Budapesti Műszaki és Gazdaságtudományi Egyetem

Intelligens elosztott rendszerek ([BMEVIMIAC02](https://edu.vik.bme.hu/course/view.php?id=4271))

Házi feladat dokumentáció

Morvay Balázs Tibor (CLT8ZP)

Csikós Marcell (P78P08)

2020. május 28.

**A feladat leírása**

A feladat során egy autóból és töltőkből álló többágenses rendszert valósítottunk meg. Az autónak a kezdeti helyzetéből egy megadott célhoz kell eljutnia. A környezetben ezeken kívül különböző helyeken különböző hatékonyságú töltők is találhatók.

Induláskor az autó felkéri a töltőket, hogy licitáljanak, hogy milyen gyorsan jut el az autó rajtuk keresztül a célhoz. Ebbe beleszámít az út hossza az autótól a töltőig, az út hossza a töltőtől a célig, illetve hogy mennyi idő alatt tölt fel az autó annál a töltőnél. Az autó megvárja, amíg be nem érkezik minden licit, majd eldönti, melyiket válassza. Nincs érvényes licitje annak a töltőnek, amelyhez nem tud eljutni az autó a kezdeti töltöttséggel, vagy ha nem tud onnan a célhoz eljutni az autó maximális töltöttséggel.

Ha egyik töltő licitje se érvényes, az autó egy helyben marad, így nem tud útközben leállni. Fontos megjegyezni azt is, hogy az autó csakis egy töltőnél fog megállni, se nem többnél, se nem kevesebbnél.

**A megoldás összefoglalása**

***Ágensek felsorolása és feladatuk***

* car: feladata a töltők felkérésre licitálásra, illetve kiválasztani a győztest, végül pedig mozgás a megfelelő helyekre
* charger (1,2,3,4): feladatuk licit számítása és küldése az autó számára és az autó töltése

***Ágensek kapcsolata***

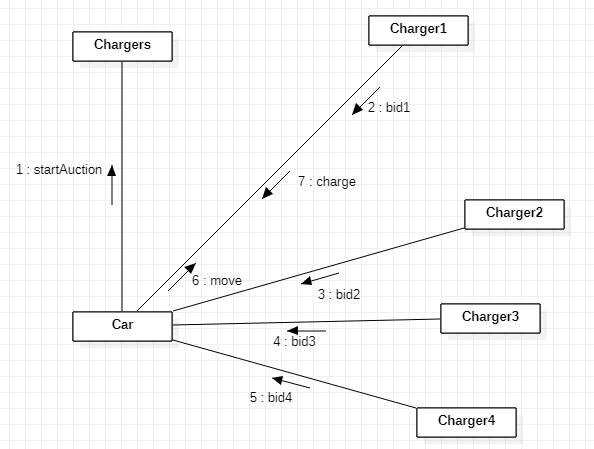
Az ágensek ismerik egymás pozícióját, az autó felhasználja minden töltőét, a töltő pedig csak az autóét. Az autó egységesen kommunikál minden töltővel. A töltők is kommunikálnak ezután: először elkérik az autó helyzetét, majd elküldik neki a licitet. A győztes megtalálása után az autó már csak a kiválasztott töltővel fog kommunikálni, azzal, hogy lekéri a helyzetét, majd töltést indít ott.

***Ágensek viselkedése***

Az autó kezdeti tudástára a helyzete. Működése kezdetén aukciót kezdeményez a töltők között. Ezután várakozó állapotba kerül, amíg minden licit be nem érkezik a töltők felől. Ezt követően kiszámítja a győztest, azaz a legalacsonyabb licitet. Ennek megfelelően olyan tervet választ, amelyben a kiválasztott töltőt akarja elérni. Amikor odaér, és még nem 100 a töltöttsége, a terv nem változik, csak töltést kezdeményez. Amikor a töltőnél van és 100 lesz a töltöttség, akkor jön a célhoz érés terve. Befejeződik az ágens működése, ahogy eléri a célt.

A töltő kezdetben várakozik arra, hogy az autó engedélyezze a licitálást. Ezután lekéri az autó helyzetét, és kiszámolja a licitet annak, a saját helyzetnek és a saját töltési teljesítménynek a függvényében. Miután ez megvan, végez a kommunikációval.

***Többágenses rendszer összefoglaló ábrája***



Az ábrán annak a helyzetnek a kommunikációját látni, amikor a második töltő nyeri meg az aukciót.

**A fejlesztés összefoglalása**

A fejlesztéshez 2.4-es Jason verziót használtunk.

ASL szinten lett megvalósítva az ágensek viselkedése különböző helyzetekben. Autónak itt van megcsinálva, hogy várjon a licitekre, a győztes kiválasztása, illetve az is, hogy arra mozogjon, amerre kell. A töltők ASL szinten nem csinálnak ennyi mindent: megvárják, amíg lehet licitálni, majd beküldik a licitet. Mindkét féle ágensnél jelen van a helyzetlekérdezés is, ami fontos egyes függvényekhez.

Java szinten lett megvalósítva többek között a grafikus megjelenítés: GridWorldModel és GridWorldView használatával történik. Ebből kifolyólag az ágensek kirajzolása is itt történik. Ezen kívül több, ágensek által használt függvény vagy ezekhez tartozó segédfüggvény is Java szinten lett megírva: licitszámítás, győztes licit megtalálása, mozgás, távolságszámítás.

**A kifejlesztett program ismertetése**

A környezet leírásával kezdem az ismertetőt.

A program fix paraméterei:

* gridSize: Azt adja meg, hogy a pálya oldalhossza hány egységnyi legyen. Értéke 30.
* auction: Azt jelzi, hogy elindult-e már az aukció a töltők között. Alapértelmezett értéke 0, autó ágens 1-re állítja.
* bids: A töltőágensek licitjeinek tömbje. Alapból minden bejegyzés 0 benne, a töltőágensek módosítják.
* winner: Azt mutatja, hogy van-e győztes, és ha van, melyik töltő az. Alapértelmezett értéke 0, autó ágens módosítja.

A program változtatható paraméterei:

* consumption: Azt adja meg, hogy az autó mennyit veszít töltöttségéből egy egységnyi mozgás során.
* chargeLeft: Az autó kezdeti töltöttségét változtathatjuk vele. Maximum 100 lehet.
* chargeRates: A töltőállomások teljesítményét tárolja. A teljesítmény azt jelenti, hogy egy lépés alatt mennyit növeli az autó töltöttségét.
* destination: A cél koordinátái.
* auto: Az autóágens kezdeti koordinátái.
* c1…c4: A töltők koordinátái.

Az init függvény betölti a WorldView-t és WorldModel-t, melyek a rendszert írják le. Az executeAction arra való, hogy az ágensek által kiadott parancsokat átalakítsa a környezetben definiált függvényekké.

Az updatePerceptsnek meg kell hívódnia minden alkalommal, amikor változik bármely ágens helyzete, tulajdonsága, vagy valamelyik általuk változtatott paraméter.

A WorldModelben vannak definiálva a függvények, melyek a rendszer működéséhez kellenek.

* moveTowards: A függvény az autó helyzetét mozdítja a paraméterben megadott (x,y) hely irányába, és csökkenti a töltöttségét is. Az autó átlós szomszédra is tud mozdulni. A mozgás után az autó mellett a töltőket is újra kell rajzolni.
* chargeCar: A függvény a paraméterben adott számú töltő teljesítményével növeli az autó töltöttségét. Ha a töltöttség így 100-nál magasabb lenne, 100-ra állítja az értéket. Végül újrarajzolja az autót az új töltöttséggel.
* bid: Egy töltőnek a licitjét számolja ki. Az autó koordinátáit paraméterben kapja meg, a töltőét megkeresi szintén paraméter alapján. Először kiszámolja, hogy ha az autó eljut a töltőhöz, akkor mennyi ideig tartana a feltöltése. Ehhez először ki kell számolni a csökkentett töltöttséget, azaz amennyi lenne, ha odamenne. A licit három komponensből adódik össze: az autó és töltő távolsága + az imént kiszámolt töltési idő + a töltő és a cél távolsága. Az érvénytelen licit értéke 9999. Ez akkor történik meg, ha az autó az eredeti helyzetéből nem tud odaérni a töltőhöz, vagy ha a teljesen feltöltött autó nem tud odaérni a célhoz. Akkor is érvénytelen lesz a licit, ha a töltő éppen nem működik, azaz teljesítménye 0.
* startAuction: Beállítja az auction változót, így a töltők már tudnak licitálni.
* decideWinner: A licitek tömbje alapján megkeresi a legalacsonyabb licitet, és azt beállítja győztesnek. Holtverseny esetén a számszerűleg előbb lévő töltő nyer. Akkor nincs nyertes, amikor minden töltő licitje érvénytelen lesz.
* calculateDistance: Segédfüggvény a licitszámításhoz. A paraméterben megadott (x1,y1) és (x2,y2) pontok közötti utat számítja ki. Ugyanúgy működik, mint a mozgást megvalósító függvény, de csak növeli a távolságot, amíg ugyanott nem lesz a két pont.

A WorldView felel a pálya és az ágensek kirajzolásáért. Maga a rendszer egy GridWorldView, azaz négyzetrácsos megjelenésű.

Az autó egy kék körként van szimbolizálva, rá van írva, hogy Car, illetve a töltöttségi szintje. A cél egy kék négyzet. A töltők zöld körök a rendszerben, rájuk van írva, hogy Charger.