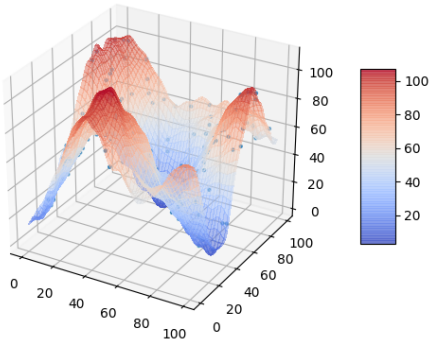
Útkeresés

1. A feladat

Bementként adott egy felület, melynek pontjait (x, y, z) koordinátákkal adjuk meg. Egy intelligens ágens sétál a felületen úgy, hogy bármelyik pontból át tud lépni egy szomszédos pontba, ha az nem tartalmaz akadályt. Két pontot akkor tekintünk szomszédosnak, hogyha a pontok x és/vagy y koordinátái közötti különbség 1. Azt szeretnénk, hogy az ágens az S kezdőpontból a lehető legoptimálisabb úton jusson el a G végpontba.



A 3 dimenziós felület és az akadályok

A két végpont közötti út több különböző szempont szerint lehet optimális, és ezek az utak nem feltétlenül egyeznek meg egymással. Néhány szempont, ami szerint minimalizálhatunk:

- lépésszám szerint – az intelligens ágens útja minimális lépésszámú legyen;
- távolság szerint – az ágens által megtett út hossza minimális legyen (pl. euklideszi távolság);
- az út megtételéhez szükséges energia szerint – vegyük észre, hogy két pont közötti út megtételéhez szükséges energia korrelál a két pont közötti távolsággal (minél hosszabb az út, annál több energiájába fog kerülni), viszont egy lejtőn lefelé könnyebben tud haladni, mint felfelé.

2. A probléma ábrázolása

A megoldás felépítéséhez kulcsfontosságú annak kiválasztása, hogy hogyan ábrázoljuk a számítógép memóriájában a bemenetként megadott felülelet. A térképet feloszthatjuk egységnyi területű cellákra, amelyeket egy-egy 3-dimenziós ponttal ábrázolhatunk. Ezekről a cellákról, a térbeli elhelyezkedésen kívül, egyéb addicionális információt is eltárolhatunk, például azt, hogy az adott cellán keresztül tud-e menni az ágens (vagy azt, hogy tartalmaz-e bónuszt/büntetést). Ezeket a cellákat azonosíthatjuk egy irányítatlan gráf csomópontjaival, és azt mondjuk, hogy a gráf két csomópontját összekötjük egy éllel, hogyha a felületen a két cella szomszédos. Így reprezentáljuk a feladat állapotterét, és a lehetséges állapot-átmeneteket.

Hogyha felépítjük a bemenetként megadott felület alapján a gráfot, akkor már ismert gráfkereső algoritmusokat is alkalmazhatunk, hogy megtaláljuk a kezdő- és végpont között a legrövidebb utat, pl. Dijkstra, Bellman-Ford, A^* algoritmus.

3. Mi a heurisztika?

A valós életből inspirálódott, komplexebb problémák esetén előfordul, hogy létezik ugyan egzakt algoritmus, amely bizonyítottan meghatározná az optimális megoldást, a gyakorlatban mégis inkább megelégszünk azzal, hogy egy közelítő megoldást találjunk. Gyakran előfordul ugyanis, hogy az egzakt megoldás kiszámítása viszonylag sok számítógépes kapacitást igényel, de a memóriafelhasználás, illetve a futási idő csökkentésének érdekében lemondunk a matematikailag bizonyítottan optimális megoldásról.

A heurisztikus függvény (röviden: heurisztika) egy olyan függvény lesz, amely egy sajátos (gyakran feladat-specifikus) mérőszámot társít minden lehetséges megoldáshoz, amellyel azt próbálja becsülni, hogy mennyire kedvező az aktuális megoldás.

Az útkeresési probléma esetén, két rögzített pont közötti út hosszúságának becslésére egy lehetséges heurisztikus függvény például az euklideszi távolság. Az euklideszi távolság egy közelítő értéket (becslést) ad arról, hogy mekkora a két pontot összekötő út valódi költsége. Természetesen megtörténhet, hogy a becsült távolság kisebb vagy nagyobb az ágens által bejárandó út valódi költségénél (például a két pont közötti becsült távolság nem veszi figyelembe az út menti akadályok vagy bónuszpontok/büntetések elhelyezkedését). Egy olyan lokális keresésre épülő gráf algoritmust szeretnénk implementálni, amely az euklideszi távolságot használja két pont közötti út hosszának becslésére.

3.1. Az A^* algoritmus

Az A^* algoritmus a best first search (BFS) egyik legismertebb változata. A módszer lényege, hogy minden lépésben egy olyan csomópont kerül kifejtésre, amelyen keresztül vezető út becsült költsége minimális. Így az állapottér pontjait az algoritmus az

$$f(n) = g(n) + h(n)$$

költségfüggvénnyel értékeli ki, ahol

$$g(n) = \text{ az } S \text{ kezdőponttól az } n \text{ csomópontig } \textbf{megtett út költsége},$$

és

$$h(n) = \text{ az } n \text{ ponttól a } G \text{ végpontig hátralevő út } \textbf{becsült költsége}.$$

A csomópontok g-értékei (g-score) így a kezdőponttól az n pontig megtett út valódi, pontos költségét adják meg, míg a h-értékek heurisztikán alapszanak, csak becsült költséget reprezentálnak. Ennek ellenére, az A^* algoritmus bizonyítottan optimális megoldást szolgáltat abban az esetben, ha a h heurisztikus függvény **megengedett**, azaz ha a $h(n)$ érték mindig alulról becsüli meg az n ponttól a célig vezető út valódi költségét, bármely n csomópont esetén. Mivel az eukildeszi távolság esetén érvényes a jól ismert **háromszög-egyenlőtlenség**, így az euklideszi metrika egy megengedett heurisztikát biztosít. Emellett az A^* implementációja során arra is figyelni kell, hogy ha egy n csomóponthoz több különböző $f(n)$ érték társul (ez csak annyit jelent, hogy az n pontot több különböző úton keresztül közelíthetjük meg az S kezdőpontból kiindulva), akkor az algoritmus mindig a legkisebb $f(n)$ értéket vegye figyelembe, és az ennél drágább n-en keresztül vezető utakat vesse el. Mivel az euklideszi távolságra érvényes a háromszög-egyenlőtlenség, ezért egy standard prioritási sor felhasználása segítségével ez az elv is teljesülni fog.

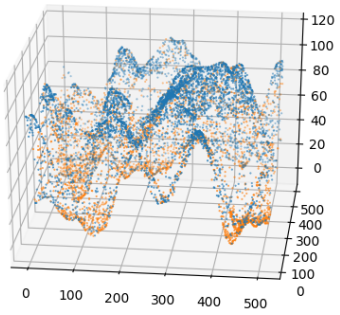
Tehát ha a megtett út hosszát szeretnénk minimalizálni, akkor az euklideszi távolságot heurisztikaként alkalmazó A^* algoritmus matematikailag is bizonyítottan optimális megoldást ad. A bizonyításhoz, és az algoritmus pszeudokódjához lásd a [3, 84. old.] könyvet.

A probléma sajnos nem ilyen egyszerű, ha a heurisztika nem megengedett, mert felülről becsüli a hátralevő út költségét (például a heurisztika nem veszi figyelembe a hátralevő út mentén talált bónuszpontokat). Mivel az A^* ebben az esetben már nem garantál optimális megoldást, viszont a memóriaigénye továbbra is nagy, komplexebb problémák esetén érdemes más módszerekhez folyamodni.

4. Útkeresés, mostmár energia-számolással

Az 1. részben vázolt feladatot kiegészítjük úgy, hogy az intelligens ágens bónuszokat, illetve büntetéseket tudjon gyűjteni a felületen történő sétája során. Ezen a felületen nem lesznek akadályok, lecseréljük őket

büntetésekre, amelyek -10 energiába fognak kerülni az ágensnek, hogyha egy olyan cellára lép, amelyen büntetés található. Ugyanakkor, az ágens $+5$ bónusz-energiát tud szerezni, hogyha olyan cellára lép, amelyen bónusz található. Az ágens csak akkor fogja összeszedni a bónuszt/büntetést, amikor először lép a megfelelő cellára. Így el tudjuk kerülni azt, hogy az ágens újra és újra visszatérjen a bónuszokat tartalmazó mezőkre ahelyett, hogy a célt keresné.



A bónuszok (kék) és büntetések (narancssárga) elhelyezkedése 3 dimenzióban. (Annak érdekében, hogy ne legyen túlzsúfolt az ábránk, nem ábrázoltuk magát a felületet.)

Az ágens felületen történő lépései energiát fognak igényelni. A következő képlet adja meg, hogy mekkora energia szükséges egy (x_1, y_1, z_1) pontból átmenni az (x_2, y_2, z_2) pontba:

$$E = 0.2d + 0.1(z_2 - z_1)$$

ahol d a két pont közötti euklideszi távolság.

5. Mi a meta-heurisztika?

A meta-heurisztikák olyan általános (magasabb-rendű) eljárások, amelyek heurisztikákat generálnak egy adott optimalizációs feladat megoldása érdekében. Lényegében, a meta-heurisztika egy "recept" arra, hogyan generáljunk közelítő megoldásokat egy konkrét feladat esetén. Gyakran alkalmaznak meta-heurisztikákat NP-teljes feladatok megoldására. Néhány példa meta-heurisztikára a teljesség igénye nélkül:

- evolúciós algoritmusok: evolúciós programozás, genetikus algoritmus, genetikus programozás, neuro-evolúció (neuroevolution), stb;
- tabu-keresés;
- szimulált hűtés;
- hangya kolónia optimalizációs algoritmus.

6. Hangya kolónia optimalizációs algoritmus (ACO)

Raj-intelligenciának (Swarm Intelligence) [2] nevezzük a nem központosított, önszerveződő, természetes vagy mesterséges rendszerek együttes viselkedését. A raj egy viszonylag homogén rendszer: egyforma egyedekből áll, esetlegesen a rajt alkotó egyedek besorolhatóak néhány jól meghatározott kategóriába. Az egyedek kommunikálhatnak egymással direkt módon vagy a környezeten keresztül.

A hangya kolónia optimalizációs algoritmus (ACO) [1] egy populáción alapuló metaheurisztika, mely a raj-intelligenciára alapoz. Az algoritmus mesterséges hangyákat készít, amelyek be fogják járni a megoldások terét. Ahhoz, hogy a módszert alkalmazni tudjuk, az optimalizálandó problémát egy súlyozott, irányított gráfként ábrázoljuk. Ekkor egy hangya által megtett út lesz a kérdéses feladat egy lehetséges

megoldása. Az egyedek közötti információcsere az ún. feromonokon keresztül valósul meg. Ezek a feromonok fogják befolyásolni azt, hogy a hangya milyen csomópontokat látogat meg a gráf bejárása során.

1. Algoritmus. (ACO)
begin
  schedule while *leállási_feltétel* = *hamis* **do begin**
    hangya_megoldás_építés()
    démon_lépés()
    feromon_frissítés()
  end
end.

6.1. Egy megoldás építése

Az m darab hangya épít egy-egy megoldást a gráf csomópontjaiból és azokat összekötő élekből. Egy hangya egy megoldást az üreshalmazból kiindulva lépésenként generál, a szomszédos csomópontok hozzáadása segítségével, amíg el nem ér a végpontba. Két szempontot fog figyelembe venni akkor, amikor szeretne hozzáadni egy új csomópontot az eddig megtett útjához: egyrészt befolyásolni fogja az előző hangyák által hátrahagyott feromon értéke; másrészt, bele fog számítani a döntésébe az, hogy a kiválasztandó csomópont mennyivel visz közelebb a feladat megoldásához.

A k. hangya felépítette az s^p utat (részmegoldást) a kezdőpontból az i. csomópontba. Ebből a csomópontból szeretne tovább lépni egy valamilyen c_{ij} csomópontba. A c_{ij} csomópont "jóságát" a következő értékek fogják jellemezni: τ_{ij}, az aktuális feromon értéke; illetve az η_{ij} heurisztikus érték. Az η heurisztikát a megoldandó optimalizációs feladat függvényében választjuk ki. Gyakran használják az $\eta_{ij} = 1/d_{ij}$ kifejezést heurisztikus függvényként, ahol d_{ij} aktuális csomópont és a c_{ij} közötti út megtételéhez szükséges energia.

Tehát, a c_{ij} csomópont kiválasztása egy adott lépésben az alábbi valószínűségi szabály szerint fog történni:

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{ir} \in N(i)} \tau_{ir}^{\alpha} \cdot \eta_{ir}^{\beta}}, \quad \forall c_{ij} \in N(i),$$

ahol $N(i)$ az i csomópont szomszédainak halmazát jelöli. Az α és β konstansok ún. hiperparaméterek, amelyek azt szabályozzák, hogy egy csomópont kiválasztásakor milyen arányban érvényesüljön a feromon, illetve a heurisztikus érték. Ezek a paraméterek kísérleti úton állíthatók be.

6.2. Démon folyamat

A Démon folyamatok olyan opcionális lépések, amelyeket az egyedülálló mesterséges hangyák nem képesek elvégezni, de gyorsítják az optimumhoz való konvergálást. Ezek lehetnek olyan probléma-specifikus lépések, amelyek szükségesek ahhoz, hogy garantálni tudjuk a megoldások helyességét. Például, hogyha vannak olyan feladat-specifikus feltételek, amelyeknek teljesülnie kell ahhoz, hogy elfogadjuk a hangyák által generált lehetséges megoldásokat, akkor ebben a lépésben ellenőrizzük le, hogy teljesülnek-e ezek a feltételek.

Egy másik gyakran használt démon folyamat, hogy a hangyák által generált utakra alkalmazunk egy lokális optimalizációt. Ezáltal megpróbáljuk gyorsítani az optimum keresést, ugyanis kiszűrjük azokat a megoldásokat, amelyek kevésbé optimálisak.

Amennyiben a hangyák útjainak generálásakor nem ellenőriztük le azt, hogy ne tudják másodszor is meglátogatni azokat a csomópontokat, ahol már jártak, egy démon folyamat során kiszűrhetjük a köröket az iterációban meghatározott utakból.

6.3. Feromon

A gráf minden csomópontjához hozzárendelünk egy τ_{ij} valós értéket, **feromont**, amely segítségével kommunikálnak a hangyák egymással. Ez az érték kezdetben egy 0-hoz közeli pozitív szám, és a szimuláció során minden ott elhaladó hangya plusz feromont hagy maga után. A való élethez hasonlóan, a feromon idővel elpárolog (evaporáció). Ennek következtében, minden iteráció végén automatikusan csökkenni fog az aktuális feromon értéke. A frissítési szabály magába foglalja az evaporációt és a hangyák által az adott iterációban hátrahagyott feromonokat. Célunk, hogy fokozatosan csökkentsük a kevésbé optimális megoldásokhoz tartozó élek/csomópontok feromon-értékeit, ugyanakkor növelheljük a kedvező megoldások éleihez vagy csomópontjaihoz tartozó értékeket:

$$\tau_{ij} \leftarrow \overbrace{(1-\rho)\cdot\tau_{ij}}^{\text{evaporáció}} + \overbrace{\rho\cdot\sum_{s\in S_{upd}|c_{ij}\in s} F(s)}^{\text{új feromon}},$$

ahol a $\rho \in (0,1]$ paraméter szabályozza a feromonok csökkenését. $F(s)$-el jelöljük azt a függvényt, ami megadja, hogy minden lépésben mennyivel fogjuk növelni az s úthoz tartozó csomópontok feromon értékét. Az S_{upd} azoknak az utaknak a halmaza, amelyek csomópontjait szeretnénk frissíteni.

Hogyha L_k a k. hangya által megtett úthoz szükéseges energia, akkor az aktuális itereció végén a frissítési érték a következőképpen alakul:

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{L_k} & \text{ha a } k. \text{ hangya meglátogatta az } (i,j)\text{-t az útja során,} \\ 0 & \text{különben.} \end{cases}$$

Arra, hogy milyen utak mentén szeretnénk új feromont hagyni, több különböző frissítési szabályt fogalmazhatunk meg:

- a legegyszerűbb megközelítés, hogy minden olyan út mentén, amit ebben az iterációban valamelyik hangya kigenerált, hagyunk extra feromont

$$S_{upd} \leftarrow S_{iter}$$

- meghatározzuk azt, hogy az újonnan generált utak közül melyik a legoptimálisabb és csak ennek a csomópontjaihoz adunk hozzá feromont

$$S_{upd} \leftarrow \arg\max_{s\in S_{iter}} F(s)$$

- csak a valaha volt legjobb út mentén adunk új feromont hozzá a pontokhoz. Amikor ezt a szabályt alkalmazzuk, előfordulhat, hogy olyan út mentén fogunk frissíteni, amelyet ebben az iterációban egyetlen hangya sem járt be.

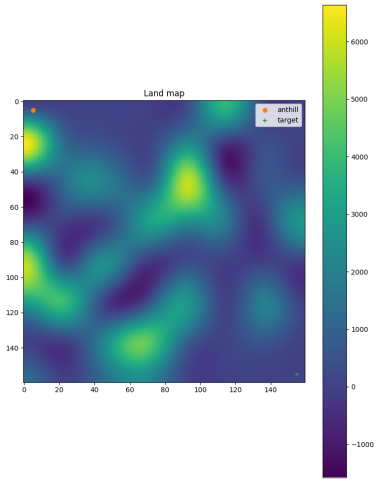
$$S_{upd} \leftarrow \arg\max_{s\in S_{all\_time}} F(s)$$

6.4. ACO változatai

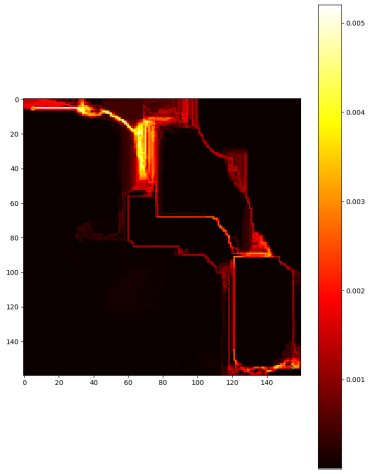
A három legismertebb fajtája az ACO algoritmusnak:

- Ant system (AS)
- Ant colony system (ACS)
- MAX-MIN ant system (MMAS)

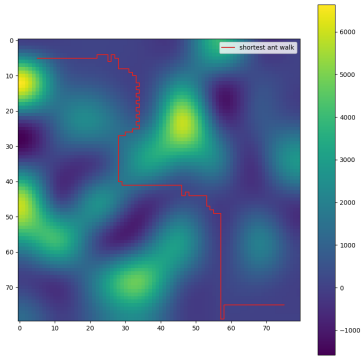
6.5. Példa



(a) A felület hőtérképe. A kezdőpont narancssárgával, míg a végpont kékkel van jelölve.



(b) Feromon térkép egy köztes iterációban



(c) Az ACO által meghatározott legrövidebb út

7. Tippek, hasznos infók

- Hogyan inicializáljuk a feromonokat?
  - Kezdetben minden csomóponthoz rendeljünk egy kicsi, de NEM 0 valós számot. Azért ne inicializáljuk 0-val a feromonok értékeit, mert ekkor a $p(c_{ij}|k)$ valószínűség is 0 lenne, így az összes szomszédos csomópontba 0 valószínűséggel lépne a hangya.
- Hogyan válasszuk meg a függvényt arra, hogy mekkora energiába kerül a hangyának egy útja a felületen?
  - Egy lehetséges megoldás, hogy összegezzük az összegyűjtött bónuszokat, büntetéseket megfelelő előjellel, illetve hozzáadjuk ehhez az értékhez azt, hogy mennyi energiát használ el a hangya ahhoz, hogy sétáljon a felületen. Ekkor, szélsőséges esetekben előfordulhat, hogy a hangya annyi bónuszt gyűjt össze az útja során, hogy a sétája negatív energiába fog kerülni.

- Hogyan tudjuk megelőzni azt, hogy egy hangya által megtett úthoz szükséges L_k energia negatív legyen?
  - Hogy elkerüljük azt, hogy negatív feromon frissítési értékeket kapjunk, eltolhatunk minden úthoz szükséges energia-értéket úgy, hogy garantálni tudjuk, hogy pozitív legyen. A szimuláció előtt ki tudjuk számolni egy adott felület esetén, hogy mennyi az a maximális bónusz energia, amennyit össze lehet gyűjteni a térképen való bolyongás során, legyen ez B_{max}. Hogyha minden hangyánál az $L'_k = (B_{max} + 1 + L_k)$ képletet használjuk a szükséges energia kiszámítására, akkor már teljesülni fog, hogy $L'_k > 0$.
  - Megválaszthatjuk az L_k-t úgy is, hogy ez csak a hangya sétájára használt energiát összegezze, és nem tartalmazza a bónuszokat/büntetéseket. Ekkor a csomópontot jellemző η_{ij} heurisztikus érték mutatná azt, hogy az illető csomópont tartalmaz-e bónuszt/büntetést.

Hivatkozások

[1] M. Dorigo. "Ant colony optimization". *Scholarpedia* 2.3 (2007). revision #90969, 1461. old. DOI: 10.4249/scholarpedia.1461.

[2] M. Dorigo és M. Birattari. "Swarm intelligence". *Scholarpedia* 2.9 (2007). revision #138640, 1462. old. DOI: 10.4249/scholarpedia.1462.

[3] S. Russel és P. Norvig. *Mesterséges Intelligencia Modern megközelítésben. Második, átdolgozott, bövített kiadás*. Panem, Budapest, 2005.