Teme de casa BLIA 2003

Temele trebuie predate in cadrul laboratorului din ultima saptamana a semestrului (saptamana 26-30 mai). Temele sunt individuale (cu exceptia cazurilor unde se mentioneaza ca mai multe persoane pot colabora la rezolvarea temei) si fiecare student trebuie sa-si aleaga o tema si sa comunice subiectul ales in cadrul laboratorului din saptamana 7-11 aprilie.

Fiecare tema va fi punctata astfel:
5p – stil de programare: codul trebuie sa fie bine structurat, comentat, cu un stil de programare clara.
10p – respectarea cerintelor: pentru rezolvarea temei, se pot folosi ipoteze simplificatoare, dar acest lucru va duce la diminuarea punctajului.
5p – calitatea solutiei propuse.
5p – bonus pentru solutiile foarte bune.
Daca tema a fost copiata de la un coleg sau luata din orice alte surse externe (de ex. Web), punctajul va fi de 0p.

Solutiile propuse vor fi implementate intr-un limbaj de programare la alegere. Pentru orice intrebari sau nelamuriri legate de subiectul ales, precum si pentru obtinerea unor materiale ajutatoare (diverse articole publicate tratand problema respectiva), contactati persoana care a propus tema (mentionata pentru fiecare tema): Cosmin Carabelea (carabelea@cs.pub.ro) sau Cristian Popescu (blia@home.ro).

Subiecte:

1. Vacuum cleaner (Cosmin Carabelea)

Se considera mai multi agenti-roboti care trebuie sa curete o camera. Camera este reprezentata ca o matrice, in fiecare celula putandu-se afla un obstacol, gunoi, sau nimic. Un agent are o perceptie limitata care ii permite sa vada doar campurile din imediata sa vecinatate (la distanta de 1 de agent). Actiunile care sunt la dispozitia agentilor sunt: rotatie de 90 de grade la stanga sau la dreapta, miscare inainte cu o celula, curatare celula curenta daca exista gunoi..

Implementati aceasta problema in limbajul de programare dorit. Pentru simulare, considerati ca gunoiul apare periodic si aleator in camera. Pentru bonus: un agent trebuie sa decida in fiecare moment urmatoarea actiune de efecutat; propuneti si implementati mai multe modalitati de decizie ale agentilor si comparatile (care dintre agenti curata mai eficient).

2. The prey and the predators (Cosmin Carabelea)

Se considera un mediu reprezentat printr-o matrice. In acest mediu exista unul sau mai multi agenti numiti ‘prada’ si cativa agenti numiti ‘pradatori’ sau ‘vanatori’. Fiecare agent vede intreaga matrice si are la dispozitie una din urmatoarele actiuni: deplaseaza in sus/stanga/jos/dreapta sau sta-pe-loc. Scopul agentilor vanatori este sa incercuiasca un agent ‘prada’ (acesta sa nu mai aiba nici o mutare disponibila), in timp ce agentii prada trebuie sa evite acest lucru. Vanatorii sunt considerati cooperativi (au acelasi scop, de a incercui o prada), in timp ce prazile nu coopereaza.

Implementati in limbajul de programare dorit aceasta problema pentru un numar oarecare de agenti din fiecare tip. Exista mai multe feluri in care agentii vanatori pot lua decizii. Alegeti unul din urmatoarele exemple sau propuneti un altul: vanatorii pot comunica intre ei pentru a avea o strategie coerenta; un vanator alege o strategie de mutare pentru toti si o comunica si celorlalti; vanatorii nu comunica intre ei, dar observa comportamentul celorlalti si se adapteaza in consecinta; vanatorii actioneaza fara a rationa asupra miscarilor celorlalti. Tema poate fi rezolvata impreuna de 2 studenti cu conditia realizarii a cel putin doua tipuri de vanatori (care iau decizii in moduri diferite) si a comparatiei intre cele doua tipuri.

3. Multi-agent foraging (Cosmin Carabelea)

Se considera un mediu in care exista mai multi agenti care au scopul de a aduna mancare (sau minereu) si a o duce la baza lor. Mancarea se gaseste atat concentrata (in cantitati infinite) in anumite locuri numite *surse*, cat si raspandita (in cantitati f.mici – 5 unitati) aleator. Agentii au initial cunostiinte limitate despre mediu (nu cunosc decat pozitia bazei) si au o perceptie limitata (vad doar celulele adiacente), dar pot explora si memora harta. Actiunile posibile ale unui agent sunt: miscare la stanga/dreapta/sus/jos, sta-pe-loc, ridicare si lasare hrana (care poate fi lasata oriunde, nu numai la baza). Se presupune ca doi agenti nu pot ocupa aceeasi pozitie si ca exista si obstacole (deci agentii trebuie sa evite obstacolele). Fiecare agent are o capacitate de stocare a hranei limitata (de exemplu, 10 unitati).

Implementati in limbajul de programare dorit aceasta problema si testati ce se intampla pentru un numar relativ mic de agenti (2-5), cat si pentru un numar mare (aglomerare). Alegeti si justificati felul in care agentii se comporta (comunica sau nu prin mesaje cu ceilalti, lasa urme de mancare catre sursa de hrana gasita pentru a ii directiona si pe altii, ii ignora complet pe ceilalti, etc.). Bonus: puteti echipa agentii cu inca o actiune posibila: trecerea hranei de la un agent la altul aflat intr-o pozitie vecina si studia astfel aparitia de lanturi de agenti. Tema poate fi realizata de 2 studenti cu conditia realizarii a cel putin doua tipuri de agenti (moduri diferite de a lua decizii) si a comparatiei intre cel doua tipuri.

4. Agents in the block world (Cosmin Carabelea)

Se considera lumea blocurilor: pe o masa infinita se afla blocuri de mai multe tipuri. Exista mai multi agenti, fiecare echipat cu un brat de macara. Fiecare agent are un scop individual: acela de a realiza o anumita configuratie a blocurilor (de a aseza

unele blocuri peste altele), iar pentru aceasta, agentii fac planuri corespunzatoare (folosind de ex. STRIPS). Din nefericire, intre planurile facute individual de agenti pot exista conflicte (un agent muta un bloc care era dorit de un alt agent). Agentii trebuie sa fie capabili sa detecteze astfel de conflicte.

Se cere implementarea acestei probleme in limbajul de programare dorit. Evitarea conflictelor poate fi facuta, de exemplu, prin replanificare (daca un agent are nevoie de un bloc folosit de un alt agent, el poate replanifica si gasi un plan care nu include folosirea acelui bloc) sau pur si simplu prin acapararea blocurilor (daca un agent a folosit un bloc, acel bloc nu va mai putea fi folosit de un alt agent). Bonus (si subiect de 2 persoane): agentii comunica pentru a rezolva conflictele.

5. Problema labirintului (Cristian Popescu)

Se considera un labirint si mai multi agenti care imping fiecare cate o cutie. Agentii sunt cognitivi. Dispunerea agentilor in labirint este random. Cand toate cutiile au fost duse la semnul IESIRE, exista un supervizor care da semnalul de oprire al agentilor. Exista mai multe semne IESIRE in labirint. Cutiile sunt dipuse random in labirint. Fiecare agent isi cunoaste pozitia si stie locatia IESIRILOR.

Perceptiile posibile ale agentului: vede cutia doar daca se afla in celula in care se afla agentul, poate sa vada pe o raza de 2 celule in jur topologia labirintului. Daca un agent vrea sa se mute intr-o celula adiacenta si vede un alt agent ca sa afla intr-o celula adiacenta ei (celulei in care doreste sa se mute), face blocare pe celula respectiva (blocare de resursa). In caz in care-i reuseste blocarea celulei, se poate muta in celula respectiva. Miscarile posibile ale agentului: intoarce 90 de grade spre stanga, intoarce 90 de grade spre dreapta, mergi inainte, apuca cutie, lasa cutie jos in dreptul semnului IESIRE, fura cutie (acolo unde este cazul). Daca agentii sunt competitivi si un agent fara cutie se intalneste cu un agent care are cutie, primul agent va incerca sa i-o fure. Rezultatul actiunii sale va fi dat de o functie ce va alege aleator intre 0 si 1 (succes, insucces).

Evaluarea performantelor unui anget: pentru fiecare misca agentul primeste o penalizare de 1 punct, pentru gasire cutie un agent primeste 100 puncte, pentru pierdere cutie (acolo unde este cazul) agentul primeste penalizare 200 puncte, pentru depunere cutie la semnul IESIRE agentul primeste 5000 de puncte.

Se cere implementarea in limbajul de programare dorit a unuia din urmatoarele cazuri :

a. Agenti cooperanti si numarul de cutii este egal cu numarul de agenti.
b. Agenti competitivi si numarul de cutii este egal cu numarul de agenti.
c. Agenti cooperanti si numarul de cutii este mai mare decat numarul de agenti.
d. Agenti competitivi si numarul de cutii este mai mare decat numarul de agenti.
e. Agenti competitivi si numarul de cutii este mai mic decat numarul de agenti.

6. Problema satelitilor (Cristian Popescu)

Se considera un sistem de sateliti, un punct de emisie de pe Pamant si mai multe puncte de receptie. Satelitii trebuie sa se dispuna astfel incat sa asigure receptia acelui punct de emisie precum si transferul optim catre punctele de receptie de pe Pamant. Punctul de emisie se misca la fel ca si punctele de receptie de pe Pamant (sau pot fi fixe). Numarul de sateliti este fix (de fapt ar trebui sa fie minim).

Pozitia fiecarui agent este initial random. Pentru fiecare miscare satelitul pierde 1 punct. Satelitii se misca 90 de grade la stanga, 90 de grade la dreapta, inainte, inapoi. Fiecare receptie de semnal de la sursa de pe Pamant va oferi agentului 50 de puncte. Fiecare receptie de semnal de la un alt satelit va oferi satelitului 5 puncte. Fiecare emisie de semnal catre un punct de receptie de pe Pamant ofera satelitului 50 puncte. Fiecare emisie de semnal cate un alt satelit, ofera agentului 20 de puncte.

Un satelit adiacent este vizibil daca se afla pe o raza de 5 celule de agentul nostru. Satelitul nostru cunoaste pozitiile satelitilor adiacenti lui de pe o raza de 5 celule. In transmiterea semnalului este specificata destinatia semnalului (pozitia receptorului de pe Pamant astfel incat se cauta satelitul cel mai apropiat de acesta. Daca receptorul se afla pe o raza de 5 celule in jurul satelitului nostru care a primit semnalul, acesta nu mai transmite altui satelit semnalul ci transmite direct acestuia semnalul. Daca nu se gaseste nici un satelit care sa fie in raza de 5 celule ale receptorului, atunci satelitul cel mai apropiat va face "un drum", abatandu-se de la pozitia initiala pana va intra in raza de 5 celule a receptorului dorit de pe Pamant, pentru a transmite semnalul, iar apoi va ramane acolo.

Daca in jurul punctului de emisie de pe Pamant nu se afla nici un satelit, atunci cand acesta va emite (dupa o distributie random, adica pentru intervalul [0..1000], se va emite de pe Pamant doar daca functia random intoarce un numar prim) toti satelitii primesc penalizare 25 de puncte. In acest caz satelitii incep sa se rearanjeze. Exista posibilitatea ca doi sateliti sa incerce sa ocupe aceeasi celula. In acest caz ambii sateliti primesc penalizare 200 de puncte si se incearca alta organizare a lor.

Se cere implementarea acestei probleme in limbajul de programare dorit.

7. SETI (Cristian Popescu)

Se considera un sistem de antene parabolice pentru studierea boltei stelare si pentru cautarea vre-unui eventual semnal extraterestru (vezi SETI). Fiecare antena este un agent. Agentii trebuie sa conlucreze pentru a prelucra cat mai multa informatie precum sunt si in competitie pentru a descoperi viata extraterestra (primeste o excursie in Bahamas).

Pozitia fiecarui agent este initial random. Pentru fiecare miscare antena pierde 1 punct. Agentul se misca 30 de grade la stanga, 30 de grade la dreapta, 15 grade in sus, 15 grade in jos. Pozitia antenelor este random.

Fiecare agent vede (in directia in care este indreptata antena) pe o raza de 5 celule. In celulele "vizibile" de catre antena apar diferite semnale (o functie random in intervalul [0..1000]). Aparitia unui semnal extraterestru intr-o celula, insemana ca functia random intoarce una dintre valorile din multimea {54, 358, 459, 670, 890, 915}. Daca semnalul extraterestru este pe o raza de 2 celule, atunci se considera ca receptia este buna, si agentul castiga 1000 de puncte. Altfel, agentul are nevoie de

ajutoare. Daca pe o raza de 10 celule in jurul agentului se afla alti agenti, atunci agentul nostru "cere ajutor" agentilor adiacenti. Agentii adiacenti trebuie sa se orienteze cate semnalul extraterestru care va dura un timp random. Daca exista cel putin inca un agent care prinde semnalul, cel de-al doilea agent primeste 400 de puncte. Agentul care a descoperit semnalul primeste 600 de puncte. Daca este pierdut, agentul care l-a descoperit pierde 10 puncte, iar agentii adiacenti (pe raza de 10 celule in jurul agentului nostru) pierd 15 puncte.

Se cere implementarea acestei probleme in limbajul de programare dorit. Antenele se vor misca astfel inca sa asigure o cat mai buna acoperire a zonei de cercetat. Toata simularea dureaza un anumit timp (il stabiliti voi). Cand acest timp a expirat un supervizor le transmite agentilor sa se opreasca.

8. "Orbul si ciungul" (Cristian Popescu)

Se considera ca avem un agent care este orb, dar aude si vorbeste si un alt agent care este ciung de ambele picioare, vede dar este surd. Orbul vrea sa ajunga in Braila, iar ciungul in Bacau. Alti eventuali agenti care pot sa apara sunt eventualii drumeti care se intalnesc cu cei doi (distributia este random, adica pentru intervalul [0..1000], grupul se intalneste cu un drumet doar daca functia random intoarce un numar prim).

Cei doi agenti trebuie sa conlucreze dar sunt si-n competitie (fiecare doreste sa ajunga in alt orash, dar nu pot fara ajutorul celuilalt). Castiga agentul care ajunge in orasul dorit. Daca grupul nostru se intalneste cu un alt agent, o functie random care intoarce 1 sau 0 il va determina pe ciung sa-i spuna sau nu orbului ca este cineva in apropiere. Daca orbul afla ca este cineva in apropiere va cere acestuia informatii cum sa ajunga in Braila (dumul de cost minim intre pozitia la care se afla grupul si orasul Braila). Orbul va apuca (si nu va asculta sfaturile ciungului) pe drumul spre Braila pentru urmatoarele 3 mutari. Apoi revine la "dirijarea" ciungului, care-l va orienta catre Bacau. Deci orbul aude si vorbeste, iar ciungul vede si vorbeste, dar nu aude.

Miscarile posibile ale grupului: intoarce 90 de grade spre stanga, intoarce 90 de grade spre dreapta, mergi inainte. Evaluarea performantelor unui anget: pentru fiecare miscare dirijata de ciung, ciungul va primi 1 punct iar orbul va primi 1 punct penalizare. De fiecare data cand orbul vorbeste cu un drumet, ciungul pierde 5 puncte, iar orbul castiga 5 puncte. De fiecare data cand orbul merge fara a-l asculta pe ciung, orbul primeste 1 punct, iar ciungul va pierde 1 punct. Cand se ajunge in orasul dorit, agentul primeste 1000 puncte iar celalalt agent pierde 1000 puncte.

Se cere implementarea acestei probleme in limbajul de programare dorit. Harta intre orasele Braila si Bacau se va considera un graf neorientat. Pozitia initiala a grupului va fi aleasa random.