Kombinatorikus optimalizálás 8. hét

Pusztai Pál
pusztai@sze.hu

Tartalom

- Utazó ügynök probléma
  - Heurisztikák
    - Nearest addition, Nearest insertion, Farthest insertion, Cheapest insertion eljárások
    - A 2-patching és a 2-optimális eljárás
    - Christofides eljárása

Utazó ügynök probléma

- Az utazó ügynök probléma

Adott n számú város és az azokat összekötő utak, amelyeknek ismert a hossza. Adott továbbá egy ügynök, akinek adott városból kiindulva, minden várost végig kell látogatnia úgy, hogy minden várost pontosan egyszer érint, és az út befejeztével visszatér a kiindulási városba.

Feladat: Határozzuk meg az ügynök legrövidebb útját.

Megjegyzés: A feladat NP-teljes probléma.

- Jelölések
  - A városokat jelöljük az 1, 2, …, n számokkal.
  - Egy tetszőleges $Q \subseteq \{1, \dots, n\}$ halmazra jelölje $\bar{Q}$ az $\{1, \dots, n\} \backslash Q$ halmazt.
  - c_{ij} : az i városból a j városba vezető út hossza (ha nincs ilyen út, akkor legyen $c_{ij} = W$, ahol W egy megfelelően nagy szám).
  - $x_{ij} = \begin{cases} 1, & \text{ha az ügynök az } i \text{ városból a } j \text{ városba megy,} \\ 0 & \text{különben.} \end{cases}$

Megjegyzés: A $\mathbf{C}^{n \times n}$ mátrix nem feltétlenül szimmetrikus.

Heurisztikák

Az NP-nehéz problémák esetén nem ismeretesek polinomkorlátos műveletigényű megoldó eljárások.

A rendelkezésre álló algoritmusok műveletigénye általában exponenciális függvénye a probléma méretének, így a nagyobb méretű feladatok megoldása esetenként nem realizálható, azaz szükség van közelítő eljárásokra.

Az optimálishoz közeli megoldást adó algoritmust **közelítő algoritmusnak** vagy **heurisztikának** nevezzük.

Tegyük fel, hogy egy optimalizálási probléma minden lehetséges megoldásának pozitív költsége van.

Optimális megoldás a **minimális** vagy **maximális** költséggel rendelkező megoldás.

Egy közelítő algoritmus **hibakorlát-függvénye** $\rho(n)$, ha az általa adott megoldás C költsége – minden n méretű bemenetre – az optimális megoldás C^* költségének legfeljebb $\rho(n)$-szerese, ill. legalább $\rho(n)$-ed része, azaz:

$$\max(C/C^*, C^*/C) \leq \rho(n).$$

Ha egy algoritmus biztosítja a $\rho(n)$ hibakorlát-függvény betartását, akkor ***ρ(n)*-közelítő algoritmusnak** nevezzük.

Ha a hibakorlát-függvény független n-től, akkor ***ρ* hibakorlátról**, és ρ-közelítő algoritmusról beszélünk.

Megjegyzés: A ρ hibakorlátot **approximációs hányadosnak** is nevezik.

TSP heurisztikák

- A TSP heurisztikák csoportosítása
  1. Körútépítő eljárások,
  2. Körút javítását szolgáló algoritmusok,
  3. Kombinált eljárások (1. és 2. kombinációi).

Jelölés: Az egyszerűség kedvéért a továbbiakban jelölje N az $\{1, \dots, n\}$ halmazt.

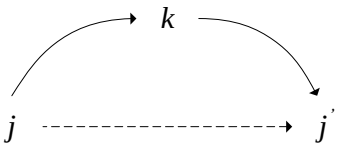
A körútépítő eljárásokban a számításokhoz egy olyan távolságvektort fogunk használunk ($\mathbf{v}_r$), amely minden iterációs lépésben (r) megadja az aktuális részkörúton kívüli városoknak a részkörúttól való távolságát.

Egy város távolságát a részkörúttól az adott város és a hozzá legközelebb eső részkörútbeli város távolsága adja.

Az első iterációs lépésben ez a $\mathbf{C}$ mátrix első sorvektora az (1, 1) indexű elem kivételével (mivel az 1 város a kiinduló város).

TSP heurisztikák: Nearest addition

- Előkészítő rész
  - Legyen $r = 1$, $I_r = \{1\}$, $E_r = \{(1,1)\}$. (Az 1. város lesz az induló részkörút).
  - Folytassuk az eljárást az iterációs résszel.
- Iterációs rész (r. iteráció)
  - Ha $r = n$, akkor vége az eljárásnak, az E_r-beli élekből álló körút az eljárással szolgáltatott lehetséges megoldás.
  - Ellenkező esetben ($r < n$) határozzunk meg egy olyan $j \in I_r$, $k \in N \backslash I_r$ indexpárt, amelyre
    $$c_{jk} = min_{t \in N \backslash I_r}\{\min\{c_{st} : s \in I_r\}\}$$
  - Legyen $I_{r+1} = I_r \cup \{k\}$. Mivel $j \in I_r$, ezért pontosan egy olyan $j' \in I_r$ index van, amelyre $(j, j') \in E_r$ ($r = 1$ esetén $j = j'$).
  - Legyen $E_{r+1} = (E_r \backslash \{(j, j')\}) \cup \{(j, k), (k, j')\}$.



  - Legyen $r = r + 1$, majd folytassuk az eljárást a következő iterációs lépéssel.

Az aktuális részkörutat tehát a körúthoz legközelebb eső, a körúton kívüli várossal bővítjük.

Hatékonyság: Az algoritmus műveletigénye $O(n^2)$, így általános esetben nem várható hozzá konstans approximációs hányados.

Példa

A Nearest addition algoritmus működése:

$$\begin{pmatrix} W & 2 & 11 & 10 & 8 & 7 & 6 & 5 \\ 6 & W & 1 & 8 & 8 & 4 & 6 & 7 \\ 5 & 12 & W & 11 & 8 & 12 & 3 & 11 \\ 11 & 9 & 10 & W & 1 & 9 & 8 & 10 \\ 11 & 11 & 9 & 4 & W & 2 & 10 & 9 \\ 12 & 8 & 5 & 2 & 11 & W & 11 & 9 \\ 10 & 11 & 12 & 10 & 9 & 12 & W & 3 \\ 7 & 10 & 10 & 10 & 6 & 3 & 1 & W \end{pmatrix}$$

r	$\mathbf{v}_r$	(j, k)	részkörút
1	$(-, 2^*, 11, 10, 8, 7, 6, 5)$	(1, 2)	(1, 2)
2	$(-, -, 1^*, 8, 8, 4, 6, 5)$	(2, 3)	(1, 2, 3)
3	$(-, -, -, 8, 8, 4, 3^*, 5)$	(3, 7)	(1, 2, 3, 7)
4	$(-, -, -, 8, 8, 4, -, 3^*)$	(7, 8)	(1, 2, 3, 7, 8)
5	$(-, -, -, 8, 6, 3^*, -, -)$	(8, 6)	(1, 2, 3, 7, 8, 6)
6	$(-, -, -, 2^*, 6, -, -, -)$	(6, 4)	(1, 2, 3, 7, 8, 6, 4)
7	$(-, -, -, -, 1^*, -, -, -)$	(4, 5)	(1, 2, 3, 7, 8, 6, 4, 5)

A táblázatban az aktuális részkörutat ciklikus permutációként adtuk meg.

A kapott körúthoz tartozó célfüggvényérték:

$$c_{12} + c_{23} + c_{37} + c_{78} + c_{86} + c_{64} + c_{45} + c_{51} = 2 + 1 + 3 + 3 + 3 + 2 + 1 + 11 = 26.$$

Megjegyzés

- Az eljárás most éppen egy optimális megoldást adott.
- A kiválasztott város most éppen mindig az aktuális részkörút végére került.

Feladatok

- Milyen körutat ad a Nearest addition algoritmus az alábbi TSP feladatra és mennyi a körút költsége? Adjuk meg az egyes iterációs lépésekhez tartozó adatokat (részkörút, távolságvektor, kiválasztott él) is! (Ha egy lépésben több „jó választás” is lehetséges, akkor válasszuk a „legkisebb indexűt”!)

$$TSP\begin{pmatrix} W & 2 & 4 & 2 \\ 2 & W & 2 & 3 \\ 1 & 2 & W & 4 \\ 3 & 2 & 4 & W \end{pmatrix}$$

TSP heurisztikák: Nearest insertion

- Előkészítő rész
  - Legyen $r = 1$, $I_r = \{1\}$, $E_r = \{(1,1)\}$. (Az 1. város lesz az induló részkörút).
  - Folytassuk az eljárást az iterációs résszel.
- Iterációs rész (r. iteráció)
  - Ha $r = n$, akkor vége az eljárásnak, az E_r-beli élekből álló körút az eljárással szolgáltatott lehetséges megoldás.
  - Ellenkező esetben ($r < n$) határozzunk meg egy olyan $j \in I_r$, $k \in N \backslash I_r$ indexpárt, amelyre
    $c_{jk} = min_{t \in N \backslash I_r}\{\min\{c_{st} : s \in I_r\}\}$
  - Legyen $I_{r+1} = I_r \cup \{k\}$, majd válasszunk egy olyan $(u, v) \in E_r$ élt, amelyre
    $\delta_{uv} = c_{uk} + c_{kv} - c_{uv} = \min\{c_{sk} + c_{kt} - c_{st} : (s,t) \in E_r\}$.
  - Legyen $E_{r+1} = (E_r \backslash \{(u,v)\}) \cup \{(u,k),(k,v)\}$.
  - Legyen $r = r + 1$, majd folytassuk az eljárást a következő iterációs lépéssel.

Az eljárás hasonló az előzőhöz, csak itt az aktuális részkörúthoz legközelebb lévő k várost nem a hozzá legközelebb eső részkörútbeli város után szúrjuk be, hanem megvizsgáljuk k lehető legjobb (legkisebb költséggel járó) beszúrását a részkörútba, és azt hajtjuk végre.

Hatékonyság: Az algoritmus műveletigénye $O(n^2)$.

Megjegyzés: Olyan TSP feladatokra, amelyek költségmátrixa szimmetrikus és teljesül rá a háromszög-egyenlőtlenség, az algoritmus approximációs hányadosa 2.

Egy mátrixra teljesül a **háromszög-egyenlőtlenség**, ha $c_{uv} \leq c_{uw} + c_{wv}$ minden u, v, w csúcsra.

Példa

A Nearest insertion algoritmus működése:

$$\begin{pmatrix} W & 2 & 11 & 10 & 8 & 7 & 6 & 5 \\ 6 & W & 1 & 8 & 8 & 4 & 6 & 7 \\ 5 & 12 & W & 11 & 8 & 12 & 3 & 11 \\ 11 & 9 & 10 & W & 1 & 9 & 8 & 10 \\ 11 & 11 & 9 & 4 & W & 2 & 10 & 9 \\ 12 & 8 & 5 & 2 & 11 & W & 11 & 9 \\ 10 & 11 & 12 & 10 & 9 & 12 & W & 3 \\ 7 & 10 & 10 & 10 & 6 & 3 & 1 & W \end{pmatrix}$$

r	$\mathbf{v}_r$	k	(u, v)	δ_{uv}	részkörút
1	(–, 2*, 11, 10, 8, 7, 6, 5)	2	(1, 1)	-	(1, 2)
2	(–, –, 1*, 8, 8, 4, 6, 5)	3	(2, 1)	0	(1, 2, 3)
3	(–, –, –, 8, 8, 4, 3*, 5)	7	(3, 1)	8	(1, 2, 3, 7)
4	(–, –, –, 8, 8, 4, –, 3*)	8	(7, 1)	0	(1, 2, 3, 7, 8)
5	(–, –, –, 8, 6, 3*, –, –)	6	(2, 3)	8	(1, 2, 6, 3, 7, 8)
6	(–, –, –, 2*, 6, –, –, –)	4	(6, 3)	7	(1, 2, 6, 4, 3, 7, 8)
7	(–, –, –, –, 1*, –, –, –)	5	(4, 3)	0	(1, 2, 6, 4, 5, 3, 7, 8)

A kapott körúthoz tartozó célfüggvényérték:

$$c_{12} + c_{26} + c_{64} + c_{45} + c_{53} + c_{37} + c_{78} + c_{81} = 2 + 4 + 2 + 1 + 9 + 3 + 3 + 7 = 31.$$

Feladatok

- Milyen körutat ad a Nearest insertion algoritmus az alábbi TSP feladatra és mennyi a körút költsége? Adjuk meg az egyes iterációs lépésekhez tartozó adatokat (részkörút, távolságvektor, kiválasztott város, kiválasztott él, minimális beszúrási költség) is! (Ha egy lépésben több „jó választás” is lehetséges, akkor válasszuk a „legkisebb indexűt”!)

$$TSP\begin{pmatrix} W & 2 & 4 & 2 \\ 2 & W & 2 & 3 \\ 1 & 2 & W & 4 \\ 3 & 2 & 4 & W \end{pmatrix}$$

  - Megjegyzés
    - Két él esetén a „kisebb indexű” legyen az, amelyiknél a kezdőpont indexe kisebb!

TSP heurisztikák: Farthest insertion

Az eljárás hasonló az előzőhöz, csak itt az aktuális részkörúttól legtávolabb lévő k várost választjuk ki, és ezt szúrjuk be a lehető legkisebb költséggel a részkörútba.

A Farthest insertion algoritmus működése:

$$\begin{pmatrix} W & 2 & 11 & 10 & 8 & 7 & 6 & 5 \\ 6 & W & 1 & 8 & 8 & 4 & 6 & 7 \\ 5 & 12 & W & 11 & 8 & 12 & 3 & 11 \\ 11 & 9 & 10 & W & 1 & 9 & 8 & 10 \\ 11 & 11 & 9 & 4 & W & 2 & 10 & 9 \\ 12 & 8 & 5 & 2 & 11 & W & 11 & 9 \\ 10 & 11 & 12 & 10 & 9 & 12 & W & 3 \\ 7 & 10 & 10 & 10 & 6 & 3 & 1 & W \end{pmatrix}$$

r	v_r	k	(u, v)	δ_{uv}	részkörút
1	(–, 2, 11*, 10, 8, 7, 6, 5)	3	(1, 1)	-	(1, 3)
2	(–, 2, –, 10*, 8, 7, 3, 5)	4	(1, 3)	9	(1, 4, 3)
3	(–, 2, –, –, 1, 7*, 3, 5)	6	(1, 4)	–1	(1, 6, 4, 3)
4	(–, 2, –, –, 1, –, 3, 5*)	8	(1, 6)	1	(1, 8, 6, 4, 3)
5	(–, 2*, –, –, 1, –, 1, –)	2	(4, 3)	0	(1, 8, 6, 4, 2, 3)
6	(–, –, –, –, 1*, –, 1, –)	5	(4, 2)	3	(1, 8, 6, 4, 5, 2, 3)
7	(–, –, –, –, –, –, 1, –)	7	(1, 8)	4	(1, 7, 8, 6, 4, 5, 2, 3)

A kapott körúthoz tartozó célfüggvényérték:

$$c_{17} + c_{78} + c_{86} + c_{64} + c_{45} + c_{52} + c_{23} + c_{31} = 6 + 3 + 3 + 2 + 1 + 11 + 1 + 5 = 32.$$

Feladatok

- Milyen körutat ad a Farthest insertion algoritmus az alábbi TSP feladatra és mennyi a körút költsége? Adjuk meg az egyes iterációs lépésekhez tartozó adatokat (részkörút, távolságvektor, kiválasztott város, kiválasztott él, minimális beszúrási költség) is! (Ha egy lépésben több „jó választás” is lehetséges, akkor válasszuk a „legkisebb indexűt”!)

$$TSP\begin{pmatrix} W & 2 & 4 & 2 \\ 2 & W & 2 & 3 \\ 1 & 2 & W & 4 \\ 3 & 2 & 4 & W \end{pmatrix}$$

TSP heurisztikák: Cheapest insertion

Az aktuális részkörutat azzal a k várossal bővítjük, amely minimális beszúrási költségű az összes lehetséges részkörúton kívüli várost és azok beszúrásait figyelembe véve.

A Cheapest insertion algoritmus működése:

$$\begin{pmatrix} W & 2 & 11 & 10 & 8 & 7 & 6 & 5 \\ 6 & W & 1 & 8 & 8 & 4 & 6 & 7 \\ 5 & 12 & W & 11 & 8 & 12 & 3 & 11 \\ 11 & 9 & 10 & W & 1 & 9 & 8 & 10 \\ 11 & 11 & 9 & 4 & W & 2 & 10 & 9 \\ 12 & 8 & 5 & 2 & 11 & W & 11 & 9 \\ 10 & 11 & 12 & 10 & 9 & 12 & W & 3 \\ 7 & 10 & 10 & 10 & 6 & 3 & 1 & W \end{pmatrix}$$

r	k	(u, v)	δ_{uv}	részkörút
1	2	(1, 1)	-	(1, 2)
2	3	(2, 1)	0	(1, 2, 3)
3	6	(2, 3)	8	(1, 2, 6, 3)
4	5	(2, 6)	6	(1, 2, 5, 6, 3)
5	4	(2, 5)	1	(1, 2, 4, 5, 6, 3)
6	7	(2, 4)	8	(1, 2, 7, 4, 5, 6, 3)
7	8	(2, 7)	2	(1, 2, 8, 7, 4, 5, 6, 3)

A kapott körúthoz tartozó célfüggvényérték:

$$c_{12} + c_{28} + c_{87} + c_{74} + c_{45} + c_{56} + c_{63} + c_{31} = 2 + 7 + 1 + 10 + 1 + 2 + 5 + 5 = 33.$$

Hatékonyság: Az algoritmus műveletigénye $O(n^3)$.

Megjegyzés: Olyan TSP feladatokra, amelyek költségmátrixa szimmetrikus és teljesül rá a háromszög-egyenlőtlenség, az algoritmus approximációs hányadosa 2.

Feladatok

- Milyen körutat ad a Cheapest insertion algoritmus az alábbi TSP feladatra és mennyi a körút költsége? Adjuk meg az egyes iterációs lépésekhez tartozó adatokat (részkörút, kiválasztott város, kiválasztott él, minimális beszúrási költség) is! (Ha egy lépésben több „jó választás” is lehetséges, akkor válasszuk a „legkisebb indexűt”!)

$$TSP\begin{pmatrix} W & 2 & 4 & 2 \\ 2 & W & 2 & 3 \\ 1 & 2 & W & 4 \\ 3 & 2 & 4 & W \end{pmatrix}$$

TSP heurisztikák

Tekintettel arra, hogy az ismertetett eljárások mindegyike igen gyors, továbbá az előállított körút függ a kiinduló várostól (mi ezt rendre 1-nek választottuk), ezért a heurisztikákat többször is végre lehet hajtani más és más kiinduló városokkal, majd az előálló körutak közül venni a legjobbat. Ilyenkor az illető eljárás **all cities változatáról** beszélünk.

Az alábbi táblázat adatai 5 darab 100×100-as euklideszi TSP-re vonatkoznak. (A városok egy euklideszi tér valamely síkjában vannak, és távolságuk a pontok távolsága.) A számok azt fejezik ki, hogy a közelítő megoldás hány százaléka az optimum értékének.

Nearest insertion (all cities)	118	117	118	114	120
Farthest insertion (all cities)	105	106	103	101	107
Cheapest insertion (all cities)	112	111	120	112	112

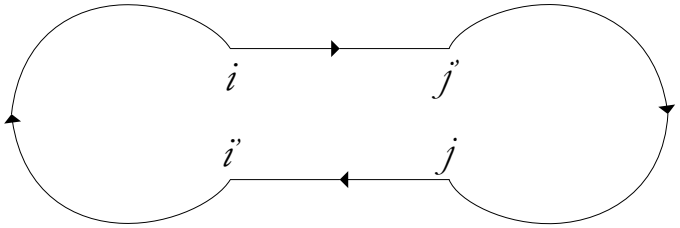
Az alábbi táblázatban a feladatok mérete rendre $n = 20, 27, 42, 57, 120$.

Nearest addition (one city)	186	113	111	114	121
Nearest insertion (one city)	138	106	110	109	117
Farthest insertion (one city)	128	100	101	101	103

TSP heurisztikák: 2-patching

Tekintsünk két részkörutat. Jelölje I, J az egyes részkörutakban szereplő városok halmazát, továbbá tetszőleges k városra jelölje k' a k-t követő várost a k-t tartalmazó részkörútban.

Legyen $i \in I$ és $j \in J$. Törölve a részkörutakból az (i, i'), (j, j') éleket, és felvéve a az (i, j'), (j, i') éleket, olyan részkörutat kapunk, amely az $I \cup J$-beli városokat tartalmazza.



A két kör összefűzésével keletkező költség: $c_{ij'} + c_{ji'} - c_{ii'} - c_{jj'}$. Vegyük ezeket a költségeket az összes $i \in I$ és $j \in J$ indexpárra, majd képezzük ezek minimumát. Az így kapott költséget **2-patching költségnek**, a részkörutak megfelelő összekapcsolását **2-patching műveletnek** nevezzük.

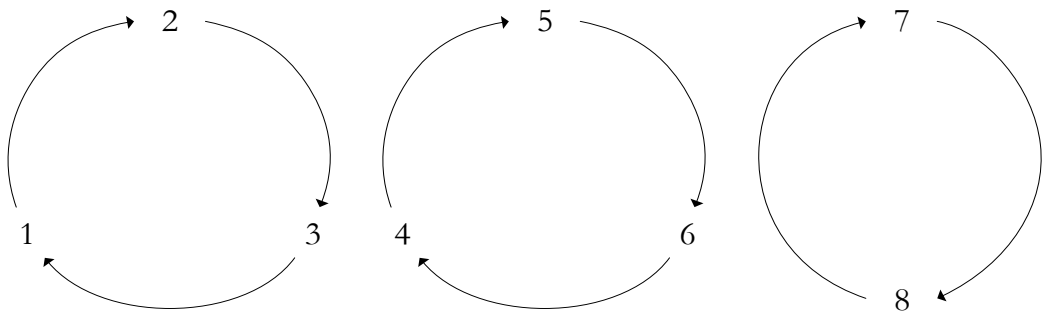
TSP heurisztikák: 2-patching

- Előkészítő rész
  - Oldjuk meg az $A(\mathbf{C})$ hozzárendelési feladatot. Ha $A(\mathbf{C})$ optimális megoldása körút, akkor vége az eljárásnak. Ellenkező esetben folytassuk az eljárást az iterációs résszel.
- Iterációs rész
  - Válasszunk ki két, legkisebb városszámú részkörutat és kapcsoljuk össze őket egy alkalmas 2-patching művelettel. Ha az előálló megoldás (hozzárendelés) körút, akkor vége az eljárásnak. Ellenkező esetben térjünk rá a következő iterációra.

Példa

A 2-patching algoritmus működése:

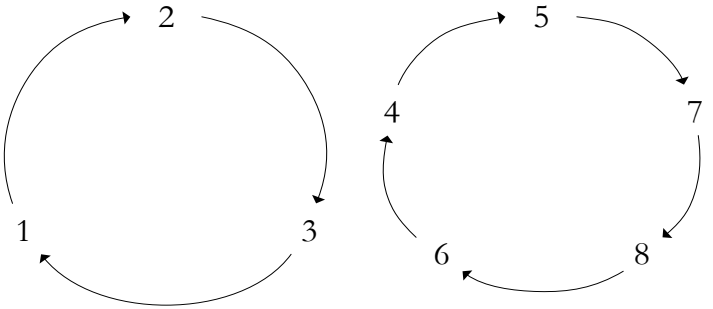
$$\begin{pmatrix} W & 2 & 11 & 10 & 8 & 7 & 6 & 5 \\ 6 & W & 1 & 8 & 8 & 4 & 6 & 7 \\ 5 & 12 & W & 11 & 8 & 12 & 3 & 11 \\ 11 & 9 & 10 & W & 1 & 9 & 8 & 10 \\ 11 & 11 & 9 & 4 & W & 2 & 10 & 9 \\ 12 & 8 & 5 & 2 & 11 & W & 11 & 9 \\ 10 & 11 & 12 & 10 & 9 & 12 & W & 3 \\ 7 & 10 & 10 & 10 & 6 & 3 & 1 & W \end{pmatrix}$$



A hozzárendelési feladat optimális megoldása a fenti három részkörútból áll. Válasszuk ki a 2. és 3. részkörutat és számoljuk ki a 2-patching költséget.

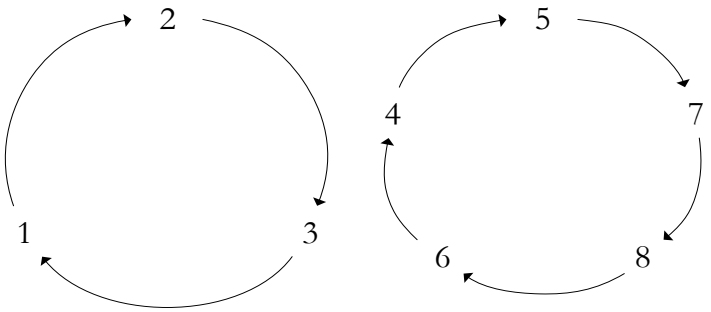
i	*j*	költség
4	7	$c_{48}^{(0)} + c_{75}^{(0)} - c_{45}^{(0)} - c_{78}^{(0)} = 10 + 9 - 1 - 3 = 15$
4	8	$c_{47}^{(0)} + c_{85}^{(0)} - c_{45}^{(0)} - c_{87}^{(0)} = 8 + 6 - 1 - 1 = 12$
5	7	$c_{58}^{(0)} + c_{76}^{(0)} - c_{56}^{(0)} - c_{78}^{(0)} = 9 + 12 - 2 - 3 = 16$
5	8	$c_{57}^{(0)} + c_{86}^{(0)} - c_{56}^{(0)} - c_{87}^{(0)} = 10 + 3 - 2 - 1 = 10$
6	7	$c_{68}^{(0)} + c_{74}^{(0)} - c_{64}^{(0)} - c_{78}^{(0)} = 9 + 10 - 2 - 3 = 14$
6	8	$c_{67}^{(0)} + c_{84}^{(0)} - c_{64}^{(0)} - c_{87}^{(0)} = 11 + 10 - 2 - 1 = 18$

A 2-patching költség 10. A megfelelő 2-patching művelet során az (5,7) és (8,6) éleket kell felvenni, és az (5,6), (8,7) éleket kell törölni. Az új megoldás az alábbi két részkörútból áll:

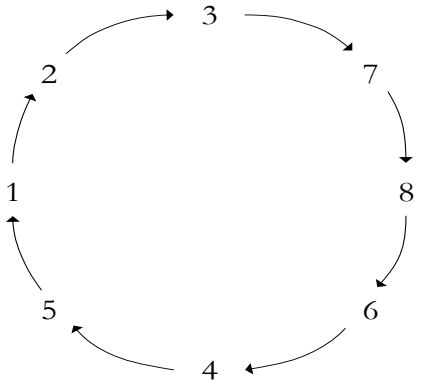


Példa

A 2-patching algoritmus működése:



Kiszámítva a fenti két körre a 2-patching költséget, az $i = 3, j = 5$ esetben kapjuk a minimumot. Törölve a (3,1), (5,7) éleket, valamint felvéve a (3,7), (5,1) éleket, az alábbi körutat kapjuk, amely éppen egy optimális megoldás.



Megjegyzés: A három részkörút (hasonló elgondoláson alapuló) egyesítésével, azaz a 3-patching műveleten alapuló algoritmus is készíthető.

TSP heurisztikák

Az alábbi táblázat adatai 100-100 darab TSP-re vonatkoznak, ahol a célfügvény együtthatók egyenletes eloszlás mellett generált véletlen egész számok a [0, 100] intervallumból.

Average ratio: az adott eljárással kapott célfüggvényértékek és az optimumértékek hányadosának átlaga.

Average sec.: az adott eljárás átlagos futási ideje (az alsó négy módszernél *all cities* változat).

Best value: az adott eljárás hányszor szolgáltatta az optimum értéket (a B&B mindig optimumot ad).

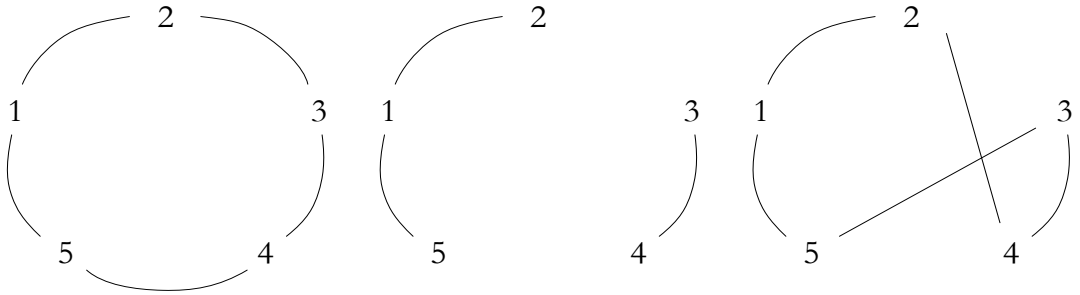
k: A patching eljárásoknál a hozzárendelési feladatok megoldása során az induló független 0-rendszerek meghatározásakor az oszlopokat véletlenszerűen választva, az előállt optimális megoldást használva a további eljárásban, mindezt *k* -szor végrehajtva, végül az előálló megoldásokból véve a legjobbat.

	n=100			*n*=250		
	Average ratio	*Average sec.*	*Best value*	*Average ratio*	*Average sec.*	*Best value*
B&B	1.000	40.78	-	1.000	320.9	-
3-patching (*k*=5)	1.054	19.53	88	1.059	370.8	80
3-patching (*k*=1)	1.069	3.84	55	1.134	72.6	24
2-patching (*k*=5)	1.090	11.14	33	1.101	158.3	37
2-patching (*k*=1)	1.108	2.21	21	1.177	31.5	12
Cheapest insertion	4.654	7.77	0	18.11	175.4	0
Nearest insertion	4.392	9.19	0	18.60	205.1	0
Farthest insertion	4.534	8.76	0	18.60	205.3	0
Nearest addition	18.43	7.09	0	98.43	149.7	0

TSP heurisztikák: 2-optimális eljárás

Feltétel: A TSP feladat költségmátrixa szimmetrikus (azaz $c_{ij} = c_{ji}, i = 1, \dots, n; j = 1, \dots, n$, ami megfelel az irányítatlan gráfok esetének.)

Észrevétel: Ha egy körút két nem szomszédos élét töröljük, akkor a körút két diszjunkt útra esik szét, és egyértelműen létezik két él, amelyekkel bővítve a két útból álló gráfot, az eredmény egy másik körút lesz.



Egy $\bar{\mathbf{X}}$ **körút szomszédjának** nevezünk minden olyan körutat, amely előáll $\bar{\mathbf{X}}$-ből két él törlésével, és két új él felvételével.

Megjegyzés: $\bar{\mathbf{X}}^{n \times n}$ szomszédjainak száma $n(n-3)/2$.

TSP heurisztikák: 2-optimális eljárás

- Előkészítő rész
  - Határozzuk meg valamilyen eljárással a feladat egy $\overline{\mathbf{X}}$ körútját.
  - Legyen $\overline{\mathbf{X}}^{(0)} = \overline{\mathbf{X}}$, $r = 0$, és folytassuk az eljárást az iterációs résszel.
- Iterációs rész (r. iteráció)
  1. Határozzuk meg $\overline{\mathbf{X}}^{(r)}$ összes szomszédját. Ha $\overline{\mathbf{X}}^{(r)}$ minden $\overline{\mathbf{X}}$ szomszédjára $z\left(\overline{\mathbf{X}}^{(r)}\right) \leq z(\overline{\mathbf{X}})$ teljesül, akkor vége az eljárásnak, $\overline{\mathbf{X}}^{(r)}$ az eljárással előállított körút. Ellenkező esetben a 2. lépés következik.
  2. Jelöljön $\overline{\mathbf{X}}$ az $\overline{\mathbf{X}}^{(r)}$ szomszédjai közül egy olyan körutat, amelyen a z függvény a szomszédokra vonatkozóan minimális értéket vesz fel. Legyen $\overline{\mathbf{X}}^{(r)} = \overline{\mathbf{X}}$, $r = r + 1$, és folytassuk az eljárást a következő iterációs lépéssel.

Megjegyzés: Kézenfekvő a 2-optimális eljárás olyan általánosítása, amikor a tekintett körútból k számú páronként nem szomszédos élt törlünk, majd a keletkező utakból k él felvételével új kört alakítunk ki, ahol $k > 1$, rögzített egész. Az ilyen heurisztikát ***k*-optimális eljárásnak** nevezzük.

Példa

A 2-optimális algoritmus működése (a törölt éleket (u, v) alakban írjuk, ahol $u<v$):

$$\mathbf{C} = \begin{pmatrix} & 3 & 4 & 1 & 2 \\ & & 2 & 5 & 3 \\ & & & 6 & 2 \\ & & & & 7 \\ & & & & \end{pmatrix}$$

Legyen $\bar{\mathbf{X}}^{(0)}$: $1-2-3-4-5-1$, ekkor $z\left(\bar{\mathbf{X}}^{(0)}\right) = 20$.

Törölt élek	$\bar{\mathbf{X}}^{(0)}$ szomszédjai	$z(\bar{\mathbf{X}})$
(1, 2), (3, 4)	$1-3-2-4-5-1$	20
(1, 2), (4, 5)	$1-4-3-2-5-1$	14
(2, 3), (4, 5)	$1-2-4-3-5-1$	18
(2, 3), (1, 5)	$1-2-5-4-3-1$	23
(3, 4), (1, 5)	$1-2-3-5-4-1$	15

Tehát $\bar{\mathbf{X}}^{(1)}$: $1-4-3-2-5-1$, $z\left(\bar{\mathbf{X}}^{(1)}\right) = 14$.

Példa

$\bar{\mathbf{X}}^{(1)}$: $1-4-3-2-5-1$, $z\left(\bar{\mathbf{X}}^{(1)}\right)=14$.

Törölt élek	$\bar{\mathbf{X}}^{(1)}$ szomszédjai	$z(\bar{\mathbf{X}})$
(1, 4), (2, 3)	$1-3-4-2-5-1$	20
(1, 4), (2, 5)	$1-2-3-4-5-1$	20
(3, 4), (2, 5)	$1-4-2-3-5-1$	12
(1, 5), (3, 4)	$1-3-2-5-4-1$	17
(2, 3), (1, 5)	$1-4-3-5-2-1$	15

$\bar{\mathbf{X}}^{(2)}$: $1-4-2-3-5-1$, $z\left(\bar{\mathbf{X}}^{(2)}\right)=12$.

Törölt élek	$\bar{\mathbf{X}}^{(2)}$ szomszédjai	$z(\bar{\mathbf{X}})$
(1, 4), (2, 3)	$1-2-4-3-5-1$	18
(1, 4), (3, 5)	$1-3-2-4-5-1$	20
(2, 4), (3, 5)	$1-4-3-2-5-1$	14
(2, 4), (1, 4)	$1-4-5-3-2-1$	15
(2, 3), (1, 5)	$1-4-2-5-3-1$	15

$\bar{\mathbf{X}}^{(2)}$ minden $\bar{\mathbf{X}}$ szomszédjára $z\left(\bar{\mathbf{X}}^{(2)}\right) \leq z(\bar{\mathbf{X}})$ teljesül, így vége az eljárásnak, a heurisztika által kapott megoldás az (1, 4, 2, 3, 5, 1) körút.

Feladatok

- Milyen körutat ad a 2-optimális algoritmus az alábbi TSP feladatra és mennyi a körút költsége? Adjuk meg az egyes iterációs lépésekhez tartozó adatokat (törölt élek, szomszédok, célfüggvényértékek) is! Az induló körút legyen az 1-2-3-4-5-1 körút! (Ha egy lépésben több „jó választás” is lehetséges, akkor válasszuk a „legkisebb indexűt”!)

$$TSP\begin{pmatrix} & 2 & 4 & 2 & 1 \\ & & 2 & 3 & 4 \\ & & & 4 & 5 \\ & & & & 3 \\ & & & & \end{pmatrix}$$

  - Megjegyzés
    - Két él esetén a „kisebb indexű” legyen az, amelyiknél a kezdőpont indexe kisebb! Ha ezek megegyeznek, akkor döntsön a végpontok indexe!
    - Két élpár esetén „kisebb indexű” legyen az, amelyiknél a „legkisebb indexű él” szerepel. Ha ez az él mindkét élpárban szerepel, akkor döntsön az élpárok másik éle!

Feladatok

- Mit mondhatunk a 2-optimális algoritmus műveletigényéről?

TSP heurisztikák: Christofides eljárása

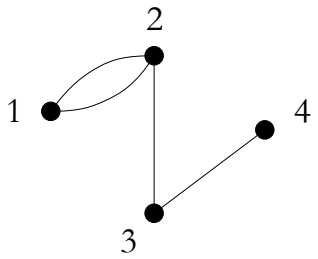
- Az eljáráshoz szükséges fogalmak

Egy $G = (V, E)$ párost **multigráfnak** nevezünk, ahol V a gráf csúcspontjainak véges, nem üres halmaza, E pedig a $V \times V \backslash I_V$ -beli élek véges rendszere, továbbá, ha $(u, v) \in E$, akkor $(v, u) \notin E$ teljesül E-re.

Megjegyzés: Az I_V a V halmaz feletti egyenlőség relációját jelöli, azaz a (v, v) $(v \in V)$ hurokéleket.

A multigráf tehát csak annyiban különbözik az irányítatlan gráftól, hogy bizonyos csúcspárokat egynél több éllel is össze lehet kötni.

Példa: Legyen $G = (\{1,2,3,4\}, \{(1,2), (1,2), (2,3), (3,4)\})$. Mivel az $(1,2)$ él kétszer szerepel az E-ben, ezért G multigráf.



Vegyük egy irányítatlan gráf vagy multigráf csúcsainak egy olyan $v_1, \ldots, v_{m+1}$ sorozatát és a gráf éleinek egy olyan $e_{i_1}, \ldots, e_{i_m}$ permutációját, hogy a v_j, v_{j+1} csúcsok illeszkedjenek az e_{i_j} élhez, továbbá legyen $v_1 = v_{m+1}$.

Az ilyen élsorozatot a gráf **Euler körének** nevezzük, és azt mondjuk, hogy a gráf **Euler-féle**, ha létezik Euler köre.

Segédtétel: Egy irányítatlan G gráf vagy multigráf akkor és csak akkor Euler-féle, ha G összefüggő és minden csúcsának páros a fokszáma.

TSP heurisztikák: Christofides eljárása

- Euler-kör készítés egy Euler-féle gráfban
  1. Induljunk egy v csúcsból egy illeszkedő élen és haladjunk addig, ameddig lehet még be nem járt éleken, közben sorszámozzuk a bejárt éleket. Csak v-ben akadhatunk el, mivel minden csúcs fokszáma páros.
  2. Ha minden élt bejártunk, akkor vége az eljárásnak, a sorszámozott élsorozat Euler-kör. Ellenkező esetben a 3. lépés következik.
  3. Van olyan be nem járt él, amely valamely bejárt u ponthoz illeszkedik mivel G összefüggő. Folytassuk innen a bejárást a be nem járt éleken u újbóli eléréséig, és írjuk át a sorszámozást úgy, hogy v-től u-ig maradnak a sorszámok, aztán u-tól u-ig majd u-tól v-ig sorszámozunk. Ezután folytassuk a 2. lépéssel.

Megjegyzés: Euler 1736-ban oldotta meg a königsbergi hidak problémáját multigráfokkal.

Egy Euler-féle multigráf Euler körének a rövidítésén pontoknak azt a sorozatát értjük, amelyet úgy kapunk, hogy a körben minden csúcsnak csak az első előfordulását hagyjuk meg, a többi előfordulását töröljük a pontok listájából.

Pl: (1, 6, 5, 4, 3, 2, 5, 2, 1) rövidítése az (1, 6, 5, 4, 3, 2).

Megjegyzés: Egy Euler kör rövidítése a pontok egy sorbarendezését adja, amelyből egy körutat kapunk, ha a kezdőpontot összekötjük a végponttal.

TSP heurisztikák: Christofides eljárása

Egy $G = (V, E)$ irányítatlan gráf vagy multigráf egy $G' = (V', E')$ részgráfját **feszítőfának** nevezzük, ha G' fa és $V' = V$.

Tekintsünk egy $G = (V, E)$ összefüggő, irányítatlan gráfot vagy multigráfot, amelyre $|V| = n$ és $|E| = m$. Továbbá tegyük fel, hogy G minden $e_t \in E$ éléhez adott egy w_{e_t} valós szám, az él hossza.

Egy feszítőfa hosszán a benne szereplő élek hosszainak összegét értjük.

Feladat: Határozzuk meg G egy minimális hosszúságú feszítőfáját.

- Kruskal eljárása

1. Legyen $T = \emptyset$, $j = 1$ és rendezzük G éleit hosszaik szerint növekvő sorrendbe. Az eredmény legyen

   $$w_{e_1} \leq \cdots \leq w_{e_m},$$

   ahol w_e jelöli az $e \in E$ él hosszát.
2. Ha $|T| = n - 1$, akkor vége az eljárásnak, $\mathcal{T} = (V, T)$ egy minimális hosszúságú feszítőfa.

   Ellenkező esetben (ha $|T| < n - 1$ és $j \leq m$) legyen $T' = T \cup \{e_{i_j}\}$.

   Ha T' nem tartalmaz kört, akkor legyen $T = T'$ (egyébként T ne változzon) .

   Legyen $j = j + 1$ és ismételjük meg a 2. lépést.

TSP heurisztikák: Christofides eljárása

Feltétel: A TSP feladat költségmátrixa szimmetrikus és teljesül rá a háromszög-egyenlőtlenség.

- Christofides eljárása

1. Határozzunk meg a feladatot leíró G hálózatban egy minimális hosszúságú feszítőfát Kruskal algoritmusával. Jelölje ezt a fát $\mathcal{T}$.
2. Legyen G' az a részgráfja G-nek, amelynek a pontjai $\mathcal{T}$ páratlan fokszámú pontjai, és az élei G azon élei, amelyek ezen pontok között mennek.

   Határozzuk meg G' egy minimális költségű teljes párosítását.

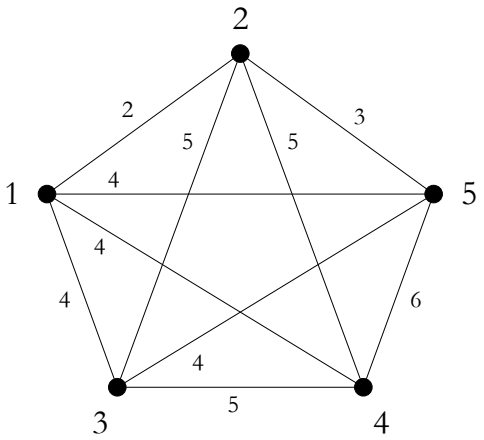
   Egészítsük ki a párosításban szereplő élekkel a $\mathcal{T}$ fát.
3. Az így kapott multigráfban határozzunk meg egy Euler kört.
4. Vegyük a kapott Euler kör rövidítését. A rövidítéshez tartozó körút a heurisztikával előállított megoldás.

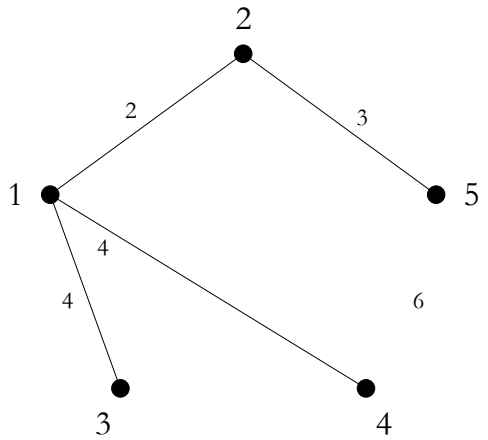
Tétel: Christofides heurisztikus algoritmusa 3/2-approximációs algoritmus.

Példa

A Christofides algoritmus működése:

$$\mathbf{C} = \begin{pmatrix} & 2 & 4 & 4 & 4 \\ & & 5 & 5 & 3 \\ & & & 5 & 4 \\ & & & & 6 \end{pmatrix}$$





Elsőként hajtsuk végre Kruskal algoritmusát. Az algoritmus a következő éleket választja ki a megadott sorrendben (1, 2), (2, 5), (1, 3), (1, 4). A kapott feszítőfa hossza 13.

A feszítőfában a páratlan fokszámú pontok 1, 3, 4, 5. Megoldva ezen pontokból és a közöttük futó élekből álló gráfra a párosítási problémát, azt kapjuk, hogy a minimális költségű teljes párosítás az (1, 4) és (3, 5) éleket tartalmazza, és a párosítás költsége 8. Tehát a 3. lépésben vizsgált multigráfnak az élei (1, 2), (1, 3), (1, 4), (1, 4), (2, 5), (3, 5).

Az Euler kört kereső algoritmus a következő pontsorozatot adja vissza: 1, 2, 5, 3, 1, 4, 1. Ezen Euler körnek a rövidítése az 1, 2, 5, 3, 4 sorrendje a pontoknak, így a Christofides algoritmus által adott körút az $1 - 2 - 5 - 3 - 4 - 1$, amely körútnak a költsége 18.