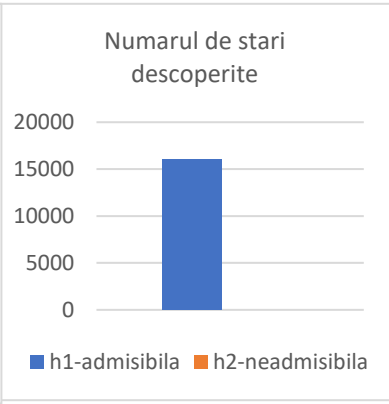


- Cube3, budget=1000, C=0.5

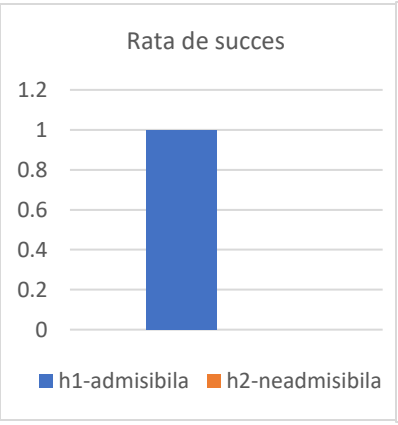


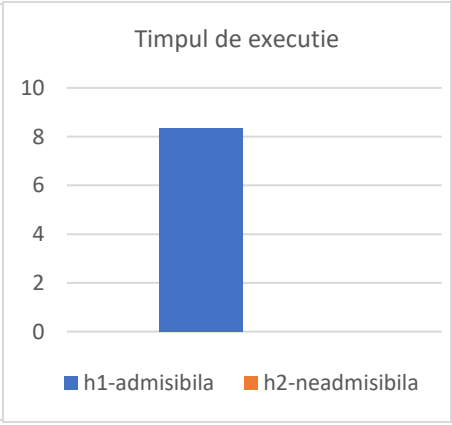


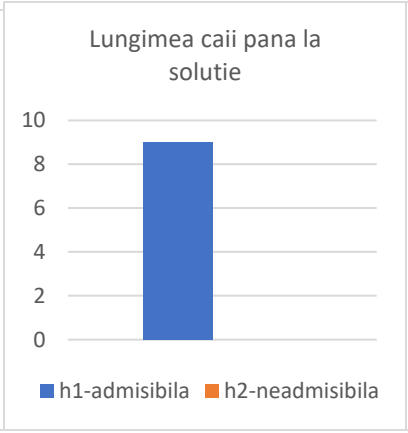


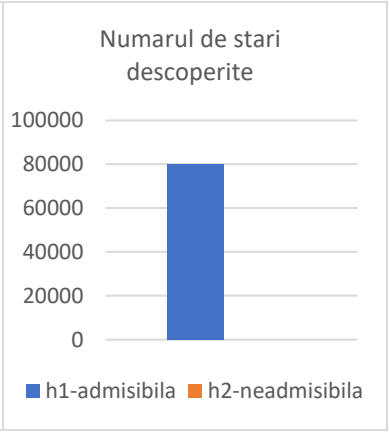


- Cube3, budget=5000, C=0.5

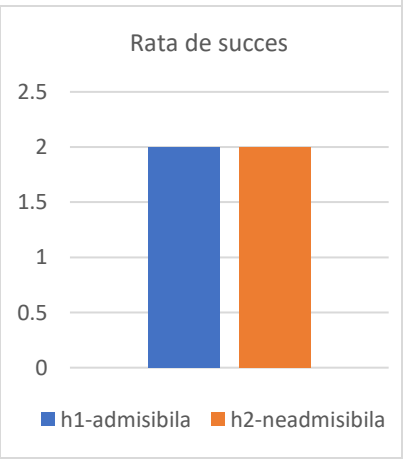


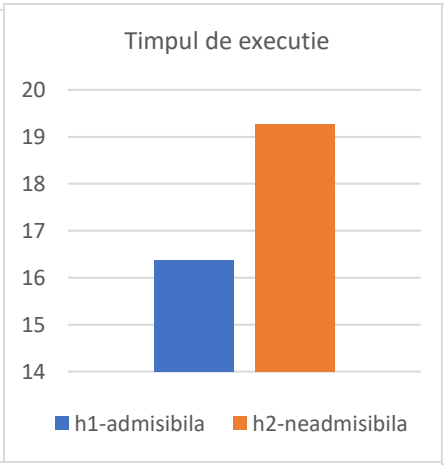


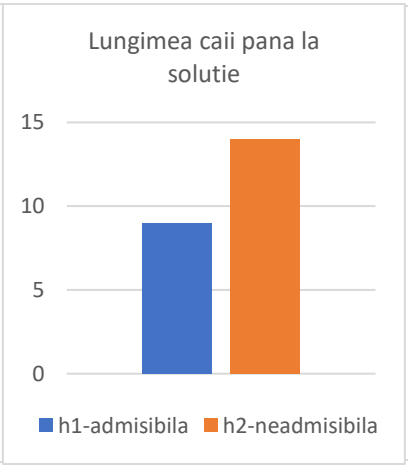


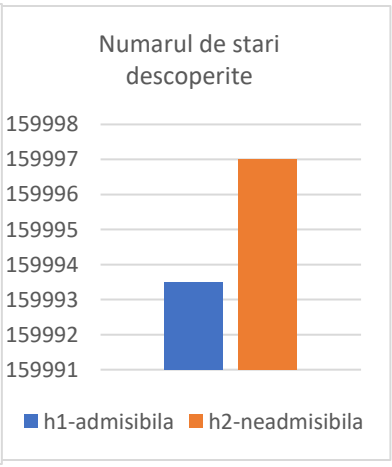


- Cube3, budget=10000, C=0.5

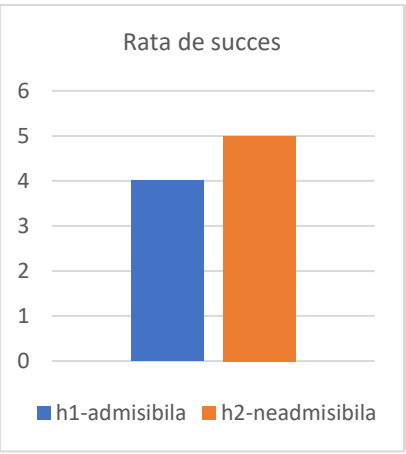


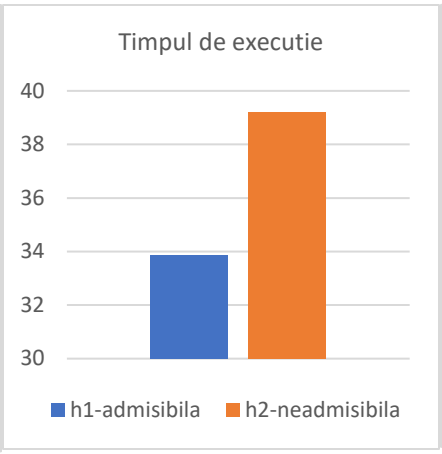


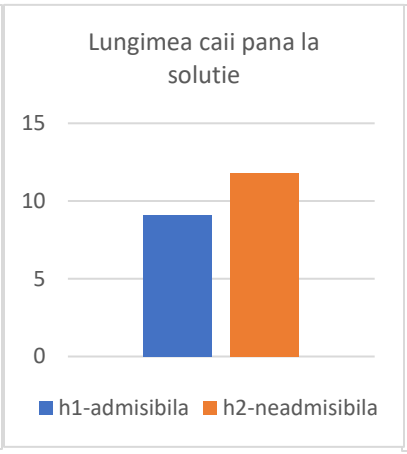


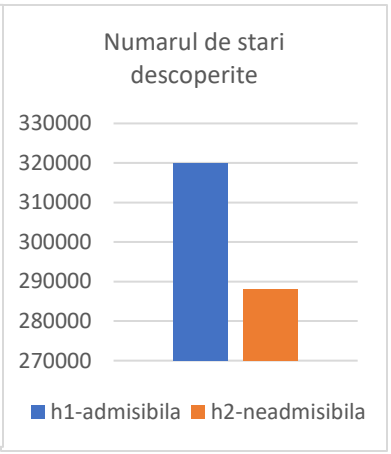


- Cube3, budget=20000, C=0.5



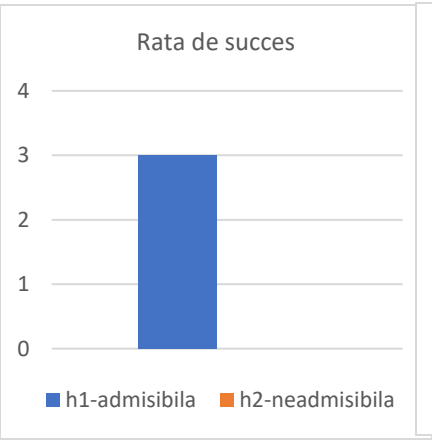


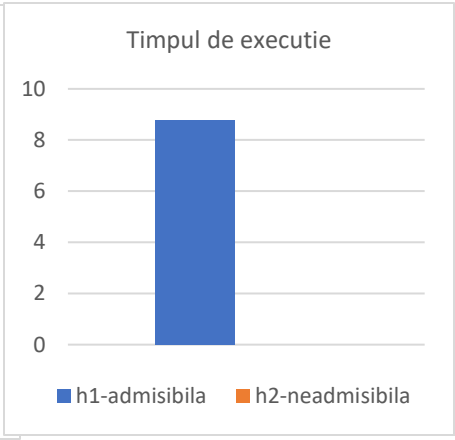


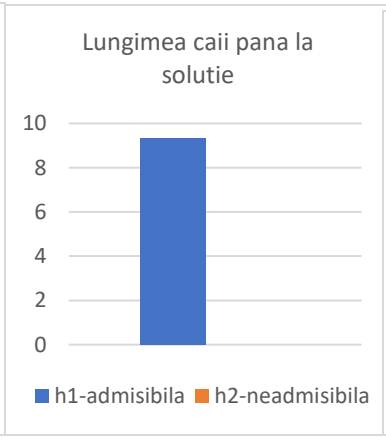


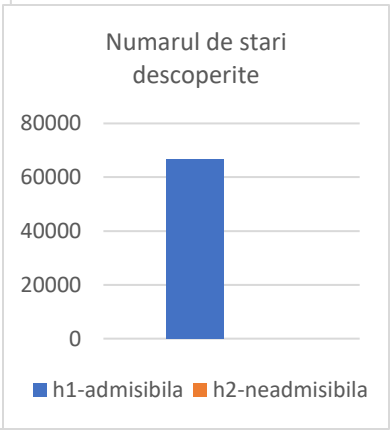
- Cube3, budget=1000, C=0.1 => 0 rata de success pentru ambele euristici

- Cube3, budget=5000, C=0.1

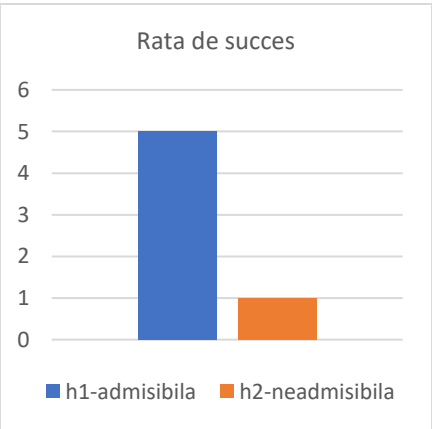


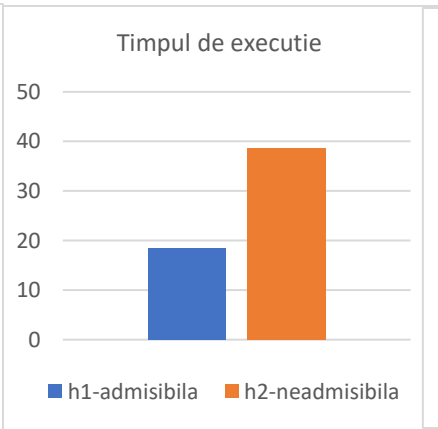


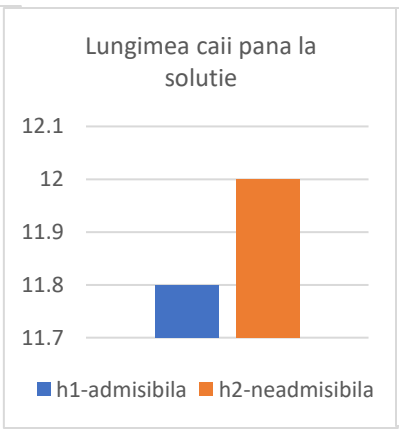


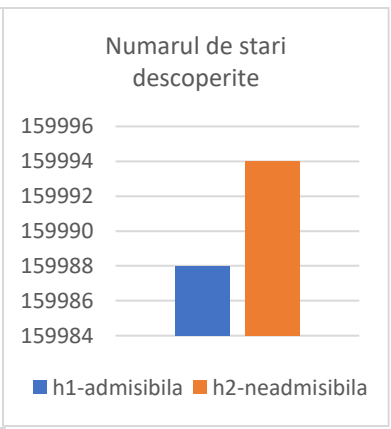


- Cube3, budget=10000, C=0.1

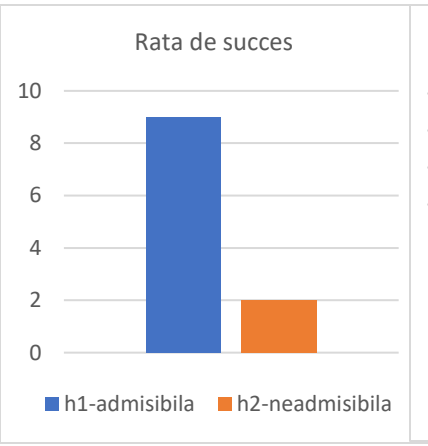


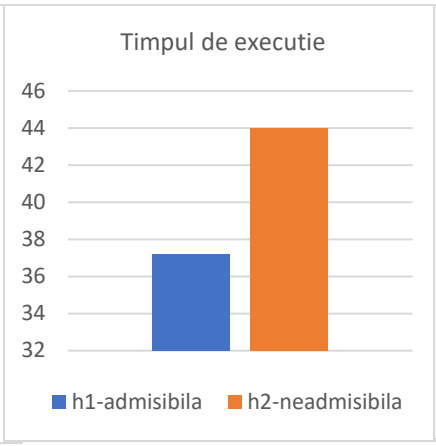


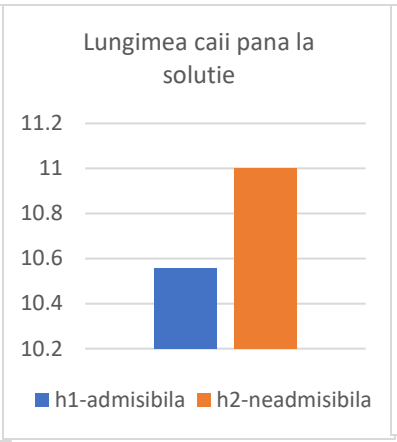


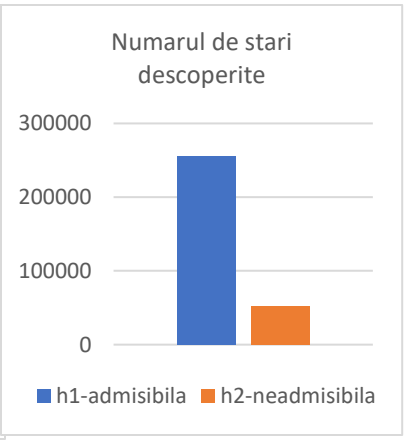


- Cube3, budget=20000, C=0.1







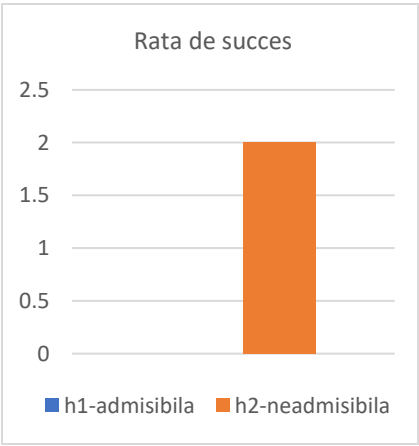


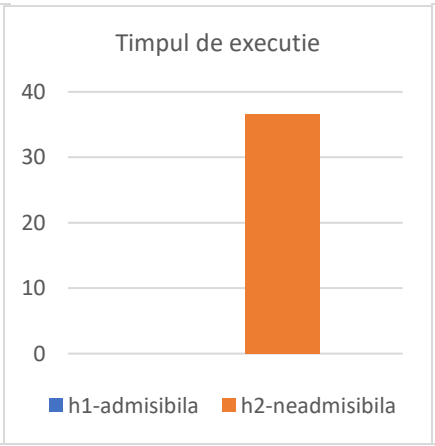
- Cube4, budget=1000, C=0.5 => 0 rata de success pentru ambele euristici

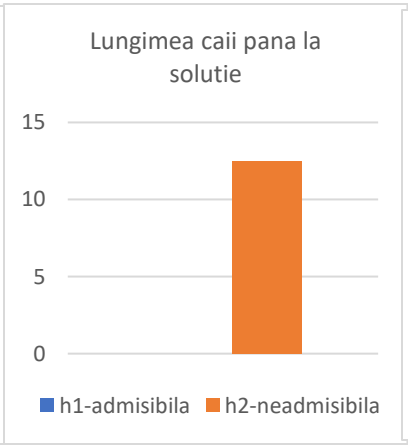
- Cube4, budget=5000, C=0.5 => 0 rata de success pentru ambele euristici

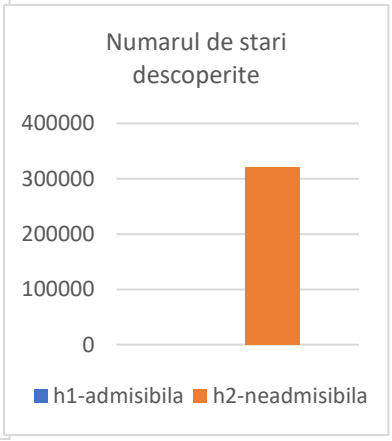
- Cube4, budget=10000, C=0.5 => 0 rata de success pentru ambele euristici

- Cube4, budget=20000, C=0.5









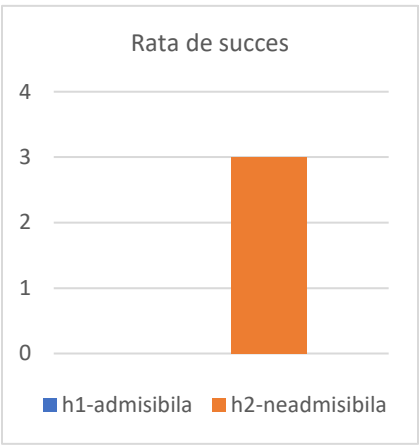
- Cube4, budget=1000, C=0.1 => 0 rata de success pentru ambele euristici

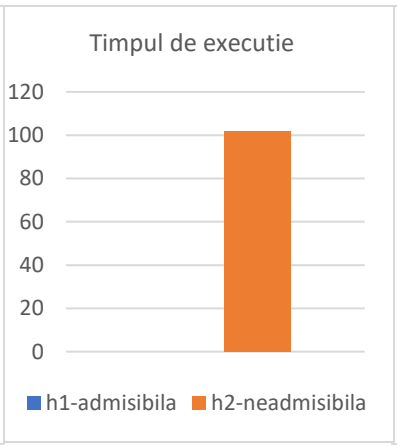
- Cube4, budget=1000, C=0.1 => 0 rata de success pentru ambele euristici

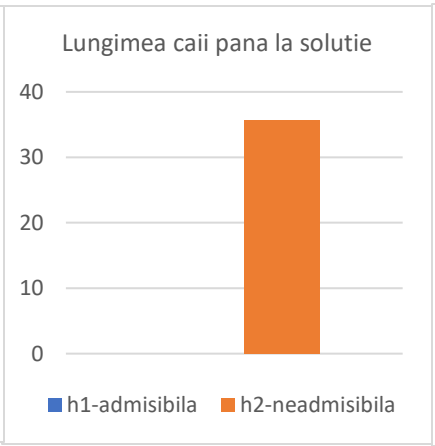
- Cube4, budget=5000, C=0.1 => 0 rata de success pentru ambele euristici

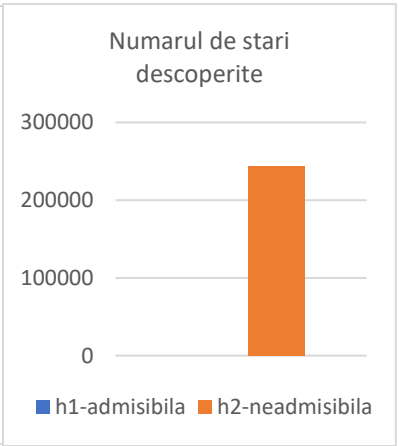
- Cube4, budget=10000, C=0.1 => 0 rata de success pentru ambele euristici

- Cube4, budget=20000, C=0.1





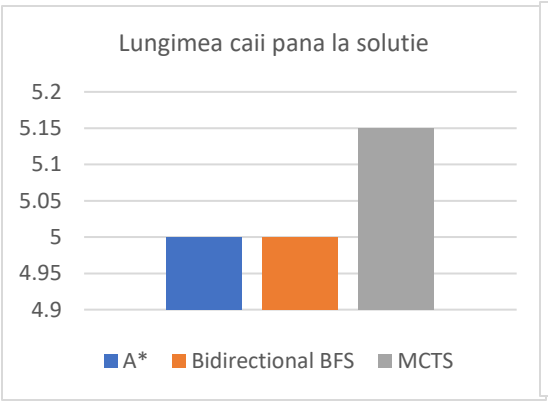


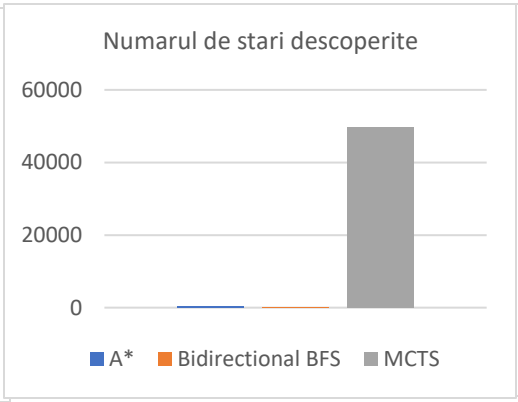


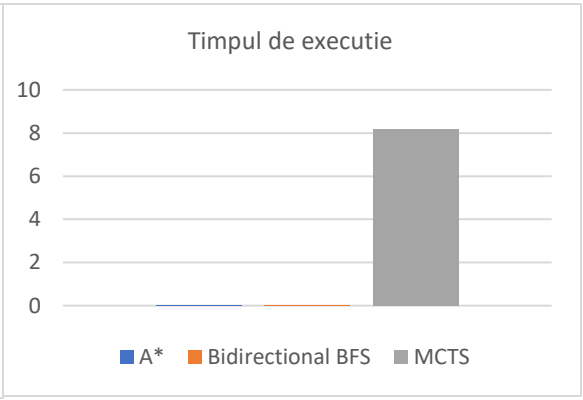
In urma comaratiei celor 2 euristic pe baza cazurilor de test date pot trage urmatoarele concluzii:

- Rata de succes pentru prima euristica este mai mare pentru bugete mai mici(1000-10000), dar cea de a 2a euristica tinde sa ofere constant solutii pentru bugete mai mari de 10000, indiferent de constanta C si complexitatea crescuta a cubului.
- In cazurile de test cu C=0.5, rezultatele au fost mai bune decat in cele cu C=0.1
- Prima euristica ofera rezultate mai rapide, iar numarul de stari exploatate este mai mic.
- In cazul unui test complex(testul 4) prima euristica nu ofera niciun rezultat indiferent de bugetul alocat, iar pentru un test simplu(testul1) aceata are rata de success de pana la 100%
- Cu cat bugetul e mai mare cu atat sansele de a primi un rezultat cresc, dar la fel si timpul si numarul starilor exploatate
- Solutia returnata de prima euristica este intotdeauna mai aproiata de cea optima, comparativ cu solutia celei de a 2a euristici care poate fi mult mai mare
- Cele mai bune rezultate overall pentru MCTS au fost:
    - Cube1: budget=5000, C=0.5, h1
    - Cube2: budget=20000, C=0.5, h1
    - Cube3: budget=20000, C=0.5, h1
    - Cube4: budget=20000, C=0.5, h2

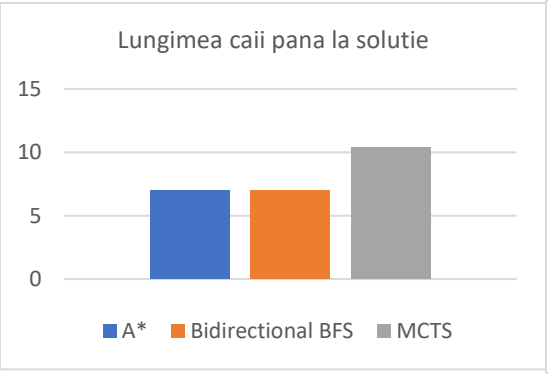
- Cube 1

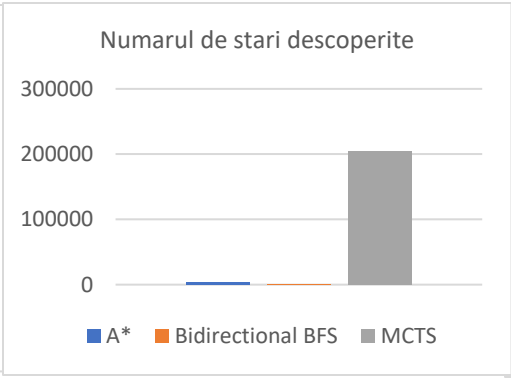


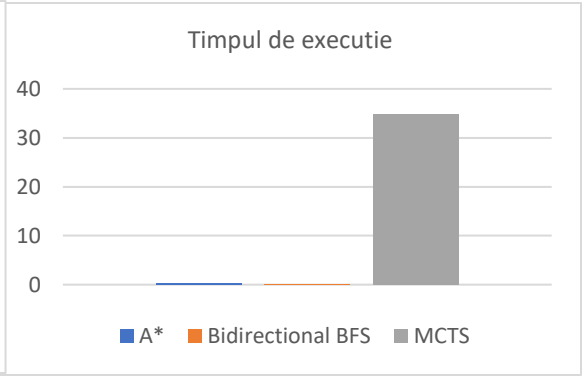




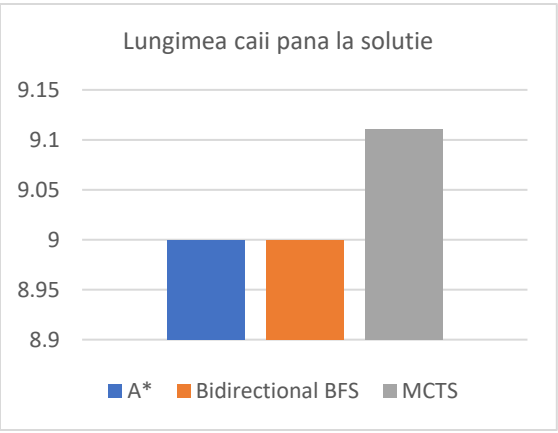
- Cube 2

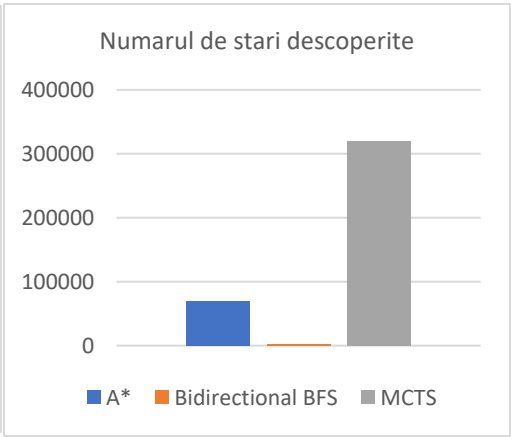


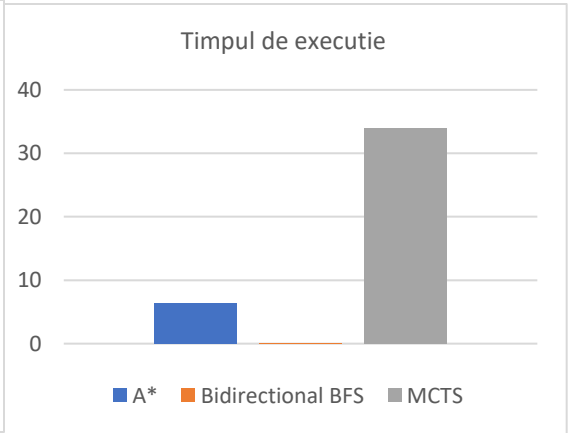




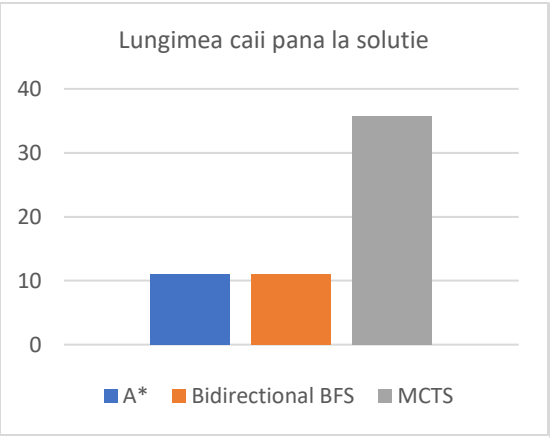
- Cube 3

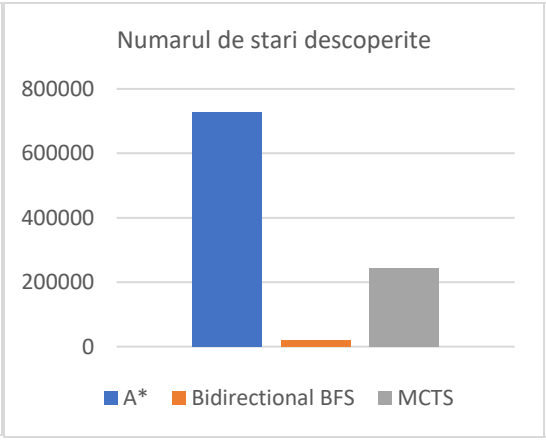


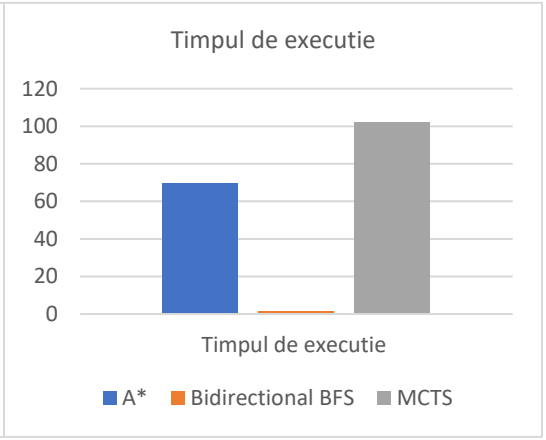




- Cube 4







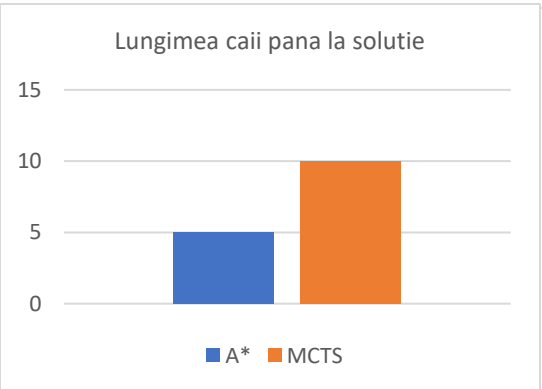
Prin comparatia A*, Bidirectional BFS si MCTS pentru cel mai bun C si budget pot trage urmatoarele concluzii:

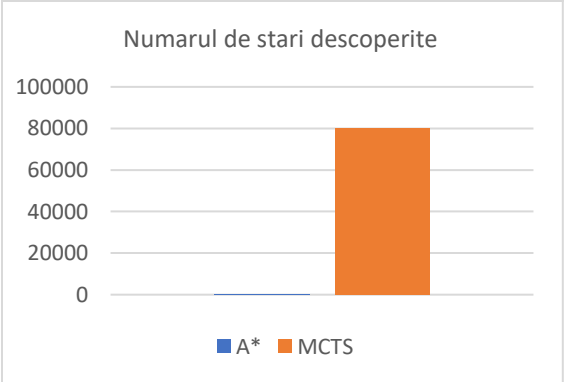
- Bidirectional BFS ramane cel mai performant algoritm din cele 3 din toate aspectele
- MCTS este cel mai costisitor din punct de Vedere al timpului si al numarului de stari descoperite, dar pentru un cub complex(testul 4) este asemanator A*.
- Lungimea caii pana la solutie in cazul MCTS nu este cea optima, ca in A* si bidirectional BFS
- Toate aceste comparratii au folosit doar valorille positive returnate de MCTS, acesta negasind intotdeauna un astfel de rezultat

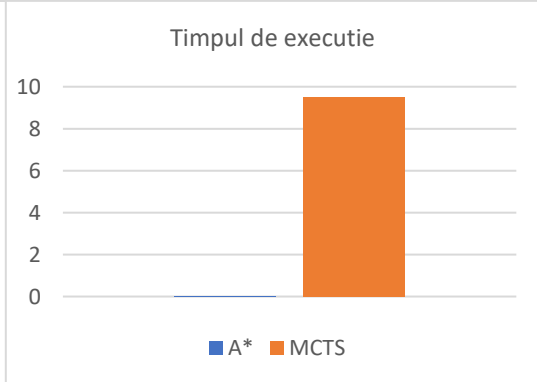
3.3. Pattern Database

Timpul de constructie al catalogului este de aproximativ 6.4519 s.

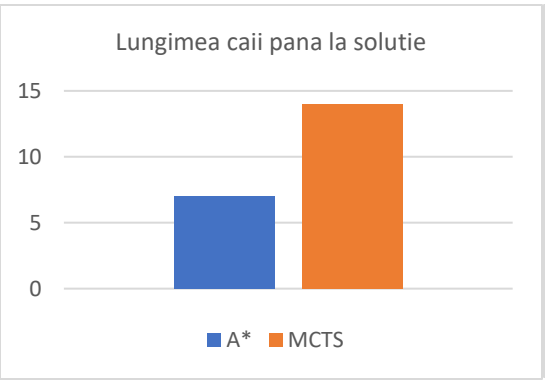
- Cube 1

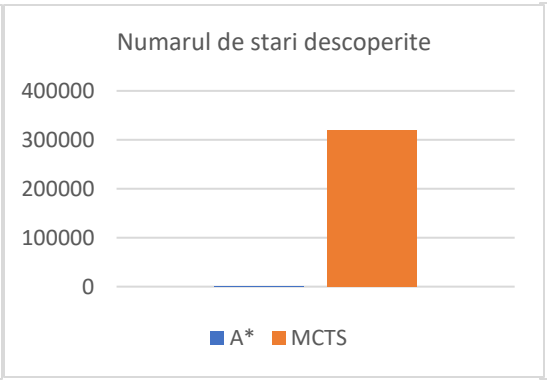


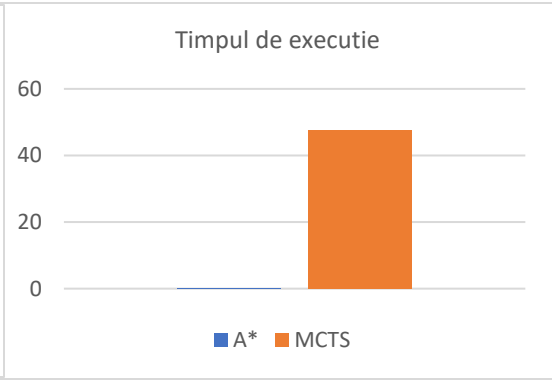




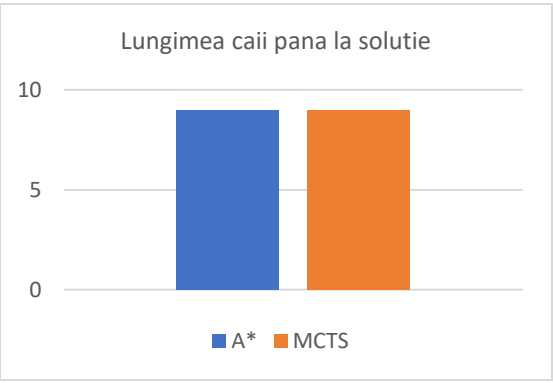
- Cube 2

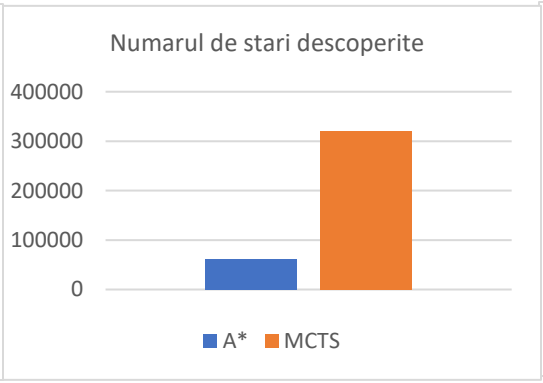


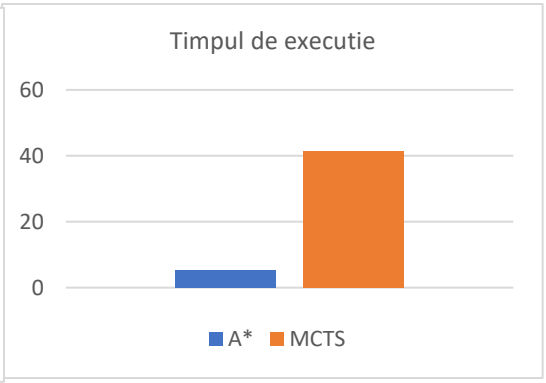




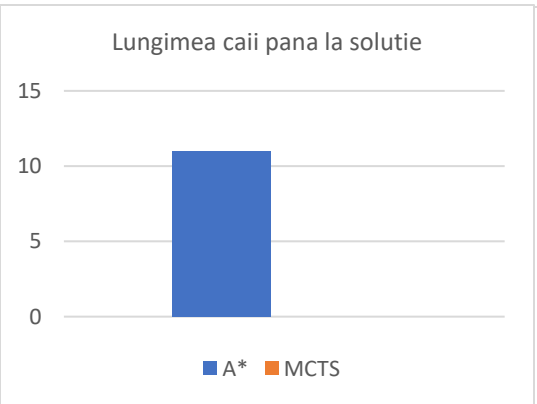
- Cube 3

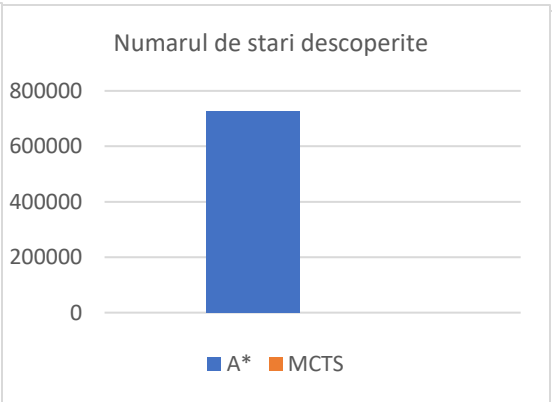


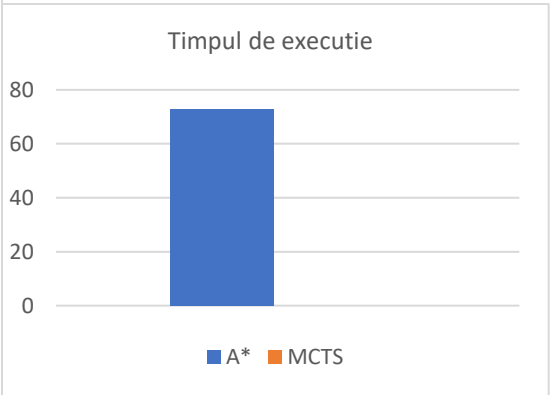




- Cube 4 – MCTS nu a gasit solutie







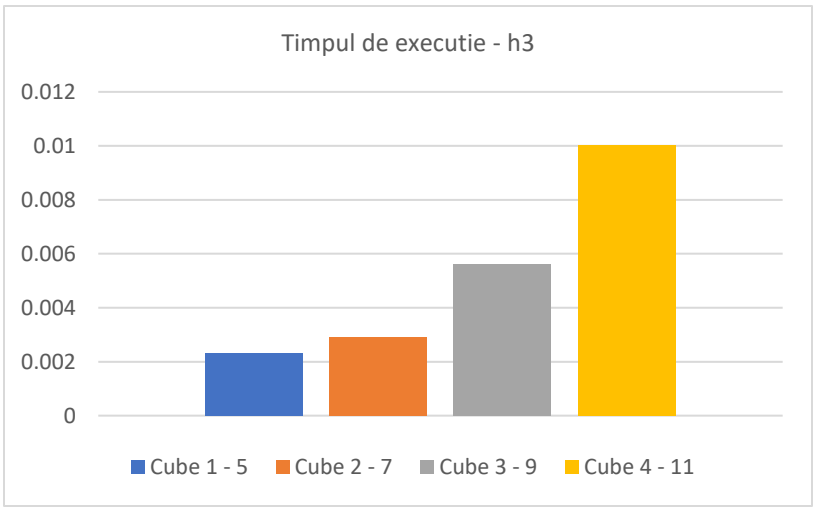
Prin comparatia A* cu MCTS folosind euristica h3 asa cum este descrisa in enunt performantele algoritmilor nu au fost semnificativ imbunatite fata de cazurile cu euristica simpla, insa daca am folosi o euristica care sa intoarca mereu valori mai mari decat 7 performantele ar fi evidente. Cu toate acestea, numarul de stari descoperite a scazut in cazul A*.

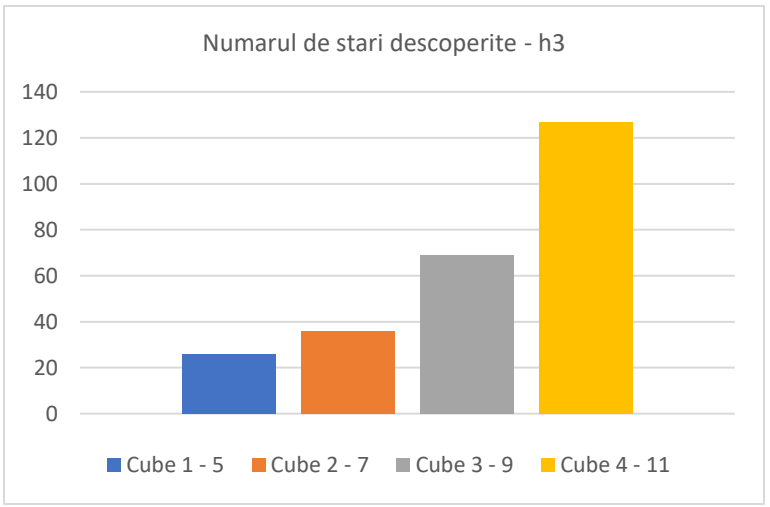
In toate cazurile de test A* este vizibil mai efficient decat MCTS, din toate punctele de vedere, mai ales ca la MCTS obtinerea unui rezultat dupa un timp mare de rulare nu este garantata.

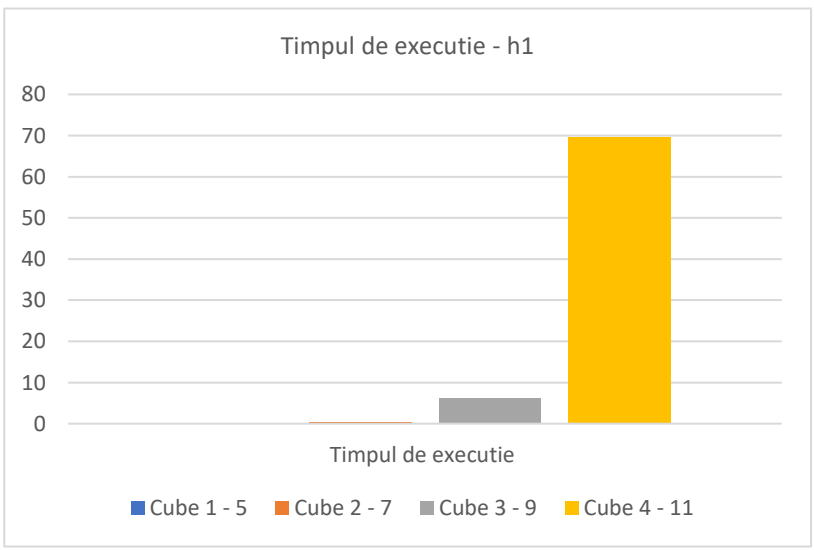
MCTS este destul de constanta pe cazurile de test, starile descoperite si timpul necesar necrescand asa de mult ca la A* odata cu complexitatea cubului ce trebuie rezolvat.

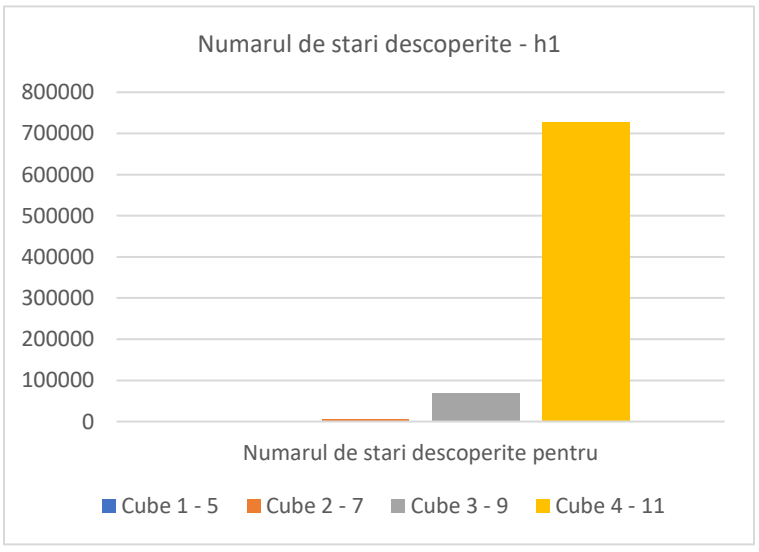
O propunere de euristica care trebuie folosita impreuna cu h3 pentru performante crescute, in loc de h1 este 7+h1.

- Comparatie A* h1 vs h3 - modificat

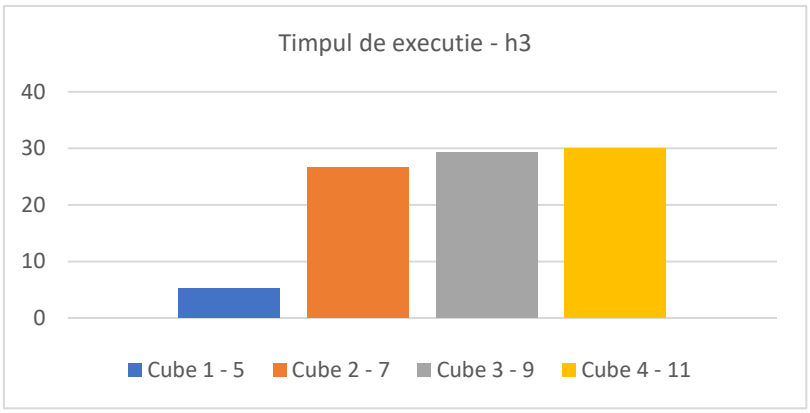


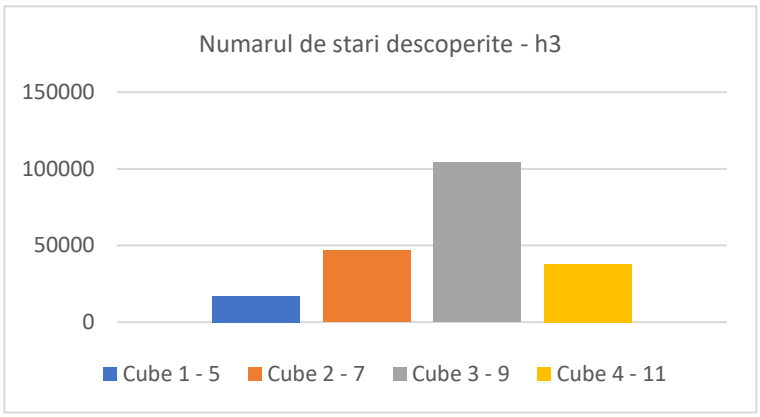


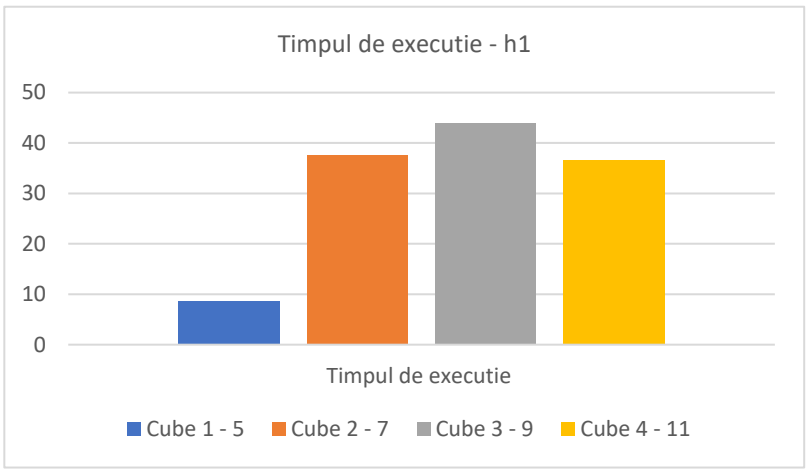


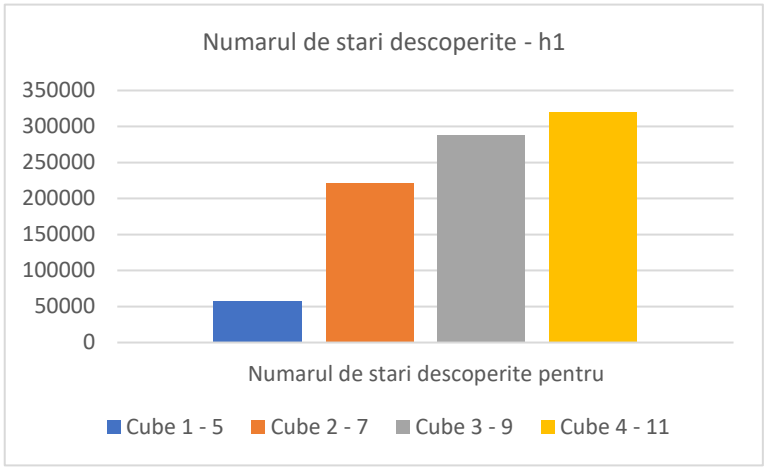


- Comparatie MCTS h1 vs h3 - modificat









Astfel cu o euristica care sa favorizeze pattern_database rezultatele obtinute pot fi foarte mult imbunatatite. In plus, success rate ul pentru MCTS a fost aproape 100% pentru fiecare caz testat.