A képen szöveg, Betűtípus, tipográfia látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

**1.1. Emberi módon cselekedni: Turing-teszt megközelítés**

**Turing-teszt:** A számítógép akkor állja ki a próbát, ha az emberi kérdező néhány írásos kérdés feltevése után nem képes eldönteni, hogy az írásos válaszok egy embertől vagy egy géptől érkeznek-e.

Szükséges:

* **természetes nyelvfeldolgozás:** a sikeres angol nyelvű párbeszédhez
* **tudásreprezentáció:** az ismert vagy hallott információ tárolására
* **automatizált következtetés:** a tárolt információt kérdések megválaszolására és új következtetések levonására használjuk
* **gépi tanulás:** az új körülményekhez való adaptálódáshoz, a mintázatok detektálására és általánosítására

A Turing-teszt a kérdező és a számítógép közötti fizikai kölcsönhatást szándékosan kerülte.

Szükséges:

* **gépi látás**: az objektumok érzékeléséhez és
* **robotika**: az objektumok mozgatásához.

**1.2. Emberi módon gondolkodni: a kognitív modellezés**

Ha azt szeretnénk kijelenteni, hogy egy program emberi módon gondolkodik, valamilyen módon meg kellene határoznunk, hogy az emberek hogyan gondolkodnak. Ehhez az emberi elme működési mechanizmusának belsejébe kellene tudnunk belenézni.

Az MI-kutatás kezdeti szakaszában a megközelítéseket sokszor össze is keverték. Egy szerző állíthatta például, hogy ha egy algoritmus egy feladaton jól vizsgázik, akkor jó modellje az emberi képességeknek, és megfordítva.

**1.3. Racionálisan gondolkodni: a gondolkodás törvénye**

Arisztotelész görög filozófus volt az elsők egyike, aki megkísérelte a „helyes gondolkodás”, azaz a megcáfolhatatlan következtetési folyamatok törvényekbe foglalását. Az MI-n belül uralkodó logicista (logicist) hagyomány azt reméli, hogy ilyen programokra alapozva intelligens rendszereket képes létrehozni.

**1.4. Racionálisan cselekedni: a racionális ágens**

Egy ágens (agent) nem más, mint valami, ami cselekszik. Számítógépes ágensektől azonban elvárjuk, hogy legyenek más jellemzői is, amelyekben különböznek a „mezei” programoktól. Ilyen jellemzők például az autonóm vezérlés felügyelte cselekvés, a környezet észlelése. Egy racionális ágens (rational agent) a legjobb kimenetel érdekében vagy – bizonytalanság jelenlétében – a legjobb várható kimenetel érdekében cselekszik.

A képen szöveg, Betűtípus, fehér, képernyőkép látható

Automatikusan generált leírás

**Állapottér:**

A kezdeti állapot és az állapotátmenet-függvény együttesen implicit módon definiálják a probléma állapotterét (state space): azon állapotok halmazát, amelyek a kiinduló állapotból elérhetők. Az állapottér egy gráfot alkot, amelynek csomópontjai az állapotok és a csomópontok közötti élek a cselekvések. Az állapottér egy útja (path) az állapotok egy sorozata, amely állapotokat a cselekvések egy sorozata köt össze.

A **célteszt** (goal test), amely meghatározza, hogy egy adott állapot célállapot-e. Néha létezik a lehetséges célállapotok egy explicit halmaza, és a teszt egyszerűen megnézi, hogy az ágens elérte-e ezek egyikét.

Egy **útköltség**- (path cost) függvény, amely minden úthoz hozzárendel egy költséget. A problémamegoldó ágens azt a költségfüggvényt fogja választani, amely a saját hatékonysági mértékének felel meg

A probléma megoldása (solution) nem más, mint a kiinduló állapotból a célállapotba vezető út.

A képen szöveg, Betűtípus, fehér, tipográfia látható

Automatikusan generált leírás

**3. Megoldások keresése:**

A megoldás megkeresése – az állapottérben végrehajtott kereséssel történik. A keresési fa gyökere az a **keresési csomópont (search node)**, amely a kezdeti állapotnak felel meg.

A képen szöveg, képernyőkép, Betűtípus, algebra látható

Automatikusan generált leírás

A képen szöveg, Betűtípus, sor, képernyőkép látható

Automatikusan generált leírás

**3. Lokális kereső algoritmusok és optimalizációs problémák**

A lokális keresési algoritmusok (local search) csak egy aktuális állapotot (current state) vesznek figyelembe (a többszörös utak helyett) és általában csak ennek az állapotnak a szomszédjaira lépnek tovább. A keresés által követett utat tipikusan nem is tárolják el. Bár a lokális keresési algoritmusok nem szisztematikusak, két kulcsfontosságú előnyük van: (1) igen kevés – általában konstans mennyiségű – memóriát használnak, és (2) sokszor nagy vagy végtelen (folytonos) keresési térben elfogadható megoldást produkálnak ott, ahol a szisztematikus algoritmusok alkalmatlanok lennének.

**3.1. Hegymászó keresés:**

A keresés egyszerűen csak egy ciklus, ami mindig javuló értékek felé – azaz felfelé – lép. Az algoritmus megáll, amikor felér a csúcsra, ahol nincsenek már magasabb értékű szomszédjai. Az algoritmus nem tart nyilván keresési fát, ezért a csomópontot leíró adatszerkezetnek csak az állapotot és a célfüggvény értékét kell nyilvántartania. A hegymászó keresés nem néz előre az aktuális állapotot közvetlenül követő szomszédokon túl. Egy kicsit arra hasonlít, mintha a Mount Everest csúcsát szeretnénk megtalálni sűrű ködben és emlékezetkihagyásban szenvedve.

A képen szöveg, Betűtípus, képernyőkép, algebra látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

**3.2. Szimulált lehűtés:**

A hegymászó keresés, amely soha nem indul „lefelé a lejtőn” a kisebb értékű (vagy nagyobb költségű) állapotok felé, garantáltan nem teljes, mert egy lokális maximumban beragadhat. Ezzel ellentétben a tisztán véletlen vándorlás – azaz a követők halmazából egyenletesen véletlen módon sorsolt követőre való átlépés – teljes, de hihetetlenül nem hatékony. Értelmes dolognak tűnik a hegymászás és a véletlen vándorlás valamiféle ötvözése, hogy mind a teljességet, mind a hatékonyságot megtarthassuk. Képzeljük el, hogy az a feladatunk, hogy egy hepehupás asztalon egy pingponglabdát a legmélyebb szakadékba juttassunk. Ha a labdát gurulni hagyjuk, egy lokális minimumba kerül. Ha a felületet megrázzuk, a labdát kiugraszthatjuk a lokális minimumból. A trükk az, hogy olyan erősen kell megrázni a felületet, hogy a labda a lokális minimumból kikerüljön, de mégsem annyira erősen, hogy a labda a globális minimumból kiugorjon. A szimulált lehűtés olyan megoldás, hogy először erősen rázunk (azaz egy magas hőmérsékleten), majd fokozatosan csökkentjük a rázás intenzitását (vagyis csökkentjük a hőmérsékletet). A szimulált lehűtés legbelső ciklusa (lásd 4.14. ábra) nagyon hasonlít a hegymászáshoz. A legjobb lépés megtétele helyett azonban egy véletlen lépést tesz. Ha a lépés javítja a helyzetet, akkor az mindig végrehajtásra kerül. Ellenkező esetben az algoritmus a lépést csak valamilyen 1-nél kisebb valószínűséggel teszi meg. A valószínűség exponenciálisan csökken a lépés „rosszaságával” – azzal a ΔE mennyiséggel, amivel a kiértékelő függvény értéke romlott. A képen szöveg, képernyőkép, Betűtípus, dokumentum látható

Automatikusan generált leírás