A díjgyűjtő utazóügynök és kapcsolódó problémák

Készítette: Czirják Lilla
Témavezető: Dr. Király Tamás
2024. MÁJUS

1. Bevezetés

Az utazóügynök problémája egy ma már klasszikusnak számító kombinatorikus optimalizálási feladat, ahol egy adott, élsúlyozott n pontú teljes gráfban keressük a legrövidebb Hamilton-kört. Közismert, hogy ez a kérdés NP-nehéz, viszont számtalan közelítő algoritmust alkottak az eldöntésére. A problémának számos különböző általánosítását fogalmazták meg. Ezen dolgozat témája az utazóügynök-probléma díjgyűjtő változata, melyben a metrikus élsúlyokon kívül adott a gráf csúcsainak egy pozitív súlyozása is: a díjak, amiket az egyes városokban gyűjthet az ügynök. Ez esetben nem követeljük meg, hogy a túra az összes csúcsot tartalmazza, azaz nem Hamilton-kört keresünk, hanem egy adott gyökérpontból kiindulva egy tetszőleges méretű kört a gráf csúcsain, melyre az összegyűjtött bevétel és az útiköltség különbsége maximális. Sok esetben azonban bevételek helyett érdemesebb a csúcssúlyokra büntetésekként tekinteni, amelyeket a kihagyott csúcsok után fizetünk. Így a problémát egy minimalizálási feladatként tudjuk formalizálni.

A projektmunka keretében a félév során a szakirodalom megismerése mellett egy egyszerűbb heurisztikus közelítő algoritmus implementálásával, majd véletlen gráfokon történő tesztelésével foglalkoztam, valamint kipróbáltam néhány túrajavító módszert, amelyek segítségével egy adott megoldásból kiindulva találhatunk jobb megoldást.

2. A díjgyűjtő utazóügynök

1. Definíció. *Adott egy* $G = (V,\ E)$ *teljes gráf,* $r \in V$ *gyökér,* $c_e \geq 0\ \forall e \in E$ *metrikus élhosszak, és* $\pi_v \geq 0\ \forall v \in V \backslash \{r\}$ *csúcssúlyok.*
A díjgyűjtő utazóügynök-feladat egy $C = (V_C,\ E_C)$ *kör megtalálása* G*-ben, amire* $r \in V_C$*, és* $\sum_{e \in E_C} + \sum_{v \in V \backslash V_C}$ *minimális.*

A feladat lineáris programozási relaxáltja:

$$min \sum_{e \in E} C_e x_e + \sum_{v \in V} \pi_v (1 - y_v)$$

$$
\begin{aligned}
x(\delta(v)) &= 2y_v \quad \forall v \in V\backslash\{r\} \\
x(\delta(r)) &\leq 2 \\
x(\delta(S)) &\geq 2y_v \quad \forall S \subseteq V\backslash\{r\},\ v \in S \\
y_r &= 1 \\
x_e &\geq 0 \quad \forall e \in E \\
y_v &\geq 0 \quad \forall v \in V
\end{aligned}
$$

Az x_e változók a gráf éleinek, míg az y_v változók a csúcsoknak feleltethetőek meg, és $S \subseteq V$ esetén $\delta(S) := \{e \in E :\ |e \cap S| = 1\}$, valamint $\delta(v) := \delta(\{v\})$.

3. Egy heurisztikus algoritmus

A heurisztikus algoritmusok esetében általában nem tudunk megadni egy felső korlátot a megoldás hibájára az optimumhoz képest, mert ez szélsőséges példákon futtatva az algoritmust igen nagy lehet. Ezzel szemben véletlen példákon sok esetben jól működnek, így gyakorlati szempontból hasznosak.

Tekintsük a következő algoritmust. Legyen adott egy n-csúcsú teljes gráf metrikus élhosszakkal, a csúcsok egy pozitív súlyozása, valamint egy rögzített gyökérpont.

A gyökérből, mint egy egy pontból álló túrából kiindulva egyesével adjuk hozzá a csúcsokat a már meglévő körhöz. Minden lépésben a legjobb lehetőséget választjuk a túra növelésére, azaz az összes kimaradó csúcsot megpróbáljuk beszúrni minden szomszédos csúcspár közé az aktuális túrába, majd azt a beszúrást végezzük el, amellyel legtöbbet javíthatunk a megoldás értékén. - Itt azt is megengedjük, hogy a javítás egy adott lépésben negatív legyen. - Végül az n darab különböző hosszúságú túra közül a legjobb eredményt adót választjuk.

3.1. Túrajavító módszerek

Ha már ismerünk egy megoldást - akár egy véletlen bejárásból is kiindulhatunk, - akkor ezt megpróbálhatjuk további heurisztikus algoritmusokkal javítani. A fent leírt algoritmust két javító eljárással teszteltem, ezek a következők.

1. Pontok törlése

   Adott túra esetén egyszerűen végignézhetjük minden csúcsra, hogy annak elhagyásával javul-e a megoldás értéke.

2. Két él cseréje

   Az adott túrában minden $(v_1v_2,\ u_1u_2)$ élpárra (ahol a v_1v_2 élen hamarabb haladunk át, mint az u_1u_2-n), megnézzük, hogy javítható-e a megoldás a következő módon. A v_1v_2 és u_1u_2 éleket töröljük a megoldásból, és helyettük bevesszük a v_1u_1 és v_2u_2 éleket. A $v_2 - u_1$ utat az $u_1 - v_2$ útra cseréljük (az ellenkező irányból járjuk be). Ezt ismételjük.

3.2. Tesztelés

Az algoritmusokat véletlen gráfokon teszteltem, amelyeket a következő paraméterekkel generáltam. A csúcsok száma előre rögzített n=100, a csúcssúlyok véletlen egészek 0 és 100 között. Az élsúlyok megkapásához véletlen koordinátákat rendeltem a csúcsokhoz, majd ezekből számoltam Euklideszi-távolságot. A futásidő gyorsítása érdekében egészre kerített értékekkel dolgoztam. A távolságokat felfelé kerekítettem, ugyanis könnyen látható, hogy így szintén metrikát kapunk.

A fentebb leírt heurisztikus algoritmust, majd a kapott túrákon a két említett javítási módszert futtattam. Összevetettem, hogy adott példákon mekkora javítást lehet elérni, ha előbb a csúcstörlést, majd az élcserét alkalmazzuk, illetve ha fordított sorrendben járunk el. Az alábbi táblázatokban néhány futás eredménye látható. Az első táblázatban a 'Javítás 1.' oszlop a csúcstörléssel elért javítást, a 'Javítás 2.' az ezt követően végrehajtott élcserékkel történő javítást tartalmazza. A második esetben ugyanazokon a gráfokon fordított sorrendben végeztem a javításokat, ennek eredményei szerepelnek a 2. táblázatban.
100 futásból 91 esetben a második módszerrel lehetett többet javítani a megoldáson, vagyis amikor az ismételt élcseréket követően alkalmaztuk a csúcstörlést.

	Túra hossza	Érték	Javítás 1.	Javítás 2.	Össz. javítás
0	95	1152	718	236	954
1	68	1107	625	265	890
2	43	759	452	91	543
3	74	420	516	240	756
4	19	145	219	112	331
5	43	374	334	150	484
6	84	316	923	394	1317
7	73	958	897	265	1162
8	62	707	610	108	718
9	90	339	506	755	1261
10	90	919	489	601	1090

	Túra hossza	Érték	Javítás 1.	Javítás 2.	Össz. javítás
0	95	1152	656	489	1145
1	68	1107	578	351	929
2	43	759	415	168	583
3	74	420	743	253	996
4	19	145	172	60	232
5	43	374	360	115	475
6	84	316	1012	438	1450
7	73	958	698	539	1237
8	62	707	375	354	729
9	90	339	901	512	1413
10	90	919	855	263	1118

A kód elérhető a következő linken:
https://colab.research.google.com/drive/1ODzrT0YPFiCL41IfABo3TVagoZoS9Jd1?usp=sharing

Hivatkozások

[1] Blauth, Klein, Nägele: A Better-Than-1.6-Approximation for Prize-Collecting TSP (2023)

[2] Goemans, Williamson: A general approximation technique for constrained forest problems (1995)

[3] Ausiello, Bonifaci, Leonardi, Marchetti-Spaccamela: Prize-Collecting Traveling Salesman and Related Problems