Heurisztikák: utazó szerelő probléma haszonnal (Heuristics for the Traveling Repairman Problem with Profits)

**Bevezetés**

A cikk itt ismerteti az utazó szerelő problémát, ami nagyjából a következőképp néz ki: vegyünk egy személyt (server), aki egyenletes sebességgel halad. Van *n* darab hely, mindegyik helyhez adott egy pi profit, ahol 1 ≤ *i* ≤ *n*. A *t* = 0 időpillanatban a server elindul, és minden meglátogatott helyen begyűjt (*pi - ti*) jutalmat, ahol ti az az időpont, amikor a server az *i.* helyre érkezik. Nem kötelező az összes helyet meglátogatni. A probléma lényege, hogy egy olyan utat kell találnunk a server számára, ami maximalizálja a begyűjtött jutalom összegét.

Egy életből vett példa a problémára, hogy egy katasztrófa sújtotta területen kell az egyes falvakba úgy eljuttatni a szükséges gyógyszereket, hogy a lehető legtöbb ember élje túl a katasztrófát. Van *n* db falu, az *i.* faluban *pi* embernek van szüksége a gyógyszerre (1 ≤ *i* ≤ *n*). Minden eltelt időpillanatban, mindegyik faluban meghal egy-egy olyan ember, aki még nem kapott segítséget. Tehát ha a segítség *ti* időpillanatban ér az *i.* faluba, akkor az adott faluban a túlélők száma (*pi - ti*). A feladat tehát a összeg maximalizálása.

**Matematikai modell**

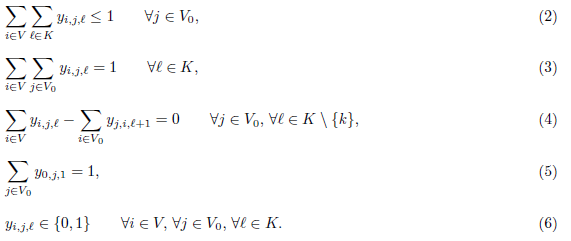
A probléma leírására gráfokat használnak a cikkben. Van egy *G* = (*V*, *E*) gráfunk, *V* csúcs- és *E* élhalmazzal. A *V* csúcshalmaz a kiindulási depót (0), és a meglátogatható helyeket (1, …, *n*) tartalmazza, az *E* élhalmaz pedig a települések közötti utakat. Minden csúcshoz, a depót leszámítva tartozik egy haszon (pi) a begyűjtött haszon egy *i* csúcsnál továbbra is (*pi - ti*). Minden *i* és *j* csúcs közötti élhez tartozik egy (utazási) idő *i*-ből *j*-be, ezt *di,j* jelöli.Egy optimális megoldás nem fog olyan élet tartalmazni, aminek a haszna kisebb-egyenlő lenne, mint a már eltelt idő (*pi* ≤ *ti)* – tehát csak hasznot hozó csúcsok lesznek a megoldásban. A feladat pedig a csúcsok egy olyan sorba rendezett részhalmazát megtalálni, ami maximális hasznot hoz.

A cikkben egy olyan matematikai modellt adtak meg, ahol a meglátogatandó csúcsok száma ismert, ez lett *k*, ennyi csúcs használt gyűjtjük be. *K*-val az {1, 2, ..., *k*} egészek halmazát jelölték.

Bevezetésre került az alábbi *yi,j.ℓ* változó is: *i* ∈ *V, j* ∈ *V0* = *V* \ {0} csúcsok és *ℓ* ∈ *K* egész szám esetén azt jelöli, hogy (*i*, *j*) él a megoldás *ℓ.* éle-e vagy sem. Az alábbi matematikai modell lett közölve a cikkben:



feltéve, hogy



Az (1) a célfüggvény összegzi a különbséget egy adott csúcs haszna és aközött a szám között, ami azt mondja meg, hogy a csúcsba vezető él hányszor került számításba az összesen eltelt idő számításakor (ha *ℓ.* élként került be). Az alábbi feltételek szerepelnek még a modellben:

* minden csúcs egyszer látogatható (2).
* *k* különböző csúcsot kell meglátogatni (3) – minden lépésre igaz, hogy egy *V*-beli csúcsból egy *V0*-beli csúcsba kell érkezni (ez azért fontos mert a depóból kell indulnunk, ami benne van *V*-ben, ugyanakkor nincs benne *V0*-ban).
* konnektivitás (4) - ha valamelyik *i* csúcsra (*i* lehet a depó is) igaz az, hogy egy belőle egy *j* (nem depó) csúcsba húzott él a megoldás *ℓ*. éle, akkor igaz az, hogy a megoldás *ℓ* + 1. éle egy *j*-ből induló másik él lesz. Az utolsó csúcsból természetesen nem húzunk már újabb élt.
* a megoldás első éle egy olyan él, amely a depóból (0) indul valamilyen nem depó *j* csúcs felé (5).
* az *yi,j.ℓ* változók binárisak – egy él vagy a megoldás *ℓ*. éle, vagy sem (6)

A cikk 5. fejezetében a fenti modellt használták az optimális megoldás megkereséséhez, valamint az LP-relaxációhoz (itt lsd. utolsó fejezet, *Eredmények*).

A fejezet további része arról szólt, hogy ahhoz, hogy a probléma optimális megoldását megtaláljuk, meg kell oldani a fenti modellt *k* összes lehetséges értékére (1 ≤ *k ≤ n*). Az itt használt jelülések:

* *k\** az optimális száma a meglátogatandó csúcsoknak, *f*\* = *f*(*k\**) a globális optimum
* *f*(*k*) az optimum abban az esetben, ha a meglátogatandó csúcsok száma *k*

Azért kell megoldani a modellt minden *k*-ra, mert:

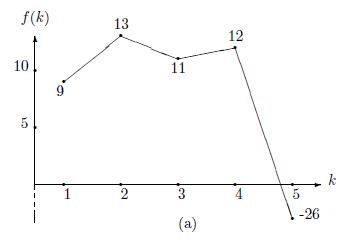
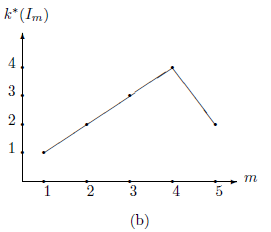
* *f*(*k*)-nak több lokális optimuma is lehet (1)
* ha gráf csúcsainak a számát növeljük, akár még csökkenthet is *k* optimális értéke, *k\** (2)

A fenti két pont szemléltetésére a cikk az alábbi példát használta:



Az első koordinálta jelöli a távolságot (és egyben az utazási időt is) a depótól, a második koordináta az adott ponthoz tartozó jutalmat jelöli.

Az (1) pontot úgy igazolták, hogy megoldották a fejezetben ismertetett matematikai modellt *k* összes lehetséges értékére (1-től 5-ig). Az egyes *k* értékekhez kapott maximális haszon az a) grafikonon látható.

Látható, hogy *f*(*k*)-nak több lokális optimuma lehet, ahogy itt is, illetve, hogy *k*\* értéke 2. Az a feltevés sem állja meg a helyét, hogy minél messzebb kerülünk *k*\* értékétől, annál rosszabb megoldást kapnánk.

A (2) pont igazolása pedig az alábbiak szerint történt (itt a b) grafikont kell nézni):

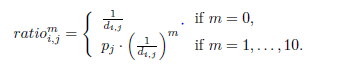
*I*m egy olyan gráfot jelöl, amely az eredeti hálózat első *m* pontját tartalmazza. *k\**(*I*m) pedig a meglátogatandó csúcsok számát az optimális, maximális hasznot hozó megoldásban, az *I*m esetén. Lehet egy olyan sejtésünk, hogy ha csúcsot adunk a hálózathoz, akkor ugyanannyi, vagy több csúcsot kell meglátogatni a maximális haszonért (optimumért). Az alábbi grafikonon látszik, hogy az új pontok hozzáadásával hogyan változott az optimumért meglátogatandó csúcsok száma. Ahogy látjuk, ugyan valóban növekedett ez az érték egy ideig az új csúcsok hozzáadásával, azonban *m* = 5 esetben nemhogy stagnált, csökkent az optimumért bejárandó csúcsok száma. Ezzel bebizonyították, hogy attól, hogy tudjuk a *k\**(*I*m) értéket egy adott *m*-re, az nem ad semmilyen információt bármely *m’* > *m* esetre vonatkozólag.

**Metaheurisztikus módszerek**

Először megadunk egy kezdő megoldást, amit aztán egy tabu kereső algoritmussal lehet javítani.

*A kezdő megoldás*

A kezdő megoldást előállító módszer egy arányszámot használ. Kezdetben az utunk/megoldásunk csak a 0 depó csúcsot tartalmazza, minden más *j* csúcs a (még meg nem látogatott csúcsok) halmaz(á)ban van. Minden lépésben a halmazból adunk egy csúcsot az úthoz. Meghatározzuk az arányszámot minden olyan csúcspárra, melynek egyik tagja már az út része *i* ∈ *V* \ , a másik *j* ∈ csúcs még nem.

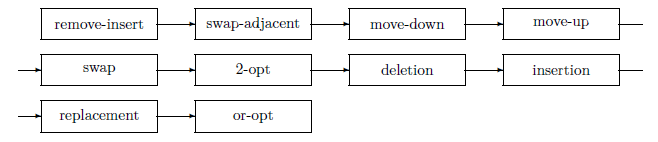


*pj* és *di,j*ugyanazt jelöli mint eddig. *m* határozza meg az (*i,j*) élhez tartozó (utazási) idő hatását az arányszámra. Azt a *j* csúcsot válasszuk be az adott körben, melyre az arányszám maximális, és úgy helyezzük el az úton belül, hogy a lehető legjobban növeljük az összegyűjtött hasznot.

*A megoldás javítása*

Egy adott megoldásból kétféleképpen lehet egy jobb megoldásra áttérni. Az egyik lehetőség a meglátogatott csúcspontok listájának megváltoztatása. A másik pedig a beválasztott csúcspontok más sorrendben való végig látogatása egy olyan sorrendben, amely rövidebb idő alatt bejárható, így kevesebb profittól esünk el. Egy új megoldásra 10-féle különböző lépéssel térhetünk át, melyek sorrendje úgy van meghatározva, hogy lehetőség szerint először csak a csúcsok végig járásának sorrendjén változtassunk, csak aztán változtassunk a csúcsok listáján. Minden iterációban egy adott fajta lépés lesz kiválasztva. Minden szomszédot megkeresünk a 10 lehetséges áttérési lépéssel, az egy bekezdéssel lentebb lévő ábrán látható sorrenden. Ha egy javító lépést találtunk, a megoldást a lehető leginkább javító szomszédot választjuk.

A cikk tartalmazott egy ábrát is a lehetséges lépések fajtáiról és azok hierarchiájáról:



A tabu keresés alapötlete az, hogy elkerüljük az egyes megoldások ismétlődését. Először is a kezdő megoldásból kiindulva keresünk, a most említett módszerrel, amíg lokális optimumba nem kerülünk.

Ezután egy hatékonyabbá tételi (intenzifikációs) fázis következik, melyet egy diverzifikációs fázis követ. Itt szerepet játszik egy *M* tulajdonságmátrix, melynek az *i,j*-edik eleme az az érték, ahányszor az (*i, j*) él az aktuális lokális optimum része volt. A diverzifikációs szakaszban arra használjuk ezt a mátrixot, hogy a gyakrabban használt élek helyett inkább a kevésbé használtakat preferáljuk, ezzel lehetővé téve, hogy a diverzifikációs fázis végén kapott megoldás elég messze legyen a megoldástérben az első néhány megoldástól. A diverzifikációs fázis után ismét a gyakran használt élek lesznek preferálva.

A következő komponens a tabu listák kompozíciója. Minden lépés típus (szomszédsági struktúra) pontosan egy listához tartozik, minden lépés után csak az ahhoz tartozó tabu lista frissül. A megállási kritériumnak két attribútuma van – az egyik az egymást követő, a megoldást nem javító lépések száma, valamint egy felső időkorlát, amit a megoldás keresésével el lehet tölteni.

*Felső korlát*

Ahhoz, hogy a tabu keresés eredményét értékelni lehessen, felső korlátot kell adni a feladatra. Feltéve, hogy ismert a meglátogatott csúcsok száma – legyen ez *k*, a következőképp jártak el.

A gráf *k*-minimális feszítőerdejének éleihez multiplicitásokat rendeltek hosszuk alapján (1, …, *k*), a leghosszabb élhez a legnagyobbat, a legrövidebbhez a legkisebbet. Az így az élekre kapott szorzatokat (*élhossz \* multiplicitás*) összeadjuk. (1)

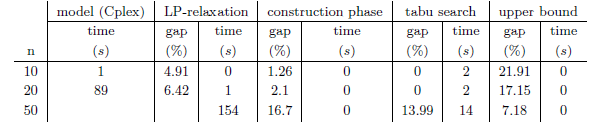
Összeadjuk a *k* legnagyobb profitú csúcs profitját. (2)

A (2) pontban kapott összegből kivonjuk az (1) pontban kapott összeget. Így kapunk egy felső határt feladatra arra az esetre, ha a meglátogatandó csúcsok száma *k.* (3)

A (3) pontban kapott értéket ki kell számolni *k* összes lehetséges értékére (1, …, *n*). Az így kapott értékeknek venni kell a maximumát. Ez a maximum lesz a feladathoz tartozó felső korlát.

**Eredmények**

Az eredmények összesítésére két táblázatot használ a cikk. Íme az első táblázat:



Az egyes sorok itt azt mutatják, hogy hogyan viselkedtek a különféle módszerek, olyan feladatok esetén, ahol a gráfban a nem depó csúcsok száma (*n*) 10, 20, illetve 50 volt (a gráfokban így az összes csúcs száma nyilván mindig *n* + 1). A matematikai modellel (ez szolgáltatja a pontos megoldást), annak LP-relaxációja, a tabu keresés kezdő megoldását előállító módszer, a tabu keresés, és a feladat megoldásának felső határát adó módszer lett összehasonlítva. A *time* oszlop a számítási időt, míg a *gap (%)* oszlop a matematikai modelltől (optimális megoldástól) való átlagos eltérést mutatja százalékban. Az utolsó sorban, n = 50-re, az optimális megoldás kiszámítása már túl sok időt vett volna igénybe, ezért ott az LP-relaxációhoz képest mutatja az átlagos eltérés mértékét.

Ami szerintem érdekes az a második sor, az *n* = 20 esetekre kapott eredmény. Látható, hogy itt a tabu keresés képes volt a valódi megoldás megtalálására, ugyanakkor kb. idő alatt végzett a matematikai modellhez képest. A táblázat alapján úgy tűnik, hogy még a tabu keresés kezdő megoldását előállító módszer is pontosabb eredményt szolgáltat, mint az LP-relaxáció, vagy mint a becsült felső határ.

A második táblázat azt mutatja meg, hogyan tudta javítani a tabu keresés az inputként kapott kezdő megoldást, illetve mekkora volt az eredményének átlagos eltérése a felső korláttól. Látható, hogy minél nagyobb volt a gráf, annál inkább sikerült javítani az inputként kapott kezdő megoldást. Ugyanakkor a felső korláttól való eltérés mértéke csak lassan növekedett.

