Eötvös Loránd Tudományegyetem
Természettudományi Kar

Mercs Erika

Matematikai módszerek a navigációban

BSc diploma

Témavezető:

Bérczi Kristóf
Operációkutatási Tanszék

Budapest, 2013

Köszönetnyilvánítás

Szeretnék köszönetet mondani témavezetőmnek, Bérczi Kristófnak, hogy folyamatosan figyelemmel kísérte munkámat, ötleteivel segített, illetve felhívta a figyelmemet az esetleges hibákra. Szeretném még megköszönni tanáraimnak, hogy hozzájárultak matematikai tudásom bővítéséhez.

Tartalomjegyzék

Bevezetés

A ma embere rengeteget utazik, és rohanó világunkban elengedhetetlen, hogy ehhez a leggyorsabb utakat válassza. Éppen ezért a navigációs készülékek egyre inkább nélkülözhetetlenné válnak, ezzel párhuzamosan rohamos fejlődésnek indultak. A dolgozat célja bemutatni az útvonalkereső algoritmusok alapjait, az alkalmazható matematikai modelleket.

Az úthálózatokat legegyszerűbb gráfokkal modellezni. Az utakat éleknek, a kereszteződéseket pedig pontoknak feleltetjük meg. Az első fejezetben az utakon determinisztikus áthaladási időkkel számolunk, azaz adott egy valós értékű, az éleken értelmezett függvény, mely megadja, hogy mennyi az adott élen való áthaladás költsége. Ezt tekinthetjük az útszakaszon való átjutás idejének. A célunk két adott pont között a legrövidebb utat megtalálni. A probléma speciális eseteire közismert algoritmusok születtek, az első fejezetben ezeket ismertetjük.

Arra az esetre, ha nincsenek negatív költségértékek az éleken, Edsger Wybe Dijkstra adott algoritmust 1956-ban. Az algoritmus felgyorsítható, ha párhuzamosan elindítjuk a kezdő, és végpontból is, ezt nevezik kétirányú Dijkstra algoritmusnak.

Ha a költségfüggvény valós értékű, de nincs negatív összköltségű kör az adott gráfban, akkor a Bellman-Ford algoritmus megoldja a feladatot. Richard Bellman 1958-ban, Lester Ford pedig 1956-ban egymástól függetlenül publikálta algoritmusát. Ha a gráfban létezik negatív összköltségű kör, nem létezik legrövidebb út két pont között, hiszen a negatív körön többször végigmenve az út összköltsége mindig csökkenthető.

Előfordulhat, hogy egy gráf valamennyi csúcspárjára szeretnénk megtudni a két csúcs közti legrövidebb utat. Ez a probléma merül fel, ha például autóstérképekhez készítjük el a városok egymástól mért távolságainak táblázatát. Természetesen alkalmazhatjuk minden pontpárra a korábban említett algoritmusokat. Ha a költségfüggvény nemnegatív, a Dijkstra algoritmus, ha negatív költségeket is megengedünk, a Bellman-Ford algoritmus alkalmazható. Az utóbbi esetben hatékonyabb módszerek is léteznek, a Floyd algoritmus a problémát dinamikus programozási feladatra vezeti

vissza. Robert Floyd 1962-ben publikálta az algoritmus azon változatát, mely csak a pontok közti legrövidebb út hosszát adja meg, az utat magát nem. Kis kiegészítéssel az algoritmusból a legrövidebb utak is rekonstruálhatók.

Az A* (A csillag) algoritmus a Dijkstra algoritmus általánosítása, szintén pozitív költségfüggvény esetén használható. Peter Hart, Nils Nilsson és Bertram Raphael írta le először 1968-ban. Az A* algoritmus heurisztikát használ a pontok cél ponttól való távolságának becslésére. Például ha két város közt keressük a legrövidebb utat, a heurisztika lehet a városok egymástól légvonalban mért távolsága. Az algoritmus a gyakorlatban is nagyon hasznosnak bizonyult, széles körben alkalmazzák útkeresésre és gráfbejárásra.

Ha azt szeretnénk, hogy valamely algoritmusunk nagy gráfok esetén is gyorsan működjön, használhatjuk a kis szeparátorok módszerét. Ez nem egy önálló algoritmus, más determinisztikus útkereső algoritmusokat lehet vele felgyorsítani. A gráf kisebb részekre partícionálásával és az útvonalkeresést a részek közötti útvonalkeresésre redukálásával csökkenti a keresés idejét.

Az egyes útszakaszokon való átjutás ideje a valóságban nem determinisztikus érték, függ a forgalmi körülményektől. A várható utazási időre pontosabb becslést kaphatunk, ha egy útszakaszon az átjutás idejét egy valószínűségi változóval modellezzük. A változó megadja, hogy mekkora valószínűséggel mennyi időbe fog telni az útszakaszon való átjutás. Tehát a modellünkben a gráf éleihez egymástól független valószínűségi változókat rendelünk. Kevésbé világos, hogy mit jelent a sztochasztikus legrövidebb út. A feladatunk már nem a leggyorsabb út megkeresése, hanem a gyorsaság (várható érték) és a megbízhatóság (szórás) valamely kombinációjának optimalizálása. Ez a kombináció az utazó preferenciáitól függ, ezért bevezetünk egy $C : \mathbb{R} \to \mathbb{R}^+$ költségfüggvényt. Ha a határidőt elnevezzük a 0 időpontnak, $C(t)$ értéke a t időpontban való érkezés költségét adja meg. A második fejezetben a költségfüggvények néhány családját fogjuk tárgyalni.

Általában az utazó célja az, hogy bizonyos határidőre érjen adott helyre. Ha például 5-kor indul a vonatom, mikor kell elindulnom otthonról, és melyik útvonalon kell haladnom, hogy elérjem és ne induljak el feleslegesen korán? Ha már úton vagyok, melyik az optimális út az adott határidőhöz? Azt gondolnánk, hogy a második kérdés a könnyebb, azonban azt fogjuk tapasztalni, hogy az általunk vizsgált költségfüggvények egy részére az első kérdés megválaszolható, míg a második kérdés NP-nehéz. Fontos különbséget tennünk a két probléma között, más indulási időre más út bizonyulhat optimálisnak. Tegyük fel például, hogy két út közül akarunk választani, és bizonyos határidőre szeretnénk megérkezni. Az első úton való áthaladás

idejének kisebb a várható értéke és a szórása, ez az út tűnik jobbnak. Azonban, ha röviddel a határidő előtt indulunk el, a lassabb és nagyobb bizonytalanságú út a hasznosabb, mert azon haladva nagyobb az esélye, hogy időben érkezünk.

A forgalom torlódása súlyos probléma, a [8] felmérés szerint nemzetközi szinten évente 4.2 milliárd órát töltünk dugóban, és 2.9 milliárd gallon üzemanyagot vesztegetünk el. Ez körülbelül 78 milliárd dollárt jelent évente. Ezért is nagyon fontos az útvonaltervező rendszerek fejlesztése, és az utazók megfelelő tájékoztatása.

1. fejezet

Determinisztikus útvonaltervezés

Ebben a fejezetben bemutatjuk a legalapvetőbb útvonalkereső algoritmusokat, melyek az utakon determinisztikus áthaladási időkkel számolva meghatároznak egy legrövidebb utat két pont között. Az algoritmusok megfogalmazása során a [2]-ben és a [7]-ben leírtakat vesszük alapul.

Az úthálózatot egy irányítatlan $G=(V,E)$ gráffal szemléltetjük, melyben az élek utaknak, a pontok pedig kereszteződéseknek felelnek meg. Az egyes útszakaszokon való áthaladási időt egy $c: E \to \mathbb{R}$ költségfüggvény írja le, mely felvehet negatív értékeket is. A továbbiakban a kiindulási pontot S-sel, míg a célállomást T-vel fogjuk jelölni. Célunk tehát egy legrövidebb $S-T$ út megtalálása. A továbbiakban legyen $|V|=n, |E|=e$. A legrövidebb út megtalálását nehezíti a negatív élsúlyok jelenléte, hiszen ha a gráf tartalmaz negatív kört, azon többször végigmenve tetszőlegesen olcsó sétát tudunk találni. A bemutatott algoritmusok ezt a nehézséget a súlyfüggvényre tett megkötésekkel küszöbölik ki.

1.1. Dijkstra algoritmus

A negatív körök kizárására az egyik lehetőség, ha az élsúlyokat nemnegatívnak választjuk. Ha a modellben szereplő költségfüggvény $c: E \to \mathbb{R}^+$, Dijkstra algoritmusa megtalálja a legrövidebb $S-T$ utat, amennyiben az létezik. Valójában akár az összes pontba meghatározza az S-ből induló legrövidebb utat.

1.1.1. Lemma. *Adott egy $c: E \to \mathbb{R}^+$ súlyfügvénnyel súlyozott $G=(V,E)$ gráf. Legyen az S pontból T-be vezető legrövidebb út $P=S,u_1,u_2,...,u_k,T$. Ekkor bármely $1 \leq i \leq j \leq k$ esetén a $P_{ij}=u_i,u_{i+1},...,u_j$ részútja P-nek legrövidebb u_i-ból v_j-be vezető út.*

Mivel a legrövidebb $S-T$ út részútja is legrövidebb út, így a lokális optimumok választásával elérhető a globális optimum.

Az algoritmushoz két halmazt használunk, az Elért, és az Eléretlen halmazokat, valamint minden pontnak van egy őse. Kezdetben az összes pont Eléretlen, az S pont őse önmaga, a többi pontnak nincs. Bevezetünk egy $d(.)$ függvényt, ami minden pontra a pont aktuális távolságát jelzi S-től, kezdetben ez S-re 0, minden más pontra végtelen. Az algoritmus mohó, minden lépésben az aktuálisan legjobbnak tűnő pontot választja. Egy általános lépésben kiválasztja azt az Eléretlen u pontot, amelynek legkisebb a $d(u)$ értéke, és azt vizsgálja. Minden olyan v szomszédjára, ami még Eléretlen, kiszámít egy új távolságértéket, $t(v) = d(u) + c(uv)$. Amennyiben $t(v) < d(v)$, $d(v) := t(v)$, a v pont őse pedig legyen u. Ezután az u pont Elért-é válik. Amint T is Elért-é válik, az algoritmus véget ér. Amennyiben az összes pont távolságára kíváncsiak vagyunk S-től, az algoritmus akkor ér véget, ha már nincs Eléretlen pont.

Az algoritmus megvalósításához használhatunk rendezetlen tömböt, ekkor a műveletigény $O(n^2)$, vagy kupacot, ekkor $O((n+e) \cdot log(n))$. Mivel ebben az esetben az élszámtól is függ a műveletigény, sűrű gráfok esetén rendezetlen tömbbel, ritka gráfok esetén kupaccal érdemes a pontokat és a hozzájuk tartozó d(.) értékeket ábrázolni, hisz ekkor a műveletigény $O(n \cdot logn)$.

1.1.1. Kétirányú Dijkstra algoritmus

Ez az algoritmus a Dijkstra algoritmust futtatja párhuzamosan a kiindulási S és a cél T pontból, ezzel felgyorsítva a keresést. Amint egy u pont mind a két irányból Elértté válik, a legrövidebb út visszakereshető.

Jelölje most $d_s(.)$ egy pont S-től való távolságát, $d_t(.)$ pedig a T-től való távolságát. Ezeket az értékeket úgy inicializáljuk, hogy $d_s(.)$ értéke S-re 0, minden más pontra végtelen, $d_t(.)$ értéke T-re 0, minden egyéb pontra végtelen. Mind a két irányú futtatáshoz tartozik egy-egy Elért, és Eléretlen halmaz, mindkét algoritmus csak a saját halmazait változtatja futás közben. Egy általános lépésben mind a két algoritmus kiválasztja azt az Eléretlen u_s valamint u_t pontot, amire a legkisebb $d_s(u_s)$ illetve a $d_t(u_t)$ távolság. Tegyük fel, hogy $d_s(u_s) \leq d_t(u_t)$, ekkor u_s pontot "kiterjesztjük" (bekerül a vizsgált pontok halmazába, és a szomszédaira új távolságértékeket számítunk) a Dijkstra algoritmussal megegyező módon. Minden v szomszédjára kiszámítjuk a $t(v) = d_s(u_s) + c(u_s v)$ értéket, és ha ez kisebb mint $d_s(v)$, akkor $d_s(v) := t(v)$ és v őse legyen u_s. Végül a u_s pont átkerül az Elért halmazba.

Hasonlóan járunk el $d_s(u_s) > d_t(u_t)$ esetén, ekkor a u_t pontot terjesztjük ki. Az algoritmus véget ér, ha egy olyan w pont válik Elértté, amely a másik irányból már Elért. Ekkor a legrövidebb $S-T$ út nem feltétlenül tartalmazza w-t.

1.1.2. Állítás. *A legrövidebb út S-ből T-be*

$$min\ d_s(x)+c(xy)+d_t(y) \tag{1.1}$$

ahol x Elért S-ből, y pedig Elért T-ből.

Bizonyítás. A min $d_s(x)+c(xy)+d_t(y)$ megtalálja a legrövidebb olyan utat, amelynek minden pontja benne van S vagy T Elért halmazában. Indirekt tegyük fel, hogy a legrövidebb $S-T$ útnak létezik legalább egy m pontja, ami mindkét irányból Eléretlen. Ebben az esetben $d_s(w) \leq d_s(m)$ és $d_t(w) \leq d_t(m)$, különben m már bekerült volna valamelyik Elért halmazba. Ez azt jelenti, hogy az m ponton átmenő $S-T$ út legalább $d_s(w)+d_t(w)$ hosszú, vagyis nem rövidebb, mint az általunk talált út. □

1.2. Bellman-Ford algoritmus

1.2.1. Definíció. Egy $G=(V,E)$ gráf esetén a c:E $\to \mathbb{R}$ költségfüggvényt konzervatívnak nevezünk, ha a gráfban nincs negatív összköltségű kör.

1.2.2. Definíció. Egy $G=(V,E)$ gráfban a P út egyszerű, ha minden pontot legfeljebb egyszer érint.

Ha a modellünkben a költségfüggvény $c: E \to \mathbb{R}$ konzervatív, a Bellman-Ford algoritmust érdemes használni. Az algoritmus azon a megfigyelésen alapszik, hogy amennyiben létezik legrövidebb $S-T$ út, létezik egyszerű legrövidebb is. Valóban, kört elhagyva nem nőhet az út költsége, mivel a súlyozás konzervatív. Egy n pontú gráfban legfeljebb $n-1$ hosszú egyszerű út lehet.

Az algoritmus első lépésben inicializálja a pontok távolságát S-től, melyet ismét jelöljön $d(.)$ (értéke minden pontra végtelen, S-re 0). Ezt követően felállít egy tetszőleges sorrendet az éleken, és minden iterációban az adott sorrendben végigmenve rajtuk próbál javítani. Legyen e egy uv él: ha $d(v) > d(u)+c(e)$, akkor $d(v)$ új értéke $d(u)+c(e)$. Az $n-1$-edik lépésben az algoritmus már biztosan megtalálta a legrövidebb $S-T$ utat, sőt, nemcsak egy T-re, hanem minden T-re. Ha az algoritmus valamelyik lépésben egyáltalán nem tud javítani, akkor is befejeződik. A lépésszáma tehát legfeljebb $ce(n-1)$.

1.3. Floyd algoritmus

A Bellman-Ford algoritmus lépésszáma függ az élszámtól, ezért sűrű gráf esetén a futásideje nagy. Ha a feltételek azonosak, de nincs feltétlenül szükség a legrövidebb útra, csak annak hosszára, a Floyd algoritmus gyorsabb. Floyd algoritmusa minden u, v pontra meghatározza a legrövidebb $u-v$ út hosszát. Sorszámozzuk meg a pontokat tetszőleges sorrendben. Jelölje $t_{u,v}(k)$ a legrövidebb u-ból v-be vezető olyan út hosszát, mely úton minden belső pont sorszáma kisebb, mint k. Első lépésként elvégezzük az inicializálást:

$$t_{u,v}(0) = \begin{cases} 0 & \text{ha } u = v, \\ c(e) & \text{ha e uv él,} \\ \infty & \text{különben.} \end{cases}$$

Ekkor könnyen látható, hogy ha minden pontra ismerjük a $t_{u,v}(k)$ értéket, akkor $t_{u,v}(k+1)$ értéke vagy változatlanul $t_{u,v}(k)$, vagy egy út ami u-ból v_{k+1}-be megy, majd v_{k+1}-ből v-be. Mivel u és v_{k+1}, valamint v_{k+1} és v között is csak $k+1$-nél kisebb sorszámú pontokon haladhat át, és ezeket már ismerjük, $t_{u,v}(k+1)$ könnyen számítható:

$$t_{u,v}(k+1) := min(t_{u,v}(k); t_{u,v_{k+1}}(k) + t_{v_{k+1},v}(k)).$$

Ekkor $t_{u,v}(n)$ adja a keresett érétket. Az algoritmus lépésszáma $c \cdot n^3$.

Ha a legrövidebb utakra is szükség van, az algoritmus módosítható, ezzel lehetővé téve a meghatározott utak rekonstruálását. Nincs szükség arra, hogy minden pontból minden más pontba eltároljuk az adott utat, elég eltárolni a legnagyobb indexű köztes csúcsot, melyen át kell haladni, ha egyik pontból el akarunk jutni egy másikba. Ekkor a rekonstruáláshoz szükséges adatok eltárolhatóak egy K $n \times n$ méretű mátrixban, ahol $K[u, v]$ a legnagyobb indexű pont, melyen keresztül kell menni az u-ből v-be vezető legrövidebb úton. Kezdetben K minden értéke 0. Ha az eredeti algoritmus új legrövidebb utat talál két pont között, akkor a K mátrixot is frissíteni kell. Végül a legrövidebb út u-ból v-be rekurzívan meghatározható: a legrövidebb út u-ból $K[u, v]$-be, aztán a legrövidebb út $K[u, v]$-ből v-be. Az 1.3 ábrán az algoritmus látható, a $t_{u,v}(k)$ jelölés helyett a $t[u][v]$ jelölést használva, a k érték jelölésére nincs szükség, a mátrix tartalma mindig felülíródik, az aktuális értéket tárolja.

```
procedure FloydRekonstrukció()
for each (u,v) pontpárra
   t[u][v] := ∞
   K[u][v] := null
for each u pontra
   t[u][v] := 0
for each (u,v) élre
   t[u][v] := c(u,v)
for k from 1 to n
   for u from 1 to n
      for v from 1 to n
         if t[u][k]+t[k][v] < t[u][v] then
         t[u][v] := t[u][k]+t[k][v]
         K[u][v] := k

function ÚT(u,v)
if t[u][v]= ∞ then
   return "nincs út"
var köztes := K[u][v]
if köztes = null then
   return " "
else
   return Út(u,köztes) + köztes + Út(köztes,v)
```

1.1. ábra. Floyd algoritmus az utak rekonstruálásával

1.4. A* algoritmus

1.4.1. Definíció. Ha $c : E \to \mathbb{R}^+$ és $h : V \to \mathbb{R}^+$, jelölje $d(u,v)$ a legrövidebb uv út költségét.

1.4.2. Definíció. A $h : V \to \mathbb{R}^+$ heurisztika elfogadható, ha semmilyen $u \in V$ esetén nem becsüli túl a távolságot a céltól, vagyis ha $d(u,T) = k$, akkor $h(u) \leq k$.

1.4.3. Definíció. A $h : V \to \mathbb{R}^+$ heurisztika konzisztens, ha minden u, v pontra $h(u) \leq d(u,v) + h(v)$.

Ha a modellben szereplő költségfüggvény $c : E \to \mathbb{R}^+$ és $h : V \to \mathbb{R}^+$ egy heurisztika a pontok céltól való távolságára, az A* algoritmus alkalmazható. A heurisztika lehet például egy térkép esetén a légvonalban mért távolság 2 pont között. Ha h elfogadható és konzisztens, az A* algoritmus garantáltan befejeződik, és optimális eredményt ad.

Az algoritmus 2 halmazt használ, az egyik az Elért, a másik az Átvizsgált pontok halmaza. Az $f : V \to \mathbb{R}^+$ függvény egy becsült érték az adott ponton átmenő legrövidebb $S-T$ út hosszára. Egy általános lépésben $f(v) = t(v) + h(v)$, vagyis az $S-v$ út minimális hosszának és az $v-T$ út becsült hosszának összege. Minél közelebb

vagyunk tehát T-hez, annál jobb közelítést ad. Kiinduláskor az S pont van csak az Elért halmazban, és $f(S) := h(S)$.

Egy általános lépésben az algoritmus kiválasztja az Elért halmazból a legkisebb $f(u)$-hoz tartozó pontot, majd u-t kiterjeszti, azaz u minden olyan v szomszédjára, mely még nem Átvizsgált, kiszámítja a $t(v)$=$t(u)$+$c(uv)$ valamint az $f(v)$=$t(v)$+$h(v)$ értékeket, majd a pontok átkerülnek az Elért halmazba. Ha az $f(v)$ és $t(v)$ kisebbek, mint az eredetiek, felülíródnak, ha nem, akkor nem változik meg az értékük. Az algoritmus véget ér, ha $f(T)$ értéke a legkisebb az Elért halmazban, ekkor $t(T) = = f(T)$. Ha a halmaz üres, de még nincs értéke a $t(T)$-nek, akkor nincs út S-ből T-be. Ez az algoritmus is mohó, hiszen minden lépésben a legjobbnak tűnő pontot választja. Ráadásul az A* algoritmus a Dijkstra egy általánosításának tekinthető, hiszen a $h \equiv 0$ választással a Dijkstra algoritmust kapjuk vissza.

Ha az algoritmus véget ér, az azt jelenti, hogy talált egy utat, aminek a valós hossza rövidebb, mint bármely másik még meg nem vizsgált útnak a becsült hossza. Mivel a becslések optimisták, ezeket az utakat már nem szükséges vizsgálni. Az elfogadhatósági kritérium miatt a megoldásként kapott út optimális. Az algoritmusnak meg kell vizsgálnia minden utat, amelynek a becsült értéke egyenlő az aktuálisan talált út hosszával.

Az A* felgyorsítható az optimalitás rovására az elfogadhatósági kritérium relaxálásával. Ekkor a megoldásként talált út hossza nem lesz nagyobb, mint az optimális út hosszának $(1+\epsilon)$-szorosa. Ekkor a talált utat ϵ-elfogadhatónak nevezzük. A relaxált algoritmusnak különböző változatai ismertek:

- A *súlyozott* A* algoritmus a $h(v)$ heurisztika helyett a $w(v) = \epsilon h(v)$ függvényt használja, ahol $\epsilon > 1$, és az új heurisztikával hajtja végre az A* lépéseit. Az algoritmus így gyorsabb, kevesebb pontot vizsgál, a megoldás pedig maximum ϵ-szorosa lesz az optimálisnak.

- A *statikus súlyozású* változatban az $f(v) = g(v) + (1+\epsilon)h(v)$ függvényt-t használja.

- A *dinamikus súlyozású* esetben $f(v) = g(v) + (1+\epsilon k(v))h(v)$, ahol

$$k(v) = \begin{cases} 1 - d(v)/N & \text{ha d(v)<N} \\ 0 & \text{egyébként} \end{cases}$$

  A $d(v)$ a v pont mélységét jelöli, N pedig egy becslés az $S-T$ út hosszára.

Az A* algoritmus a gyakorlatban is hasznosnak bizonyult, és nem csak az autók navigációs rendszerei számára. A legtöbb RTS (real-time strategy) játékban használják a mesterséges intelligencia útkereső algoritmusának, hiszen gyorsan talál jó utat valós időben. Robotokat használnak csapdába esett, vagy sérült emberek megtalálására, megmentésére, ezen gépek számára is hatékony az A* útkereső algoritmus.

1.5. Kis szeparátorok

A kis szeparátorok módszere nem egy önálló algoritmus, más determinisztikus útkereső algoritmusokat lehet vele felgyorsítani. Az úthálózatokat megadó gráfok nagy általánosságban síkgráfok, azaz az élek csak pontokban metszik egymást. Ezért a síkgráfokra kifejlesztett technikák gyakran alkalmazhatók úthálózatokon is. Például előfeldolgozással csökkenthető az algoritmusok futásideje: $O(nlog^2n)$ helyet és előfeldolgozási időt is számolva elérhetünk $O(\sqrt{n}logn)$ futásidőt irányított konzervatív síkgráf esetén [3]. Több hasonló elméleti megközelítés létezik, de a gyakorlatban ezek sokszor nehezen megvalósíthatóak, mert bonyolultak és nagy a tárigényük.

Egy gyakorlatban is jól hasznosítható megoldás az útvonalkeresés felgyorsítására a G gráf partícionálása kisebb részekre pontok egy halmazának elhagyásával. Jelölje az elhagyott pontokat V_1, a megmaradt gráfot G_0. Tekintsünk most egy $G_1=(V_1, E_1)$ segédgráfot, amelynek az élei megfelelnek olyan G-beli legrövidebb utaknak, melyek nem tartalmaznak V_1-beli pontot. Ezek után az útvonalkeresés megszorítható a G_1 gráfban történő keresésre, ha S és T is komponensek. A módszer iterálható, több szintet létrehozva. Korlátozza a lehetőségeinket, hogy a segédgráf egyre sűrűbbé válik az egyes szinteken, valamint nagy gráfokra költséges a legrövidebb utak és szeparálók megtalálása.

1.6. Főútvonal hierarchia

A főútvonal hierarchia az útvonaltervezés felgyorsítására szolgál. A célunk az, hogy nagy gráfokon is gyorsan tudjunk útvonalat keresni. Ha például Nyugat-Európát vesszük alapul, a gráfunk kb. 20.000.000 pontot fog tartalmazni. Ekkora gráf esetén már nagyon sokáig tart az eddig ismertetett módszerekkel megvizsgálni a lehetséges utakat. Az alapötlet az, hogy a legrövidebb út kis utakat általában csak lokálisan használ a kiindulási és a végpont környékén. Az út maradék része főútvonalakon halad keresztül. Vezessünk be egy H hangolási paramétert, amely megadja, hogy mennyire fogjuk csökkenteni a gráf méretét.

1.6.1. Definíció. Egy $e = (uv)$ élt főútnak nevezünk, ha léteznek olyan S és T pontok, hogy a legrövidebb $S-T$ út keresztülmegy e-n és u nincs benne S-nek a H sugarú környezetében, v pedig nincs benne T-nek a H sugarú környezetében.

Az algoritmus csoportosítja a pontokat és éleket a következő két lépést váltogatva:

(1) Összevonás: Eltávolítja az összes pontot, melynek fokszáma legfeljebb 2, kiváltja őket újonnan bevezetett élekkel.

(2) Él elhagyás: Elhagyja a nem főút éleket, vagyis azokat, amelyek legrövidebb utakban csak a forrás vagy a célponthoz közel szerepelnek.

Így létrejön egy többszintű hierarchia, a gráf mérete minden szinten egyre kisebb lesz. A főútvonal hierarchia algoritmus volt az első gyorsítási technika, ami a legnagyobb elérhető gráfokra is milliszekundumokban mérhető ideig keresett. A sikerének kulcsa, hogy ugyan szükség van előfeldolgozásra, de minden pontban csak lokálisan, ez pedig idő és helytakarékos.

2. fejezet

Sztochasztikus útvonaltervezés

A determinisztikus modell egy élen való átjutás költségét fix értékként határozza meg. Ez nem tökéletesen írja le a valóságot, egy adott úton különböző időpontokban más és más lehet az átjutás ideje. Gondolhatunk például a hétköznapi csúcsforgalomra, mikor jelentősen lelassul a forgalom. A legpontosabb becslést akkor kapjuk a menetidőre, ha minden élhez a determinisztikus érték helyett valószínűségi változók sorozatát rendeljük. Egy u élen az átjutás ideje t időpontban indulva egy eloszlást követ, amit a $Y_{u,t}$ valószínűségi változó határoz meg, ez minden t -re különböző lehet.

Most egy ennél egyszerűbb modellel fogunk foglalkozni. Legyen adott egy úthálózat, melynek megfeleltetünk egy $G = (V, E)$ gráfot, és ismét optimális utat keresünk adott S és T pontok között. Az előbbiektől eltérően most egy $e \in E$ élen való átjutás ideje az indulási időtől függetlenül valamilyen eloszlást követ, melyet egy $f_e(.)$ sűrűségfüggvényű valószínűségi változó ad meg. Ebben a fejezetben a [4] [5] és [6] cikkek eredményeit ismertetjük.

A legrövidebb út nem minden esetben a legjobb, ha a cél az utazási idő minimalizálása, vagy az időben való megérkezés valószínűségének a maximalizálása. Az utazók célja általában az, hogy egy adott időpont előtt érjenek a célállomásra, legyen ez a határidő a 0 időpont. A felhasználó preferenciáit egy költségfüggvénnyel írjuk le, melynek értéke a t időpontban való érkezés esetén $C(t)$ (t pozitív lesz késés esetén, negatív korai érkezés esetén).

Legyen $e \in E$ az utolsó él egy P úton, mely S-ből T-be megy. Ezentúl az e élen való áthaladás idejét megadó valószínűségi változót Y-nal jelöljük. A sűrűségfüggvénye f, a várható értéke pedig μ lesz. Ekkor annak a várható költsége, hogy a t időpillanatban kezdünk el áthaladni rajta, megadható a következő konvolúcióval:

$$EC_e(t) = \int_0^\infty f_e(y)C(t+y)dy.$$

Legyen mostantól $P = \{e_1, e_2, ...e_n\}$ tetszőleges $S-T$ út. Jelölje az e_i élen való áthaladás idejének sűrűségfüggvényét f_i, várható értékét μ_i, szórását pedig σ_i. Ha az éleken való áthaladás idejét megadó valószínűségi változók egymástól függetlenek, akkor egy t időpontban indulva a P úton való végighaladás várható költsége:

$$EC_P(t) = \int_0^\infty ... \int_0^\infty [f_{e_1}(y_1)...f_{e_n}(y_n)C(t+y_1+...+y_n)]dy_1...dy_n.$$

Több lehetőségünk is van az éleken való áthaladás eloszlásának megválasztására, ahogy a költségfüggvény is többféle lehet.

Tegyük fel, hogy minden élen az átjutás ideje normális eloszlást követ, valamint hogy ezek egymástól függetlenek. Független normális eloszlású valószínűségi változók összege is normális eloszlású. Ekkor a P úton való átjutási idő $t_P \sim N(\mu_P, \sigma_P)$ eloszlást követ, ahol $\mu_P = \sum_{i=1}^n \mu_i$ és $\sigma_P = \sum_{i=1}^n \sigma_i$. A normális eloszlás hátránya, hogy pozitív valószínűséggel negatív az értéke, vagyis az utazási idő lehet negatív, így nem modellezi tökéletesen a valóságot.

Ha tetszőleges helyen egy autó megjelenését Poisson folyamatként modellezzük, akkor az utazási idők Gamma eloszlásúak lesznek. A Gamma eloszlás szigorúan nemnegatív, ráadásul beállíthatunk alsó korlátot is az értékének a paraméterek megfelelő megválasztásával.

A továbbiakban a $C(t)$ költségfüggvény minimalizálása a célunk, ennek különböző változatait fogjuk vizsgálni. Minden esetben a következő kérdésekre keressük a választ:

(a) Mi az optimális út, és az optimális indulási idő?

(b) Mi az optimális út adott indulási időre?

Az utóbbi kérdés egyszerűbbnek tűnik, de széles körű azon költségfüggvények családja, ahol ez a feladat NP-nehéz.

2.1. Optimalizálás adott indulási idő mellett

Ebben a részben megmutatjuk, hogy adott indulási idő mellett az optimális út megtalálása nehéz. Szemléltetésül vegyünk egy olyan költségfüggvényt, melynek nagy az értéke korai érkezés esetén. Ha az adott indulási idő is korán van, akkor az optimális út keresése ekvivalens a leghosszabb út keresésével, ami NP-nehéz. Később azt is megmutatjuk, hogy ésszerűbb indulási időre és nem egyszerű útra is NP-nehéz megtalálni az optimális utat.

2.1.1. Tétel. *Legyen a t időpontban való érkezés költsége $C(t)$. A függvény globális minimumát jelölje t_{min}, több minimum esetén t_{min} legyen a legkisebb. Ekkor egy legolcsóbb $S-T$ út megtalálása adott indulási idő mellett NP-nehéz.*

Bizonyítás. Tegyük fel, hogy minden él hossza 1. Ekkor t időpontban indulva egy L hosszú úton való áthaladás költség $C(t+L)$. Legyen az indulási idő $t = t_{min} - (n - -n^{\epsilon})$. Tegyük fel, hogy létezik olyan út, melynek hossza $n-n^{\epsilon}$. Ekkor az adott út optimális, hiszen tetszőleges L út esetén:

$$C(t+n-n^{\epsilon}) = C(t_{min}) \leq C(t+L)$$

Mivel t_{min} a legkisebb globális minimum volt, bármely $L < n-n^{\epsilon}$ esetén szigorú egyenlőtlenség áll fenn. Tegyük fel, hogy létezik egy algoritmus, ami megtalálja az optimális utat, legyen a hossza L^*. Ekkor három eset lehetséges:

1. $L^* < n-n^{\epsilon}$ Ekkor nincs $n-n^{\epsilon}$ hosszú út. Ha lenne, akkor L^* nem lehetne optimum.

2. $L^* = n-n^{\epsilon}$ Ekkor találtunk egy $n-n^{\epsilon}$ hosszú utat.

3. $L^* > n-n^{\epsilon}$ Ekkor éleket elhagyva találunk olyan utat, ami pontosan $n-n^{\epsilon}$ hosszú.

Ez azt jelentené, hogy az algoritmusunk adott gráfban $n-n^{\epsilon}$ hosszú utat keres, amiről tudjuk, hogy NP-nehéz.

□

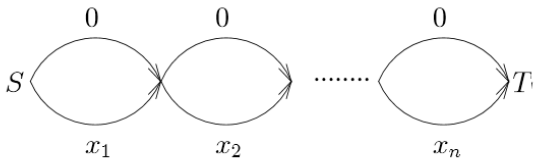
2.1.2. Állítás. *Legyen $C(t)$ szigorúan monoton csökkenő, pozitív költségfüggvény, ahol egy $[-\infty, t_{min}]$ intervallumon a $\frac{C(t_2)-C(t_1)}{t_2-t_1}$ abszolútértéke minden t_1, t_2 pár esetén nagyobb, mint $\lambda > 0$. Ekkor adott indulási idő mellett egy legolcsóbb, egyszerű út megtalálására nem létezik polinomiális konstans faktorú közelítő algoritmus.*

Bizonyítás. Legyen C_{opt} az optimális út költsége, $\alpha > 0$ pedig konstans. Tegyük fel, hogy tudunk találni $C = (1+\alpha)C_{opt}$ költségű utat. Legyenek az élek egység hosszúak, és az indulási idő $t = t_{min} - (n-1)$. Jelölje L_{opt} az optimális út hosszát, L pedig az általunk talált út hosszát. Ekkor $L < L_{opt}$, így L_{opt} a leghosszabb $S-T$ út, hiszen a költségfüggvény monoton csökkenő. Ekkor

$$\frac{C - C_{opt}}{L_{opt} - L} \leq \lambda$$

különben lenne egy pont a $[-\infty, t_{min}]$ intervallumon, ahol a $\frac{C(t_2)-C(t_1)}{t_2-t_1}$ abszolútértéke kisebb, mint λ. Ebből következik, hogy

$$\frac{L_{opt}}{L} \leq 1 + \frac{\alpha C_{opt}}{\lambda L} \leq 1 + \frac{\alpha C_{opt}}{\lambda}$$



2.1. ábra. Az optimális út megtalálása adott indulási idő mellett NP-teljes

ahol $C_{opt} = C(t_{min})$ konstans, és így polinomiális konstans faktorú közelítő algoritmust kaptunk a leghosszabb út keresési problémára, ami csak akkor lehetséges, ha $P = NP$. □

2.1.3. Tétel. *Tegyük fel, hogy a $\tilde{t}$ időpontban való megérkezés költsége $C(\tilde{t})$, és a függvény egyetlen minimumát t_{min}-ben veszi fel. Ekkor t időpontban való indulás esetén egy P út várható költsége*

$$EC_P(t) = \int_0^\infty f_P(y)C(t+y)dy$$

Annak eldöntése, hogy létezik-e olyan P $S-T$ út, amelyre $EC_P(t) \leq K$, NP-teljes.

Bizonyítás. Legyen $K = C(t_{min})$, és vegyük a 2.1 ábra gráfját determinisztikus élsúlyokkal. A felső éleken 0, az alsó éleken rendre $x_1, x_2, ..., x_n$. A $t' = t_{min} - t$ időpontban indulva bármely $S-T$ úton az átjutási idő $\sum_{i \in P} x_i$, és a költsége $C(t' + \sum_{i \in P} x_i)$. Mivel a költségfüggvénynek egyetlen minimuma van t_{min}-ben, $K = C(t_{min})$ csak akkor lehetséges, ha $t_{min} = t' + \sum_{i \in P} x_i$, vagyis, ha létezik x_i-knek olyan részhalmaza, ahol az elemek összege t. Ez éppen a jól ismert részletösszeg probléma, mely NP-teljes. □

2.2. Lineáris költség

A $C(t) = t$ választással a költségfüggvény a minimumát a legkorábbi lehetséges érkezésre veszi fel. Az e élen való áthaladás várható költsége a t időpontban való indulás mellett:

$$EC_e(t) = \int_0^\infty f(y)(t+y)dy = t + E(Y) = t + \mu$$

Ebből a P úton való áthaladás várható költsége:

$$EC_P(t) = \int_0^\infty \int_0^\infty ... \int_0^\infty f_1(y_1)f_2(y_2)...f_n(y_n)(t+y_1+y_2+...+y_n)dy_1dy_2...dy_n$$
$$= t + \sum_1^n \mu_i.$$

Leolvasható, hogy az optimális indulási idő a legkorábbi indulás, az optimális út a legkisebb várható értékű út. Továbbá adott indulási idő mellett az optimális út szintén a legkisebb várható értékű út. Tehát ez a költségfüggvény akkor adja meg az optimális utat, ha az utazó célja a lehető legkorábbi érkezés.

2.3. Exponenciális költség

Legyen $C(t) = e^{kt}$. Ez a függvény a késést nem lineárisan, hanem exponenciálisan bünteti. A k a költség növekedésének mértéke, egyéntől függő változó. Ekkor az e élen való áthaladás várható költsége, ha t időben indulunk:

$$EC_e(t) = \int_0^\infty f(y)e^{kt+y}dy = e^{kt}E(e^{kY})$$

Ebből a P úton való áthaladás várható költsége:

$$EC_P(t) = \int_0^\infty \int_0^\infty ... \int_0^\infty f_1(y_1)f_2(y_2)...f_r(y_r)e^{k(t+y_1+y_2+...+y_r)}dy_1dy_2...dy_r$$
$$= e^{kt}\prod_1^r E(e^{kY_i}),$$

ahol $E(e^{kY_i})$ a momentum generáló függvény. Vegyük először azt az esetet, mikor $Y_i \sim N(\mu_i, \sigma_i^2)$. Ekkor $E(e^{kY_i}) = e^{k\sum_i(\mu_i + k\sigma_i^2/2)}$, és így a költségfüggvény várható értéke

$$EC_P(t) = e^{kt}e^{k\sum_i(\mu_i+k\sigma_i^2/2)} = e^{k(t+\sum_i(\mu_i+k\sigma_i^2/2)}.$$

A képletből látszik, hogy az optimális út minimalizálja a $\sum_i(\mu_i + k\sigma_i^2/2)$ értéket, az optimális indulási idő pedig a lehető legkorábbi. Továbbá adott indulási idő mellett az optimális út szintén a $\sum_i(\mu_i + k\sigma_i^2/2)$ értéket minimalizálja. Ez az út megtalálható egyszerű determinisztikus legrövidebb út algoritmussal, ha az e_i él súlyának $\mu_i + k\sigma_i^2/2$ értéket választjuk.

Legyen most $Y_i \sim Gamma(\alpha_i, 1/\beta_i)$. Ebben az esetben $E(e^{kY_i}) = (1-k\beta_i)^{-\alpha_i}$, és így a költségfüggvény várható értéke

$$EC_P(t) = e^{kt}\prod_i (1-k\beta_i)^{-\alpha_i}.$$

Leolvasható, hogy az optimális út minimalizálja a $\prod_i (1-k\beta_i)^{-\alpha_i}$ értéket, az optimális indulási idő pedig a lehető legkorábbi. Ha a $-\alpha_i log(1-k\beta_i)$ érték minden e_i él esetén pozitív, az optimális út minimalizálja a $\sum_i -\alpha_i log(1-k\beta_i)$ értéket is. Ezt pedig megtalálhatjuk determinisztikus legrövidebb út algoritmusokkal, ha minden e_i él súlyának a $-\alpha_i log(1-k\beta_i)$ értéket választjuk. Adott indulási idő esetén az optimális út szintén az, amelyik minimalizálja a $\prod_i (1-k\beta_i)^{-\alpha_i}$ értéket.

2.4. Másodfokú költség

Tegyük fel, hogy $C(t) = t^2$. Ekkor az e élt vizsgálva a t időpontban való indulás várható költsége:

$$EC_e(t) = \int_0^\infty f(y)(t+y)^2 dy = t^2 + 2tE(Y) + E(Y^2) = (t+\mu)^2 + \sigma^2.$$

Hiszen $\sigma^2 = E(Y^2) - E(Y)^2$. Ebből adódik a P út várható költsége:

$$\begin{aligned} EC_P(t) &= \int_0^\infty \int_0^\infty ... \int_0^\infty f_1(y_1) f_2(y_2) ... f_r(y_r)(t+y_1+y_2+...+y_r)^2 dy_1 dy_2 ... dy_r \\ &= (t+\sum_{i=1}^{r} \mu_i)^2 + \sum_{i=1}^{r} \sigma_i^2) \end{aligned}$$

A képletből látszik, hogy minimális várható értékű út esetén $t = -\sum_{i=1}^r \mu_i$, az úton való áthaladás várható értékének -1-szerese. Ebben az esetben az út várható költsége $EC_{min} = \sum_{i=1}^r \sigma_i^2$. Tehát az optimális út az, ahol a szórásnégyzetek összege a legkisebb. Ezt megtalálhatjuk standard legrövidebb út algoritmusokkal (pl. Dijkstra), ha minden él súlyának a saját szórásnégyzetét, σ_i^2-et választjuk. Az első kérdésre a válasz tehát az, hogy az optimális út megtalálható Dijkstra algoritmusával, az indulási időt pedig ezen út várható értéke adja meg. Ezzel pedig elérkeztünk a második kérdéshez, melyik útnak minimális a várható költsége, $EC(t_{start})$, ha t_{start} adott?

2.4.1. Tétel. *Adott indulási idő mellett másodfokú költségfüggvény esetén az optimális út megtalálása NP-nehéz.*

Bizonyítás. A 2.1.3 tétel NP-nehéz feladatához hasonlóan tekintsük a 2.1 ábra gráfját. Adott $x_1, x_2, ..., x_n, b$ esetén válasszuk meg az i-edik élpáron való átjutási időket megadó valószínűségi változókat úgy, hogy az alsó élen az átjutás várható értéke x_i, a felső élen 0 legyen. A szórásokat minden élpárra válasszuk egyenlőnek, ekkor a $\sum_{i=1}^r \sigma_i^2$ érték bármely út esetén egyenlő lesz. Tehát a

$$(t+\sum_{e\in P} \mu_e)^2 + \sum_{e\in P} \sigma_e^2 = b$$

egyenletet szeretnénk megoldani adott t esetén. Mivel a szórást tartalmazó rész minden útra egyenlő, a feladat ekvivalens a

$$\sum_{e \in P} \mu_e^2 = b'$$

egyenlettel. Ha tudnánk találni ilyen utat, akkor meg tudnánk oldani a részletösszeg problémát, ami NP-teljes. □

Most egy pszeudopolinomiális algoritmus leírása következik, mely megtalálja adott indulási idő mellett az optimális utat. Tekintsük az egyszerű $S-T$ utakon való áthaladás várható idejét, legyen M a legnagyobb. Jelöljük $\pi(u,m)$-mel a u csúcs ősét azon az $S-u$ úton, aminek a várható értéke m, a szórása pedig minimális. Legyen $\Phi(u,m)$ az előbbi út szórásnégyzete. Szeretnénk minimalizálni a

$$EC_P(t) = (t + \sum_{e \in P} \mu_e)^2 + \sum_{e \in P} \sigma_e^2$$

értéket. Mivel az éleken való áthaladást megadó valószínűségi változók egymástól függetlenek, az egyes utak szórásnégyzete, az út által tartalmazott élek szórásnégyzeteinek összege. Tehát, ha a legkisebb szórásnégyzetű út S-ből u-ba áthalad v-n, akkor az a legkisebb szórásnégyzetű S-v utat használja. Inicializáljuk az értékeket:

$$\Phi(u,0) = \begin{cases} 0 & \text{ha u=S,} \\ \sigma_{Su}^2 & \text{ha S-u egy él, és } \mu_{Su} = 0, \\ 0 & \text{S többi szomszédjára.} \end{cases}$$

$$\pi(u,0) = \begin{cases} S & \text{ha u=S,} \\ S & \text{ha S-u egy él, és } \mu_{Su} = 0, \\ 0 & \text{S többi szomszédjára.} \end{cases}$$

Ezután kiszámítjuk $\pi(v,m)$-et, és $\Phi(v,m)$-et minden $0 \leqslant m \leqslant M$-re, és azon belül minden v-re, a szomszédokra való kiterjesztéssel:

$$\Phi(v,m) = min_{uv \in E}[\Phi(u, m - \mu_{uv}) + \sigma_{uv}^2]$$

$$\pi(v,m) = argmin_{uv \in E}[\Phi(u, m - \mu_{uv}) + \sigma_{uv}^2]$$

Ahol μ_{uv} a uv élen való áthaladás várható értéke, míg σ_{uv}^2 a szórásnégyzete. Ekkor az optimális út várható értéke:

$$m_{opt} = argmin_{m \in 0,\ldots,M}[(t+m)^2 + \Phi(T,m)].$$

Az optimális út:

$$\pi_{opt} = \pi(T, m_{opt}).$$

Az út várható költsége pedig

$$EC_{min}(t) = (t + m_{opt})^2 + \Phi(T, m_{opt}).$$

Ha feltesszük, hogy a vizsgált gráfban minden csúcsnak legfeljebb d szomszédja van, akkor az algoritmus futásideje $O(Mdn)$. A kapott dinamikus programozási algoritmus pszeudopolinomiális, ami az M-től függ polinomiálisan. Az algoritmus a való életben használva hatékonynak bizonyult, mert M általában kicsi a hálózat méretéhez képest.

2.5. Másodfokú + exponenciális költség

Legyen most $C(t) = t^2 + \lambda e^{kt}$, ahol $\lambda \geq 0$, és k paraméterek, melyek meghatározzák a büntetés nagyságát késés esetén. A k értéke negatív is lehet, ha jobban akarjuk büntetni a korábbi érkezést, mint a késést. Az e élt vizsgálva a t időpontban való indulás várható költsége:

$$\begin{aligned} EC_e(t) &= \int_0^\infty f(y)((t+y)^2 + \lambda e^{k(t+y)})dy \\ &= t^2 + 2tE(Y) + E(Y^2) + \lambda e^{kt} E(e^k Y) \\ &= (t+\mu)^2 + \sigma^2 + \lambda e^{kt} E(e^k Y) \end{aligned}$$

Ebből adódik az általános eset a P útra:

$$EC_P(t) = (t + \sum_{i=1}^{r} \mu_i)^2 + \sum_{i=1}^{r} \sigma_i^2 + \lambda e^k t \prod_{i=1}^{r} E(e^{kY_i}),$$

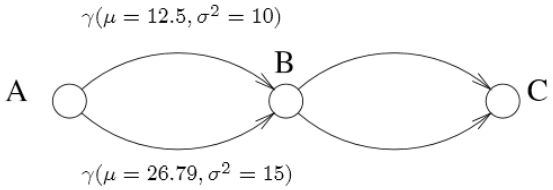
ahol $E(e^{kY_i})$ a momentum generáló függvény.

Tekintsük először azt az esetet, mikor $Y \sim N(\mu, \sigma^2)$, így $E(e^{kY_i}) = e^{k\mu + k^2\sigma^2/2}$. Ebből egy ST út várható költségére a következőt kapjuk:

$$EC_P(t) = (t + \sum_{i=1}^{r} \mu_i)^2 + \sum_{i=1}^{r} \sigma_i^2 + \lambda e^{kt} e^{k\sum_i (\mu_i + k\sigma_i^2/2)}$$

Legyen, $\tilde{t} = t + \sum_{e \in P} \mu_e$, $s = \sum_{e \in P} \sigma_e^2$. Ekkor az egyenlet új alakja:

$$EC_P(t) = \tilde{t}^2 + s + \lambda e^{k\tilde{t}} e^{k^2 s/2}$$



2.2. ábra. Optimális út keresése

Tegyük fel, hogy adott két $S-T$ út, P_1 és P_2. Legyen $s_1 < s_2$ és $k \neq 0$. Ebben az esetben fix $\tilde{t}$-re a költségfüggvényekre teljesül, hogy:

$$\tilde{t}^2 + s_1 + \lambda e^{k\tilde{t}} e^{k^2 s_1/2} < \tilde{t}^2 + s_2 + \lambda e^{k\tilde{t}} e^{k^2 s_2/2}$$

Tehát a legkisebb szórásnégyzetű útnak lesz a várható költsége a legkisebb. Ezt megtalálhatjuk legrövidebb út algoritmussal, ha élsúlynak az él saját szórásnégyzetét választjuk. A négyzetes költségfüggvény esetén is ezt kaptuk, a két költségfüggvényre nézve ugyanaz az út lesz optimális. A különbség az optimális indulási időben van, a négyzetes + exponenciális költségfüggvény esetén az indulási idő korábbi lesz.

Vizsgáljuk most azt az esetet, mikor $Y \sim Gamma(\alpha, 1/\beta)$, így $\mu_e = \alpha_e \beta_e$, és $\sigma_e^2 = \alpha_e \beta_e^2$. A Gamma eloszlás sűrűségfüggvénye

$$\gamma(y) = \frac{y^{\alpha-1} e^{-y/\beta}}{b^a \Gamma(\alpha)}$$

Ahol a $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t} dt$ a gamma függvény. Gamma eloszlás esetén $E(e^{kY}) = = (1-k\beta)^{-\alpha}$, így egy út költségének várható értéke:

$$EC_P(t) = (t + \sum_{i=1}^{r} \alpha_i \beta_i)^2 + \sum_{i=1}^{r} \alpha_i \beta_i^2 + \lambda e^{kt} \left[\prod_{i=1}^{r} (1 - k\beta_i)^{-\alpha_i} \right]$$

2.5.1. Tétel. *Négyzetes + exponenciális költségfüggvény és gamma eloszlás mellett nem feltétlenül teljesül a részút optimalitás, ezért a standard dinamikus programozási eljárások nem adnak optimális megoldást.*

Bizonyítás. Nézzünk egy konkrét példát.

Az éleken való átjutási idők legyenek egymástól független gamma eloszlású valószínűségi változók. A 2.2 ábra szerint a felső éleken $\mu = 12.5, \sigma^2 = 10$, míg az alsó éleken $\mu = 26.79, \sigma^2 = 15$. Ha a költségfüggvény $C(t) = t^2 + e^t$, akkor az optimális út A-ból B-be valamint B-ből C-be a felső éleken vezet -22.18 optimális indulási idővel és 123.14 költséggel. Az A-ból C-be vezető út a felső éleken 526.7 költségű, -46.5

indulási idővel. Azonban a minimális út A-ból C-be az alsó utakon vezet, -74.79 indulási idővel, és 522.65 költséggel. Ebből következik, hogy az optimális útnak egyik részútja sem optimális.

□

2.5.2. Tétel. *Adott indulási idő és Gamma eloszlás mellett másodfokú+exponenciális költségfüggvény esetén az optimális út megtalálása NP-nehéz.*

Bizonyítás. A másodfokú esethez hasonlóan a célunk most is az, hogy a részletösszeg problémát visszavezessük erre a feladatra a várhatóértékek megfelelő megválasztásával. Legyen adott a részletösszeg probléma halmaza $x_1, ..., x_n$ és b valamint tekintsük a legegyszerűbb költségfüggvényt, $C(t) = t^2 + e^t$. A 2.1-es ábrán lévő gráf alsó élein való átjutás idejét a $\gamma(\alpha_i, 1/\beta_i)$ eloszlások, a felső éleken pedig $\gamma(\alpha_i', 1/\beta_i')$ eloszlások adják meg. Ekkor a célunk adott t esetén olyan P út keresése, melyre

$$(t + \sum \alpha_i\beta_i)^2 + \sum \alpha_i\beta_i^2 + [\prod (1-\beta_i)^{-\alpha_i}]e^t = b$$

Azt szeretnénk, hogy tetszőleges úton végighaladva a $\sum \alpha_i\beta_i^2 + [\prod (1-\beta_i)^{-\alpha_i}]e^t$ kifejezés értéke azonos konstans legyen, így az út várható költsége csak az első tagtól függne. Keressünk tehát olyan $\alpha_i, \beta_i, \alpha_i', \beta_i'$ értékeket, melyekre:

$$\alpha_i\beta_i = z_i + q_i$$
$$\alpha_i'\beta_i' = q_i$$
$$\alpha_i\beta_i^2 = \alpha_i'\beta_i'^2$$
$$(1-\beta_i)^{-\alpha_i} = (1-\beta_i')^{-\alpha_i'}$$

teljesülnek. Ha az egyenletrendszernek létezik megoldása, akkor a feladat a

$$\sum \alpha_i\beta_i = b'$$

problémára egyszerűsödik, vagyis ekvivalens a $\sum z_i = b''$ problémával. A q_i értékekre azért van szükség, mert a γ eloszlás szigorúan pozitív, így a felső élen a várható érték nem lehet 0. Mivel minden i esetén ugyanazok a feltételek, hagyjuk el az indexelést. Az utolsó egyenletből következik, hogy $0 < \beta < 1$. Az első két egyenletből $\alpha = \frac{z+q}{\beta}$ és $\alpha' = \frac{q}{\beta'}$. Az első és a harmadik egyenletből adódik a $\beta(z+q) = \beta' q$, az első és a negyedik egyenletből $(1-\beta)^{-\frac{z+q}{\beta}} = (1-\beta')^{-\frac{q}{\beta'}}$. Legyen $k = 1 + \frac{z}{q}$, ekkor $\beta' = \beta\frac{z+q}{q} = \beta k$. Ekkor $(1-\beta)^{1/\beta} = (1-\beta')^{\frac{q}{\beta'(z+q)}} = (1-k\beta)^{\frac{1}{k^2\beta}}$.

Válasszunk tetszőleges $q_i > 0$ értékeket, ekkor $z_i := x_i - q_i$. Ha $z_i > 0$, válasszunk $k > 1$-et, pl $k = 1/0.9$, ebből adódik a $(1-\beta)^{1/\beta} = (1-k\beta)^{\frac{1}{k^2\beta}}$ egyenletből a $\beta = = 0.676948$, amiből $\beta' = \beta/0.9 = 0.752165$. Innen már α és α' értéke is egyértelmű.

Ez a k választás megfelelő minden i-re, ha $z_i > 0$. Ellenkező esetben válasszuk k-t 1-nél kisebbnek, pl $k = 0.9$, és járjunk el hasonlóan. Ha ezekkel az értékekkel meg tudjuk oldani a

$$(t + \sum \alpha_i \beta_i)^2 + \sum \alpha_i \beta_i^2 + [\prod (1 - \beta_i)^{-\alpha_i}] e^t = b$$

feladatot, akkor $\sum z_i = b''$ feladatot is, ami egy megoldást ad a részletösszeg problémára, ami NP-teljes. □

Adott t esetén az optimális út megkeresésére fogunk adni egy pszeudopolinomiális algoritmust, mely nagyon hasonlít a másodfokú költségfüggvény esetére. Ismét egy dinamikus programozási algoritmusra lehet visszavezetni a feladatot. Jelölje $\pi(u, m, \sigma^2)$ az u csúcs ősét, az S pontból u-be vezető m várható értékű, σ^2 szórásnégyzetű, és minimális $\prod_{e \in P} E(e^{Y_e})$ értékű úton. Jelölje továbbá $\Phi(u, m, \sigma^2)$ a $\prod_{e \in P} E(e^{Y_e})$ értéket az adott úton. Feltehetjük ismét, hogy M, a maximális az $S-T$ utakon való átjutási idők várható értékének, mely felülről korlátozza az utak szórásnégyzetét is, hiszen egy út várható értéke felülről korlátozza annak szórásnégyzetét. Ha már kiszámítottuk $\Phi(v, \mu, \sigma^2)$-t és $\pi(v, \mu, \sigma^2)$-t minden $\mu < m$, és $\sigma^2 = 0{,}1, ..., M$ esetén, akkor kiszámíthatjuk $\Phi(v, m, \sigma^2)$ és $\pi(v, m, \sigma^2)$ értékét:

$$\Phi(v, m, \sigma^2) = min_{uv \in E}[\Phi(u, m - \mu_{uv}, \sigma^2 - \sigma_{uv}^2) E(e^{Y_{uv}})]$$

$$\pi(v, m, \sigma^2) = argmin_{uv \in E}[\Phi(u, m - \mu_{uv}, \sigma^2 - \sigma_{uv}^2) E(e^{Y_{uv}})]$$

Ahol Y_{uv} egy valószínűségi változó, ami az uv élen való átjutás idejét adja meg. Az előző esethez hasonlóan, $\prod_{e \in P} E(e^{Y_e})$-ra teljesül az optimális-részstruktúra tulajdonság, és megkapjuk a legkisebb várható költségű utat, ha kiszámítjuk a

$$(t + m)^2 + \sigma^2 + \lambda e^{-t} \Phi(T, m, \sigma^2)$$

minimumát minden $m = 0, ...M$ és $\sigma^2 = 0, ..., M$ esetén. A minimum felvétetik valamely m_{opt} és σ_{opt}^2 esetén, és az út visszakereshető, mert az ősök el vannak tárolva π-ben. Az algoritmus futásideje $O(M^2 dn)$.

2.6. Lépcsős költség

Ebben a részben azt az utat keressük, amelyen végigmenve a legnagyobb valószínűséggel érkezünk a célba adott időre. Ez a sztochasztikus legrövidebb út probléma. Legyen mostantól a 0 időpont az aktuális idő, d pedig a határidő, amikorra meg

kell érkezni. A lépcsős függvény olyan költségfüggvény, ami csak a késést bünteti, $C(t,d) = u(t-d)$, ahol

$$u(x) = \begin{cases} 0 & \text{ha } x \leq 0, \\ 1 & \text{ha } x > 0 \end{cases}$$

t pedig az érkezési idő. Kapcsolatot fogunk létrehozni a sztochasztikus legrövidebb út problémája, és a paraméteres legrövidebb út probléma között.

Legyen G egy gráf, amelyben S a kiindulási pont, T pedig a cél. Minden i él hossza $u_i + \lambda w_i$, ahol u_i, w_i nemnegatív konstansok, $\lambda \in [0, \infty)$ paraméter. A paraméteres legrövidebb út probléma azon λ paraméterértékek, úgynevezett töréspontok megtalálása, melyekre a legrövidebb út megváltozik. Az ilyen pontok száma a legrosszabb esetben $n^{\Omega(logn)}$ [1].

2.6.1. Definíció. Egy $f : C \to (-\infty, \infty]$ függvény konvex, ha minden $x, y \in C$ és $\alpha \in$ [0,1]-re

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y).$$

2.6.2. Definíció. Egy $f : C \to (-\infty, \infty]$ függvény kvázi-konvex, ha minden alacsonyabb dimenziós részhalmaza $L_\gamma = \{x \mid x \in C, f(x) \leq \gamma\}$ konvex.

2.6.3. Definíció. Egy C halmaznak x extrém pontja, ha nem áll elő C másik két pontjának konvex kombinációjaként.

2.6.4. Definíció. Egy $C \in R^m$ halmaz dominánsa azon pontok halmaza, melyek nagyobbak, mint C egy pontja.

$$\{x \in R^m \mid \exists y \in C : x \geq y\}$$

2.6.5. Lemma. *Legyen $C \in R^m$ kompakt, konvex halmaz. Ha egy $f : C \to R$ függvény felveszi a maximumát C-n, akkor C egy extrém pontján is felveszi.*

Legyen $G = (V, E)$ gráf, S a kiindulási pont és T a cél. Az indulási idő is adott, az éleken való áthaladási időket pedig egy-egy normális eloszlású valószínűségi változó adja meg, melyek páronként függetlenek: $Y_i \sim N(\mu_i, \sigma_i)$. A P úton való áthaladás idejét megadó valószínűségi változó sűrűségfüggvényét f_P-vel jelölve az úton való áthaladás várható költsége,:

$$EC_P(t) = \int_{-\infty}^{\infty} u(t-d) f_P(t) dt = \int_{d}^{\infty} f_P(t) dt,$$

ami felírható:

$$P(t \geq d) = 1 - P(t < d) = 1 - P(\frac{t - \sum \mu_i}{\sqrt{\sum \sigma_i^2}} < \frac{d - \sum \mu_i}{\sqrt{\sum \sigma_i^2}}) = 1 - \Phi(\frac{d - \sum \mu_i}{\sqrt{\sum \sigma_i^2}})$$

alakban, ahol Φ a standard normális eloszlás eloszlásfüggvénye. Mivel Φ monoton növő, a feladat ekvivalens az argumentumának maximalizálásával. Mostantól a célunk

$$max_P \frac{d-\sum_i \mu_i}{\sqrt{\sum_i \sigma_i^2}} \tag{2.1}$$

meghatározása.

Ha az élek száma m, a célfüggvény megfogalmazható folytonos optimalizálási problémaként egy út-politópon $\mathbb{R}^m$-ben. Számozzuk meg az éleket: 1-től m-ig. Az élek minden részhalmazát ábrázoljuk az $x \in \mathbb{R}^m$ incidenciavektorával, ahol $x_i = 1$ ha az i él benne van a halmazban és $x_i = 0$ ha nincs. Ezek a részhalmazok megfelelnek az egység hiperkocka csúcsainak $\mathbb{R}^m$-ben. Az $S-T$ utak politópja az egyszerű $S-T$ utak incidenciavektorainak konvex burka, jelölje ezt Q. Ez az $\mathbb{R}^m$-beli egység hiperkocka egy részhalmaza, a csúcsai pedig a hiperkocka csúcsainak egy részhalmazát képezik. Így tehát az optimális $S-T$ út a következő probléma megoldása:

$$\begin{aligned} max \frac{d-\sum_{i=1}^m \mu_i x_i}{\sqrt{\sum_{i=1}^m \sigma_i^2 x_i}} \\ x \in Q \\ x \in \{0,1\}^m \end{aligned} \tag{2.2}$$

Tekintsük most az út-politóp vetületét a 2 dimenziós koordinátarendszerbe, ahol a vízszintes tengely a várható értéket, a függőleges tengely a szórásnégyzetet reprezentálja. Itt az $S-T$ utak meghatároznak egy konvex sokszöget, jelölje ezt K.

2.6.6. Tétel. *Tegyük fel, hogy a d határidő nem kisebb, mint a legkisebb várható értékű út várható értéke. Ekkor a (2.2) egyenlet megoldása az út-politóp vetülete dominánsának extrém pontja.*

Bizonyítás. Legyen $\mu = \sum_{i=1}^m \mu_i x_i$, $\sigma^2 = \sum_{i=1}^m \sigma_i^2 x_i$, ekkor 2.2 relaxáltja

$$\begin{aligned} max \frac{d-\mu}{\sqrt{\sigma^2}} \\ (\mu, \sigma^2) \in K \end{aligned} \tag{2.3}$$

Legyen $f(\mu, \sigma^2) = \frac{d-\mu}{\sqrt{\sigma^2}}$ és $\bar{K} = K \cap \{\mu \mid \mu < d\}$. Ekkor $\bar{K}$ nem üres, mert feltettük, hogy van egy út, melynek várható értéke kisebb mint d. A $\mu < d$, ezért $f(\mu, \sigma^2)$ az $\bar{K}$-on pozitív. Legyen $L_\gamma = \{(\mu, \sigma^2) \in R^2 \mid f(\mu, \sigma^2) \leq \gamma\}$. Ez a halmaz tartalmazza a (μ, σ^2) pontpárt, amelyekre:

$$\frac{d-\mu}{\sqrt{\sigma^2}} \leq \gamma \Longleftrightarrow \sigma^2 \geq (\frac{d-\mu}{\gamma})^2$$

Vagyis pozitív γ és $\mu < d$ értékekre L_γ konvex. Így $f(\mu, \sigma^2)$ kvázi-konvex $\bar{K}$ -en. $\bar{K}$ része az út-politóp vetületének. A 2.6.5 lemma alapján f a maximumát $\bar{K}$ egy extrém pontján veszi fel. Ráadásul $f(\mu, \sigma^2)$ monoton csökkenő mind μ, mind σ^2-ben, tehát a megoldás a vetület dominánsának egy extrém pontján vétetik föl. K bármely extrém pontja az eredeti út-politóp egy extrém pontjának a vetülete. Mivel az eredeti politópnak egész értékű koordinátái vannak, a relaxált 2.3 feladat optimális megoldása egészértékű lesz. Ezért az egész értékű 2.2 feladatnak is megoldása lesz. □

Elég tehát a vetület dominánsának extrém pontjait megtalálnunk, ezek pedig megtalálhatóak lineáris időben, hiszen minden extrém pont megoldja a

$$min\ cz$$
$$z \in K$$

lineáris feladatot, ahol $c = (c_1, c_2) \geq 0$. Vagyis minden extrém pont megfelel egy útnak, ami minimalizálja $c_1\mu + c_2\sigma^2$ -t, ahol $\mu = \mu x$ az út teljes várható értéke, σ^2 pedig a teljes szórásnégyzete. Azaz $c_1, c_2 \geq 0$ esetén megtalálhatóak legrövidebb út algoritmussal.

Ahhoz, hogy az összes extrém pontot megtaláljuk, kezdetben kiválasztjuk K legkisebb várható értékű pontját, jelölje a koordinátáit a $\mu - \sigma^2$ koordinátarendszerben $\pi_1 = (\tilde{\mu}_1, \tilde{\sigma}_1^2)$. A legkisebb szórásnégyzetű pontot jelölje $\pi_2 = (\tilde{\mu}_2, \tilde{\sigma}_2^2)$. π_1 és $\pi_2 \in \mathbb{R}^2$, $\tilde{\mu}_i$ a π_i által jelölt út várható értéke, $\tilde{\sigma}_i^2$ pedig a szórásnégyzete. Ezután kiszámítjuk a $c_1\tilde{\mu} + c_2\tilde{\sigma}^2$ értéket ha $\tilde{\mu}_2 - \tilde{\mu}_1 \neq 0$ akkor $(c_1, c_2) = (-\frac{\tilde{\sigma}_2^2 - \tilde{\sigma}_1^2}{\tilde{\mu}_2 - \tilde{\mu}_1}, 1)$ helyettesítéssel, egyébként $(c_1, c_2) = (1,0)$. Az új megoldás $\pi_3 = (\tilde{\mu}_3, \tilde{\sigma}_3^2)$ egy csúcs π_1 és π_2 között a vetület határán. Abban az esetben, ha π_3 különbözik mind π_1, mind π_2-től, megismételjük az eljárást π_1, π_3 és π_2, π_3 között. Tehát minden pontot megtalálunk a pontok számának lineáris függvényében.

2.6.7. Lemma. *Az út-politóp vetülete dominánsának extrém pontjai egyértelműen megfeleltethetőek a parametrikus legrövidebb út probléma töréspontjainak, ahol az élsúlyok $\mu_i + \sigma_i^2$.*

2.6.8. Következmény. *Az út-politóp vetülete dominánsának legfeljebb $n^{\Theta(logn)}$ extrém pontja van.*

Tehát az algoritmus futásideje legfeljebb $n^{\Theta(logn)}$ [6].

Amennyiben a d határidő kisebb, mint bármely $S-T$ út várható értéke, a feladat már nem kvázi-konvex. Ekkor a nagy szórásnégyzetű utak jelenthetnek megoldást.

Mivel a legnagyobb szórásnégyzetű út megtalálása NP-nehéz, az sejthető, hogy nincs jó polinom idejű közelítő algoritmus.

2.7. A futásidő csökkentése

Ebben a részben megmutatjuk, hogy az előbbiekben leírt algoritmust hogyan lehet felgyorsítani. Induljunk ki a (2.1) problémából, és legyen

$$\varphi(P) = \frac{d - \mu_P}{\sigma_P^2}$$

ahol μ_P a P út várható értéke, σ_P pedig a szórása. Ekkor az egyenletet átalakítva kapjuk

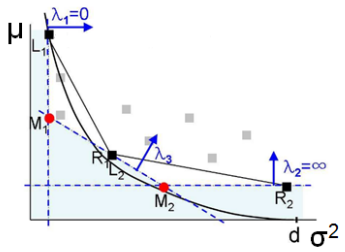
$$\sigma_P^2 = \frac{1}{\varphi(P)^2}(\mu_P - d)^2$$

alakot. A $\mu - \sigma^2$ koordinátarendszerben ez egy parabola egyenlete, melynek csúcsa a $(d,0)$ pont. A célunk, a $\varphi(P)$ érték maximalizálása, amit a parabola görbülete ad meg. Vagyis a legkisebb görbületű út az optimális. Intuitívan ez azt jelenti, hogy a $(d,0)$ csúcspontú parabolát emelni kezdjük vízszintes helyzetből, és az első út ami metszi lesz az optimális.

2.7.1. Definíció. A gráf élein tekintsünk egy c_λ determinisztikus költségfüggvényt, az élen való áthaladási idő várható értékének és a szórásnégyzetének lineáris kombinációját: $c_\lambda(i) = \mu_i + \lambda\sigma_i^2$. Erre a költségfüggvényre tekintett optimális utat λ-optimális megoldásnak nevezzük.

Azt már tudjuk a 2.6.6 tételből, hogy az optimális megoldás K egy extrém pontja. Az extrém pontok mind λ-optimális megoldások, ráadásul megtalálhatók determinisztikus legrövidebb út algoritmussal. Az előző szakaszban leírt algoritmus egyszerűen felsorolja az összes extrém pontot. Először megkeresi a λ-optimális utat $\lambda{=}0$-ra és $\lambda{=}\infty$-re. Ha ezek megegyeznek, akkor az út optimális, ha nem, megkeresi a λ-optimális utat $\lambda = -\frac{\mu_0 - \mu_\infty}{\sigma_0^2 - \sigma_\infty^2}$-re. Ha nem találtunk új utat, a legkisebb φ értékű út az optimális. Ha új utat találtunk, az a keresési régiót két részre osztja, és mind a két régióban újra keresünk. A következő részben megmutatjuk, hogy kevesebb λ értékre is elég megkeresni a λ-optimális utat.

Tegyük fel, hogy már megtaláltuk a λ_i és a λ_j-optimális megoldásokat, melyek szomszédosak. A $\mu - \sigma^2$ koordinátarendszerben ezeket jelölje L_i és R_i. A $\mu + \lambda_i\sigma^2$ egyenes átmegy L_i-n, a $\mu + \lambda_j\sigma^2$ egyenes átmegy R_i-n, és a két egyenes metszi egymást, nevezzük ezt a pontot M_i-nek. Ekkor $L_iM_iR_i$ egy háromszög, melynek M_i a középső csúcsa.



2.3. ábra. Jelölt régiók: $L_1M_1R_1$, $L_2M_2R_2$ és próba pontok: M_1, M_2

2.7.2. Definíció. Jelölt régiónak nevezzük a most definiált háromszög alakú területet, ahol jobb út létezhet.

2.7.3. Definíció. Próba pontnak nevezzük a jelölt régió középső csúcsát.

A 2.3 ábrán egy példa látható, az első három λ-optimális út megtalálása utáni állapotról. Minden négyzet egy λ-optimális utat jelöl, a feketéket már megtalálta az algoritmus, a szürkéket még nem. A kék terület biztosan nem tartalmaz utat, a fehér háromszögek jelölt régiók, a piros körök próba pontok.

Ha tudjuk, hogy egy adott régió lehető legjobb útjának a φ értéke is kisebb, mint a jelenlegi optimális, akkor nem kell futtatnunk a régióban a λ-optimális utat kereső algoritmust. A következő részben ezt a gondolatot formalizáljuk.

2.7.4. Tétel. *Ha a próba pont φ értéke kisebb, mint az jelenlegi optimális érték, a jelölt régió nem tartalmazza az optimális utat.*

Bizonyítás. Tegyük fel, hogy van egy út az adott próba pontban. Ekkor nem lehet másik extrém pont a próba ponthoz tartozó jelölt régióban, belső pont pedig nem lehet optimális a 2.6.6 tétel miatt. Abban az esetben, ha nincs út a próba pontban, képzeljünk oda egyet. Ez a pont nem fogja befolyásolni az optimális út megtalálását, hiszen nem jobb, mint a jelenlegi optimális. A jelölt régió pontjai pedig az előző gondolatmenet miatt nem lehetnek optimálisak. □

Tehát nem kell foglalkoznunk azzal jelölt régióval, melynek a próba pontjára teljesül az előző tétel feltétele. Továbbá megkötéseket tehetünk a λ érétkeire is.

2.7.5. Állítás. *Legyen λ_u a 0-optimális és a ∞-optimális keresések egyeneseinek metszéspontjában a parabola meredekségének negatív inverze. Az optimális út megtalálható, ha a λ értékeket felülről korlátozzuk λ_u-val.*

2.7.6. Állítás. *Legyen λ_l az aktuális λ-optimális parabola és a 0-optimális keresési egyenes metszéspontjában a λ-optimális parabola meredekségének negatív inverze. Az optimális út megtalálható, ha a λ értékeket alulról korlátozzuk λ_l-lel.*

Mindkét állítás bizonyítása megtalálható [4]-ben. Ezen észrevételek használata az algoritmusban felgyorsítja a keresést, mert nem hív meg szükségtelenül determinisztikus legrövidebb út algoritmust.

A javított parametrikus keresési algoritmus első lépésben elvégzi az inicializálást, az aktuális legjobb út még nem létezik, a régiókat tartalmazó sor pedig üres. Ezután kiszámítja a 0-optimális és a ∞-optimális utakat. Ha a két út megegyezik, az algoritmus végetér, megtaláltuk az optimális utat. Ha nem, a régiók sorába bekerül a két út által meghatározott jelölt régió. Ezután kiszámítjuk a két korlátot, λ_u-t és λ_l-et. Amíg a régiók sora nem üres, a következő lépéseket ismételjük:

(1) Kiveszünk egy elemet a sorból.

(2) Ha a próbapontjának φ értéke kisebb, mint az aktuális legjobb úté, visszalépünk az előző pontra.

(3) Kiszámítjuk λ értékét: $\lambda = -\frac{\mu_l - \mu_r}{\sigma_l^2 - \sigma_r^2}$.

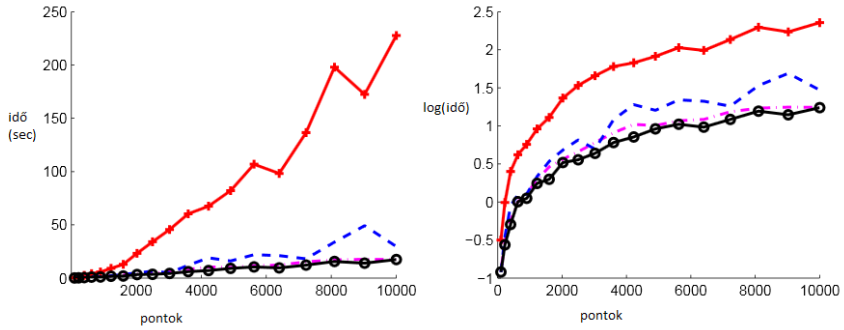
(4) Ha $\lambda \geq \lambda_u$
   a) és még nem kerestünk λ_u-optimális utat, $\lambda := \lambda_u$
   b) és már kerestünk λ_u-optimális utat, visszalépünk az első pontra.

(5) Ha $\lambda \leq \lambda_l$
   a) és még nem kerestünk λ_l-optimális utat, $\lambda := \lambda_l$
   b) és már kerestünk λ_l-optimális utat, visszalépünk az első pontra.

(6) Megkeressük a λ-optimális utat.

(7) Ha a talált út nem egyezik meg a saját jelölt régióját meghatározó utak egyikével sem, akkor a következő lépéseket hajtjuk végre:
   a) ha az út φ értéke nagyobb, mint az aktuális legjobbé, lecseréljük a legjobbat;
   b) kiszámítjuk az út által létrehozott 2 jelölt régió próbapontjait;
   c) amelyik próbapontoknak nagyobb a φ értéke a jelenlegi legjobb út φ értékénél, annak a jelölt régiója bekerül a régiók sorába.

Ezen algoritmus átlagos futásideje $O(n^2 log^4 n)$, ahol az n^2-es tag a Dijkstra algoritmus lépésszáma a λ-optimális út megtalálására. Az algoritmust [4]-ben teszteknek



2.4. ábra. Az algoritmus futásának tesztje.

vetették alá egy valós úthálózaton, a térkép kb. 29000 ponttal és nagyjából 39000 éllel rendelkezett. A gráf élein random 0 és 1 közti várható értékű és szórásnégyzetű valószínűségi változókkal számolva, az eredmény a 2.4 ábrán látható. A piros jelzi az eredeti algoritmust, a lila csak a jelölt régiók vizsgálatával történő javítást, a kék csak a λ korlátozásával történő javítást, a fekete pedig a tényleges javított algoritmust. A javítás legalább 10-szeres az előző algoritmushoz képest, ami annak köszönhető, hogy kevesebb λ-optimális útkeresést kell végrehajtani. 10000 pont esetén a λ-optimális keresések száma az eredeti algoritmusban 119, míg a javítottban 7. A javított algoritmus futásideje 144 km hosszú útvonal tervezése és 3 órás határidő esetén 14 másodperc 5 λ-optimális kereséssel.

Irodalomjegyzék

[1] P. J. Carstensen. Complexity of some problems in parametric linear and combinatorial programming. *Dissertation Abstracts International Part B: Science and Engineering[DISS. ABST. INT. PT. B- SCI. & ENG.],*, 44(2), 1983.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. Algoritmusok. *Műszaki Könyvkiadó, Budapest*, 1997.

[3] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72(5):868–889, 2006.

[4] S. Lim, H. Balakrishnan, D. Gifford, S. Madden, and D. Rus. Stochastic motion planning and applications to traffic. In *Algorithmic Foundation of Robotics VIII*, pages 483–500. Springer, 2009.

[5] E. Nikolova, M. Brand, and D. R. Karger. Optimal route planning under uncertainty. In *Proceedings of International Conference on Automated Planning and Scheduling*, 2006.

[6] E. Nikolova, J. A. Kelner, M. Brand, and M. Mitzenmacher. Stochastic shortest paths via quasi-convex maximization. In *Algorithms–ESA 2006*, pages 552–563. Springer, 2006.

[7] P. Sanders and D. Schultes. Engineering fast route planning algorithms. In *Experimental Algorithms*, pages 23–36. Springer, 2007.

[8] D. L. Schrank and T. J. Lomax. *The 2007 urban mobility report.* Texas Transportation Institute, Texas A & M University, 2007.