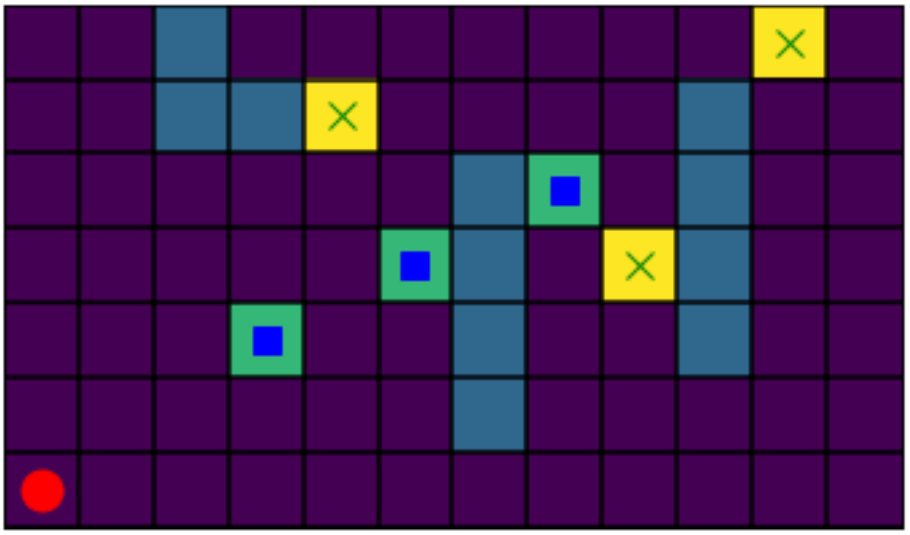
Compararea algoritmilor Beam Search şi IDA* pentru Sokoban



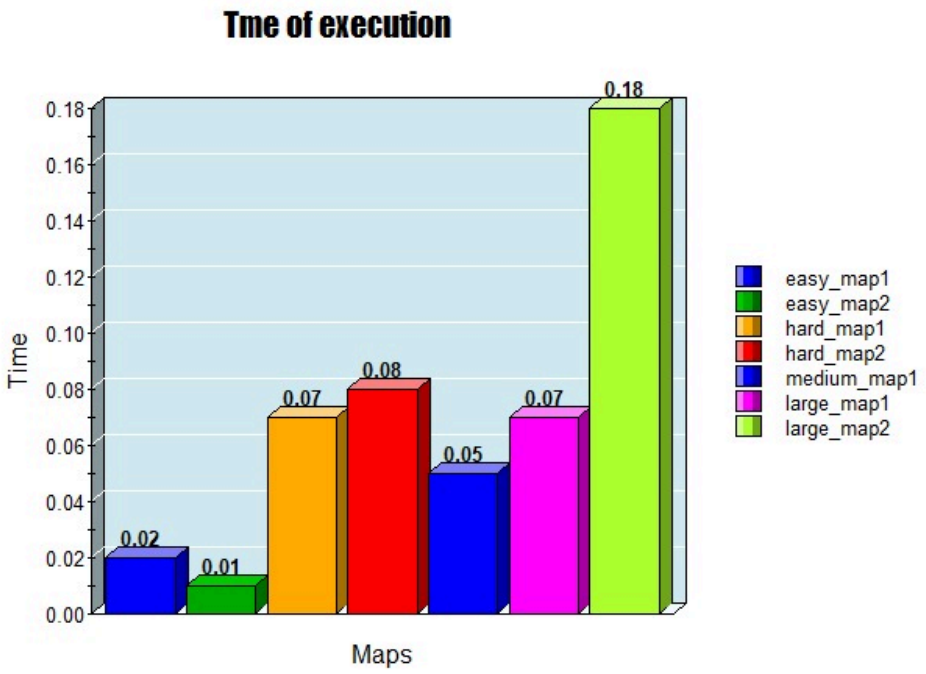
- **Restricţie “push only”**

Am decis să nu permit mişcări de “pull” pentru a păstra specificaţia standard a jocului, astfel am modificat functia is_valid_move din clasa Map pentru a-mi bloca miscarile de pull.

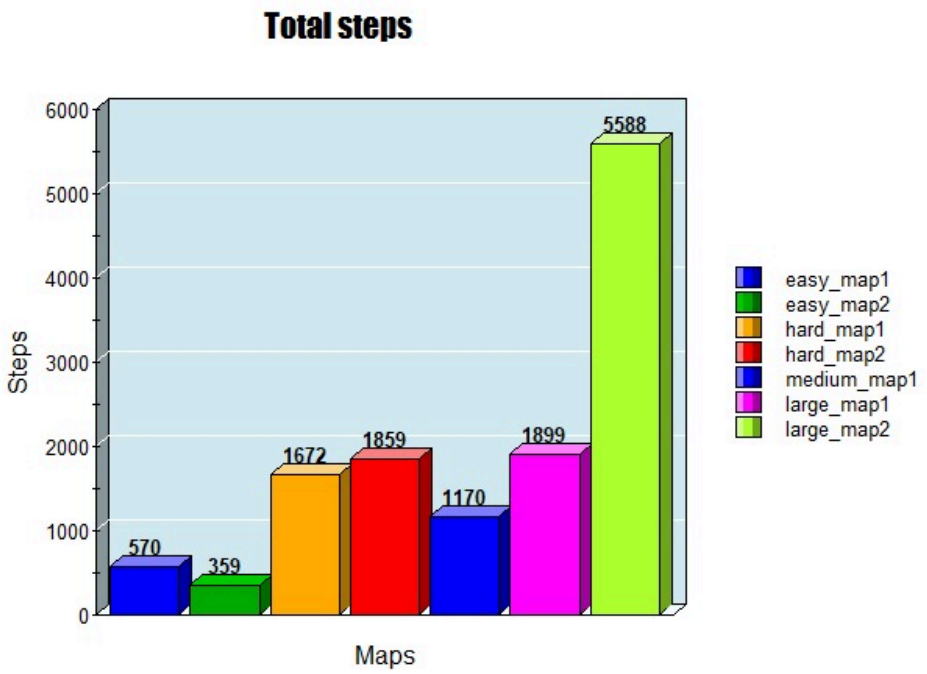
Euristica pentru Beam Search (`sokoban_heuristic`)

La început am încercat distanţa Euclidiană, însă jocul oscila între două stări fără a avansa. Am trecut apoi la Manhattan, dar fără matching beam-ul explora prea multe stări redundante. Introducerea matching-ului greedy a redus drastic numărul de stări explorate. Adăugarea detectării deadlock-urilor în euristică a tăiat timpul de rezolvare de la minute la sutimi de secunde.

Am implementat o euristică care construiește mai întâi o matrice a distanţelor Manhattan între fiecare cutie şi fiecare ţintă, apoi alocă greedy fiecărei cutii cea mai apropiată ţintă neocupată. În plus, detectez deadlock-urile clare (colţuri) şi adaug penalizări mari, iar blocajele parţiale le penalizez moderat. În final, includ jumătate din distanţa minimă jucător–cutie, pentru a reflecta costul mutării jucătorului.



Se poate observa din grafic că, pe măsură ce harta devine mai dificilă, timpul de execuție al Beam Search crește atât pentru că fiecare stare generează mult mai mulţi succesori pe care trebuie să-i evaluez cu euristica, cât şi pentru că trebuie să păstrez, sortez şi filtrez această mulţime la fiecare nivel (operații de heapq.nsmallest). În plus, soluţiile pe hărţi complexe sunt mai adânci, ceea ce înseamnă mai multe iteraţii ale buclei principale şi apeluri repetate ale funcţiei euristice, iar per total costul de calcul creşte semnificativ. Cel mai bine se poate observa din graficul de mai jos ce reprezinta totalitatea de stari explorate ale jucatorului si numarul de pasi prin care trece pentru a obtine solutia finala.

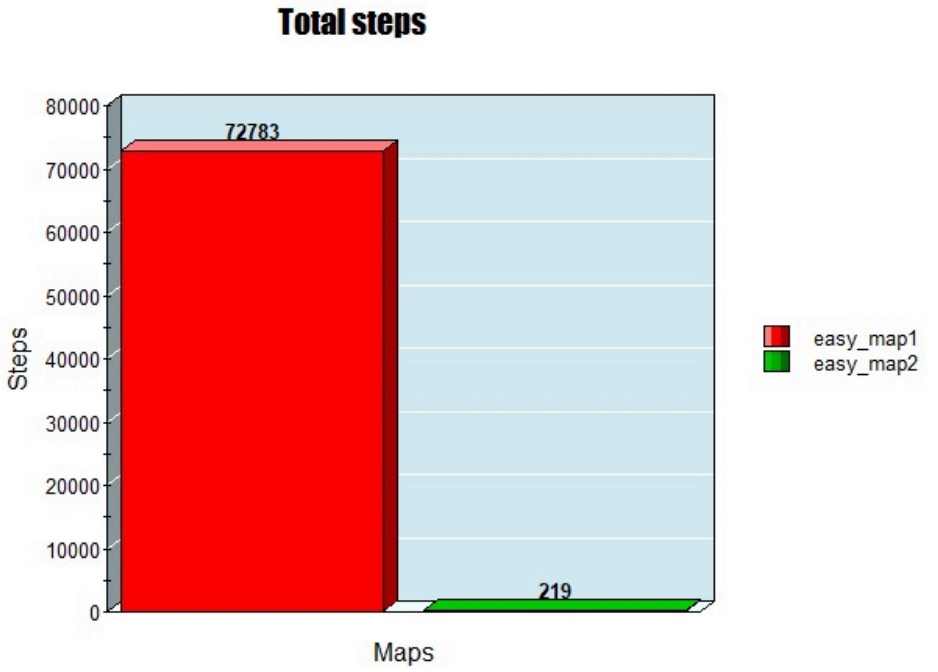


Euristica pentru IDA* (`sokoban_heuristic_ida`)

Am adaptat matching-ul greedy la un nivel global: scot mereu cel mai apropiat cuplu cutie–ţintă din listele rămase, ca să reduc complexitatea fiecărui apel. Penalizez deadlock-urile cu 300 de puncte (mai puţin agresiv decât în Beam) şi adaug întreaga distanţă minimă jucător–cutie pentru a surprinde costul deplasării în adâncimile DFS-ului iterativ.

La început am folosit o euristică bazată doar pe suma distanțelor Euclidiene între cutii și ținte, fără niciun mecanism de detectare a deadlock-urilor. Din cauza lipsei acestor penalizări, algoritmul explora foarte multe stări redundante și rula foarte mult fara sa ajunga la o solutie finala, până am înlocuit-o cu varianta Manhattan + matching + deadlock. Mai jos se pot observa graficele pentru timpul de executie si numarul de iteratii totalepentru fiecare harta in parte

Din pacate nu am reusit sa gasesc o euristica suficient de buna pentru a rezolva si celelalte mape, dar o voi face pe viitor cu siguranta



Tomita Ionut 334CD