Szenzorlokalizációs algoritmusok kombinatorikus vizsgálata

MSc Szakdolgozat

Kohut Lilla

Alkalmazott matematikus MSc

Számítástudomány szakirány

Témavezető:

Kiss Attila

Gépi Érzékelés Kutatólaboratórium

Számítástechnikai és Automatizálási Kutató Intézet

Magyar Tudományos Akadémia

Eötvös Loránd Tudományegyetem,

Természettudományi Kar

2018

Tartalomjegyzék

1. Bevezetés **3**

2. A szenzorlokalizációs probléma **6**

3. Az algoritmusok csoportosítása **9**

4. Horgonypontot használó algoritmusok **12**

4.1. Az AHLoS algoritmus 12

4.1.1. Az algoritmus 12

4.2. Kereszt-entrópia módszerén alapuló algoritmus 15

4.2.1. Az algoritmus 16

4.2.2. Hogyan módosíthatunk az algoritmuson? 18

5. Horgonypont nélküli algoritmusok **22**

5.1. MDS 22

5.1.1. Az algoritmus 22

5.2. Az AFL algoritmus 25

5.2.1. Az algoritmus 25

5.2.2. Hogyan módosíthatunk az algoritmuson? 29

5.3. MDS-MAP(P) 31

5.3.1. Az algoritmus 32

5.3.2. Hogyan módosíthatunk az algoritmuson? 33

6. Ortogonális Prokrusztész probléma **36**

7. Összefoglalás **38**

8. Hivatkozások **39**

1. Bevezetés

A vezeték nélküli technológia fejlődése lehetővé tette olyan olcsó, kisméretű szenzorok alkalmazását, melyek rövid távolságon belül rádiókapcsolaton keresztül képesek egymással kommunikálni. Egy vezeték nélküli szenzorhálózat több száz, akár több ezer érzékelőből is állhat, melyek információt gyűjtenek a környezetükben végbemenő folyamatokról. Az adatgyűjtés mellett a szenzorok különféle számításokat is végezhetnek, illetve az így nyert adatokat képesek memóriájukban eltárolni.

A vezetéknélküli szenzorhálózatok rengeteg területen alkalmazhatóak, többek között biztonsági megfigyelésre, meteorológiai adatgyűjtésre, illetve egészségügyi és katonai célokra is.

Észak-Ausztráliában, a Burdekin folyó mentén vezetéknélküli szenzorhálózatot használnak, hogy tanulmányozzák a folyó vizével való öntözés hatásait [1]. Az amerikai egyesült államokbeli Maine államhoz tartozó egyik szigeten 32 szenzort helyeztek el, melyek lehetővé teszik a tengeri madarak élőhelyének és életmódjának megfigyelését [2].

Az épületekben elhelyezett kamerák, füstérzékelők és egyéb szenzorok vészhelyzet esetén segítséget nyújtanak az emberek gyors kimenekítésében [3]. Shen és társai a helyszínt egy irányított gráfnak tekintik, és ebben a gráfban egy folyammal modellezik az emberek mozgását [4]. Maximális folyam és minimális vágás megkeresésével a veszélyes, de a mentési folyamat szempontjából kulcsfontosságú területekhez tudják irányítani a tűzoltókat, akik az akadály elhárítása után könnyebben és gyorsabban tudják kimenekíteni az épületben rekedt embereket.

A Pickberry Szőlőskert egy körülbelül 12 hektárnyi területen helyezett el szenzorokat [5], melyek adatokat szolgáltatnak a páratartalomról, szélről, valamint a föld és a levegő hőmérsékletéről. A begyűjtött adatok és az időjárás-előrejelzések segítségével a szőlészet hatékonyabban képes eldönteni mennyi vizet érdemes öntözésre fordítani, hogy a termés a lehető legjobb legyen.

A vezetéknélküli szenzorhálózat egy másik lehetséges alkalmazási területe az erdőtüzek jelzése [6]. Az érzékelt hő, páratartalom, füst, valamint szélirány és -sebesség segítségével még az előtt jelezhetjük az erdőtüzet, hogy az széles körben elterjedne, és akár emberéleteket is veszélyeztetne.

Azonban akármilyen területen alkalmazzuk is a szenzorokat, szükségünk van rá, hogy pontosan tudjuk honnan is érkezik az adat, azaz ismernünk kell a pontos helyzetüket.

Könnyű dolgunk van, ha a szenzorokat előre meghatározott helyre telepítjük, azonban a gyakorlatban nem mindig ez a helyzet. Jó példa erre az állatok élőhelyének megfigyelése, melynek során a nehezen megközelíthetőség miatt gyakran repülőgépből ledobott szenzorokat használnak, melyek helyzete földet érés előtt nem meghatározható. Mozgó szenzorok alkalmazásánál pedig szükség van a szenzor pozíciójának folyamatos frissítésére.

Kézenfekvő megoldás lenne a szenzorok GPS-vevővel való felszerelése, azonban ezzel a megoldással több probléma is van. Nagyszámú szenzorok esetén igen költséges lenne az összes érzékelő GPS-vevővel való ellátása, valamint energiaigénye miatt jelentősen megrövidítené a szenzorok élettartamát. Emellett zárt térben, vagy valamilyen akadály következtében a műholdakkal való kommunikáció nem mindig megoldható. Ezért szükség van valamilyen lokalizációs technikára.

A szenzorlokalizáció célja, hogy a számunkra elérhető információkból (szenzorok közötti kommunikáció megléte, szenzorok távolsága, ismert helyzetű szenzorok) meghatározzuk a szenzorok pontos helyzetét.

A szakdolgozatomban bemutatom a szenzorlokalizácis problémát, és felvázolom, hogy a probléma megoldását célzó algoritmusok milyen megközelítést alkalmaznak. Bemutatom, milyen szempontok alapján lehet csoportosítani az algoritmusokat, majd öt algoritmust részletesebben be is mutatok, és ezek közül háromnál vizsgálom, hogyan lehetne javítani az algoritmuson, illetve ismertetem, hogy a módosítás milyen kompromisszummal jár az algoritmus futásideje és hibája között. Végül bemutatom az ún. ortogonális Prokrusztész problémát, illetve a megoldására szolgáló módszert, mely fontos szerepet

kap azoknál az algoritmusoknál, melyeknél csak a szenzorok relatív helyzetét tudjuk meghatározni, azonban valamiképp mérni szeretnénk az algoritmus hibáját.

Az ismertetett algoritmusokat és módosításaikat MATLAB-ban leprogramoztam, a szakdolgozatban szereplő, az algoritmusok eredményeit bemutató ábrák ennek eredményei.

2. A szenzorlokalizációs probléma

Egy szenzorhálózat jól reprezentálható egy $G = (V, E)$ gráffal, melynek csúcsai a szenzorok, és két csúcs akkor szomszédos, ha a nekik megfelelő szenzorok egymás kommunikációs távolságán belül találhatóak. Ha nem csak a szenzorok közötti kommunikáció megléte ismeretes, hanem a közöttük lévő távolság is, akkor az élekre a távolságoknak megfelelő $d(u, v)$ költségek írhatóak ($u, v \in V$).

Célunk meghatározni a szenzorok pontos helyzetét a sík (tér) egy előre rögzített koordinátarendszerében az elérhető adatok (pl. a szenzorok egymástól való távolsága, esetleg néhány szenzor már kezdetben ismert helyzete) alapján.

A továbbiakban a kétdimenziós problémát tekintjük, azaz egy olyan $p : V \to \mathbb{R}^2$ projekciót szeretnénk meghatározni, amelyre teljesül, hogy $\|p(i) - p(j)\| = d(i, j)$, ahol $d(i, j)$ az i és j csúcsoknak megfelelő szenzorok euklideszi távolsága, és $(i, j) \in E$.

Hasznosak lesznek az alábbi fogalmak:

2.1. Definíció (Hop-távolság). *Két szenzor hop távolsága a köztük lévő adatküldéshez szükséges élek száma, azaz a gráfban az őket összekötő legrövidebb út hossza.*

2.2. Definíció (Horgonypont). *A rögzített koordinátával rendelkező csúcsokat horgonypontnak nevezzük.*

A továbbiakban a következő jelöléseket fogjuk alkalmazni:

$(x_i, y_i) \in \mathbb{R}^2$: az i. szenzor koordinátái

$(\hat{x}_i, \hat{y}_i) \in \mathbb{R}^2$: az i. szenzornak az algoritmus által meghatározott koordinátái

$(x_{a_i}, y_{a_i}) \in \mathbb{R}^2$: az i. horgonypont koordinátái

$d_{ij} \in \mathbb{R}$: az i. és j. szenzor távolsága

$\hat{d}_{ij} \in \mathbb{R}$: az i. és j. szenzor távolsága az algoritmus által kiszámolt pozíciójuk alapján

$h_{ij} \in \mathbb{N}$: az i. és j. szenzor hop távolsága

A szenzorok közötti távolságokat a TDOA (Time Difference Of Arrival)[7][8], TOA (Time Of Arrival)[27] vagy RSSI(Received Signal Strength Indicator)[9] módszerek egyikével határozhatjuk meg (ez a hardvertől függ). Néhány algoritmus a szenzorok által leadott rádiójelek érkezési szögének ismeretében határozza meg a szenzorok helyzetét (AOA - Angle Of Arrival)[10]. A szenzorok hop távolságát a Dijkstra vagy Floyd-Warshall algoritmus egyikével tudjuk meghatározni.

Ívmetszés segítségével az euklideszi síkon egy pontnak egyértelműen meg tudjuk határozni a helyzetét, ha ismerjük legalább három fix helyzetű, nem kollineáris ponttól vett távolságát. Azonban a gyakorlatban a mérési pontatlanságoknak köszönhetően, az ismert helyzetű szenzorok és a tőlük vett távolságok segítségével felírható egyenletrendszernek nem mindig létezik megoldása. Ezt kiküszöbölendő a lokalizációt felfoghatjuk optimalizálási feladatként is. A szenzorok egymástól és a horgonypontoktól vett távolságai alapján felírhatunk egy egyenlőtlenség-rendszert. Ezt kiegészítve egy minimalizálandó hibafüggvénnyel egy konvex optimalizálási feladatot kapunk.

Egy másik megközelítési lehetőség az ún. multidimensional scaling (MDS) [22], melynek lényege, hogy a pontok távolsága (távolságmátrixa) alapján egy olyan vetítést határozunk meg (esetünkben a 2-dimenziós síkba), mely minél

kisebb hibával közelíti a pontok páronkénti távolságát. Az MDS algoritmussal az 5. fejezetben részletesebben is foglalkozunk.

3. Az algoritmusok csoportosítása

Range-free vagy range-based?

Az alapján, hogy az algoritmus a szenzorok hop távolságát vagy az euklideszi távolságát használja a lokalizációhoz, beszélhetünk ún. range-free ([11], [12]), illetve range-based ([13],[14]) algoritmusról (Az utóbbiakhoz soroljuk az érkezési szöget használó algoritmusokat is). A legtöbb range-free algoritmus range-based algoritmusként is működik, ha a hop távolságot euklideszi távolsággal helyettesítjük.

Centralizált vagy elosztott számítási szerkezetű?

A centralizált lokalizációs algoritmusok ([15], [16]) a szomszédaiktól mért távolságukat egy központi állomásnak továbbítják, amely aztán egymaga határozza meg az összes szenzor koordinátáit. Ezzel szemben az elosztott számítási szerkezetű ([17], [18]) algoritmusoknál minden szenzor a saját helyzetének meghatározásáért felelős. Kevesebb kommunikációra van szükség a szenzorok között, hiszen ahelyett, hogy minden szenzor egy (esetlegesen távol lévő) központi állomásnak továbbítaná az adatokat, csak a néhány hop távolságra lévő szenzorok között történik üzenetváltás. Azonban a globális információ hiánya miatt az elosztott számítási szerkezetű algoritmusoknak általában nagyobb a lokalizációs hibája, mint a centralizált számítási szerkezetűeknek.

Használ-e horgonypontokat?

Attól függően, hogy a szenzorok között van-e olyan, melynek helyzete előre meghatározott (pl. GPS vevővel van felszerelve), beszélhetünk horgonypont nélküli ([20],[21]), és horgonypontot használó ([19],[15]) algoritmusokról

is. Érdemes megjegyezni, hogy a szenzorok koordinátáinak meghatározásához legalább három horgonypontra mindenképp szükségünk van. Ha ennél kevesebb szenzor koordinátáit ismerjük, akkor a lokalizáció során csak a szenzorok relatív helyzetét tudjuk kiszámolni.

Statikus vagy mozgó referenciapontok?

Ha rendelkezésünkre áll megfelelő hardver, és a földrajzi adottságok is megfelelőek, akkor lehetőség van a szenzorok helyzetének mozgó eszközökkel való meghatározására ([35], [36], [37]). A mobil eszközök a megfigyelési területen belül mozognak, és bizonyos időközönként elküldik helyzetüket a közelükben lévő szenzoroknak. A szenzorok az így kapott adatok alapján határozzák meg pozíciójukat.

A lokalizációs algoritmusok kiválasztásánál sok szempontot figyelembe kell vennünk: a rendelkezésre álló horgonypontok száma, a szenzorok memóriája és számítási kapacitása, energiatakarékosság, elvárt lokalizációs pontosság, stb. Nem létezik egy minden szempontból legjobb algoritmus. Amiben az egyik algoritmus gyengébben teljesít, az egy másik algritmus erőssége, így körültekintően kell kiválasztanunk melyik algoritmus felel meg leginkább az igényeinknek.

A szakdolgolgozatomban csak a statikus helyzetű szenzorok és referenciapontok esetére szorítkoztam, és a horgonypontok használata szerint csoportosítottam a később részletesen bemutatásra kerülő algorimusokat. Ennek oka, hogy véleményem szerint ez a legszembetűnőbb jellemzője az algoritmusoknak. Ha egyik szenzor helyzetét sem ismerjük előre, akkor nem tudunk horgonypontot használó lokalizációs módszert alkalmazni, míg ha néhány szenzor koordinátái előre meghatározottak, akkor nem érdemes hogonypont

nélküli algoritmust használnunk, hiszen elvesztenénk annak előnyét, hogy a tőlük vett távolságok (a mérési hibáktól eltekintve) sokkal megbízhatóbb információt szolgáltatnak, mintha csak relatív helyzeteket ismernénk.

A következő fejezetekben bemutatott algoritmusok kiválasztásánál igyekeztem minél változatosabb algoritmusokat választani. Találhatunk köztük horgonypontot használó (CCE, AHLoS), és horgonypont nélküli algoritmusokat (MDS,MDS-MAP(P), AFL, AHLoS), centralizált (CCE) és elosztott számítási szerkezetűeket(MDS-MAP(P)) is. Az MDS-MAP(P) és AFL algoritmusok a range-based csoportba sorolhatóak, míg a CCE és AHLoS csak hop távolságokat használnak. A két csoporton belül az algoritmusok a publikálásuk sorrendjében szerepelnek.

4. Horgonypontot használó algoritmusok

A következő algoritmusok bemutatásánál feltesszük, hogy a szenzorok egy bizonyos százalékának előre ismerjük a koordinátáit. Ezek ismeretében feladatunk, hogy a többi, ismeretlen helyzetű szenzorok koordinátáit meghatározzuk.

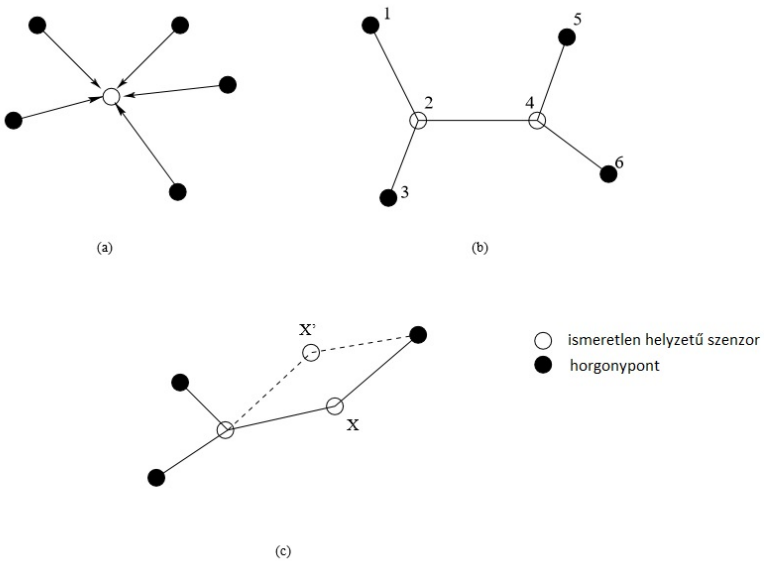
4.1. Az AHLoS algoritmus

Savvides és szerzőtársainak AHLoS (Ad-Hoc Localization System) algoritmusa [27] csak azt követeli meg, hogy a szenzorok egy kis része legyen ismert helyzetű, ha elég sűrű a hálózat. Ha egy szenzor három (nem kollineáris) szomszédjának helyzete már ismert, akkor egyértelműen meg tudja határozni a koordinátáit (5. a. ábra). Ezt használja ki az algoritmus, mely sorba veszi a gráf csúcsait, azzal kezdve, melynek legtöbb a horgonypont szomszédja. Ha egy szenzor meghatározta koordinátáit, akkor ezentúl ő is horgonypontként viselkedik. Egészen addig iteráljuk ezt a lépést, amíg van olyan csúcs, mely legalább három horgonyponttal szomszédos.

Elképzelhető, hogy egy csúcs helyzete akkor is meghatározható, ha szomszédai közül legfeljebb kettő elhelyezkedését ismerjük (5. b. ábra). Ehhez a kommunikációs sugárnál nagyobb távolságra lévő pontokat is figyelembe kell venni, melyek így együtt próbálják meghatározni helyzetüket. Azonban vigyázni kell, nem minden esetben lehet egyértelműen meghatározni a koordinátákat (1. c. ábra), csakis abban az esetben, ha a csúcsok közötti távolságokra felírható egyenletrendszernek egyértelmű megoldása van.

4.1.1. Az algoritmus

Válasszunk egy olyan v szenzort, melynek maximálisan sok horgonypont szomszédja van. Ekkor a v szenzorra és egy i horgony szomszédaira a következő egyenletrendszert szeretnénk megoldani:



1. ábra.

$$d_{iv} = \sqrt{(x_i - \hat{x}_v)^2 + (y_i - \hat{y}_v)^2}.$$

Átrendezve:

$$-x_i^2 - y_i^2 = (\hat{x}_v^2 + \hat{y}_v^2) + \hat{x}_v(-2x_i) + \hat{y}_v(-2y_i) - d_{iv}^2$$

Mivel a v szenzornak $k \geq 3$ horgonypont szomszédja van, ezért a k. egyenletet kivonva a többiből megszabadulhatunk az $(\hat{x}_v^2 + \hat{y}_v^2)$ tagtól.

$$-x_i^2 - y_i^2 + x_k^2 + y_k^2 = 2\hat{x}_v(x_k - x_i) + 2\hat{y}_v(y_k - y_i) + (d_{ik}^2 - d_{iv}^2)$$

Tehát az egyenletrendszer $y = b\mathbf{X}$ alakba írható, ahol

$$\mathbf{X} = \begin{bmatrix} 2(x_k - x_1) & 2(y_k - y_1) & d_{1k}^2 - d_{1v}^2 \\ 2(x_k - x_2) & 2(y_k - y_2) & d_{2k}^2 - d_{2v}^2 \\ \vdots & \vdots & \vdots \\ 2(x_k - x_{k-1}) & 2(y_k - y_{k-1}) & d_{(k-1)k}^2 - d_{(k-1)v}^2 \end{bmatrix}$$

$$y = \begin{bmatrix} -x_1^2 - y_1^2 + x_k^2 + y_k^2 \\ -x_2^2 - y_2^2 + x_k^2 + y_k^2 \\ \vdots \\ -x_{k-1}^2 - y_{k-1}^2 + x_k^2 + y_k^2 \end{bmatrix}$$

és

$$b = \begin{bmatrix} \hat{x}_v \\ \hat{y}_v \\ \vdots \\ 1 \end{bmatrix}.$$

Ennek megoldása $b = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top y$.

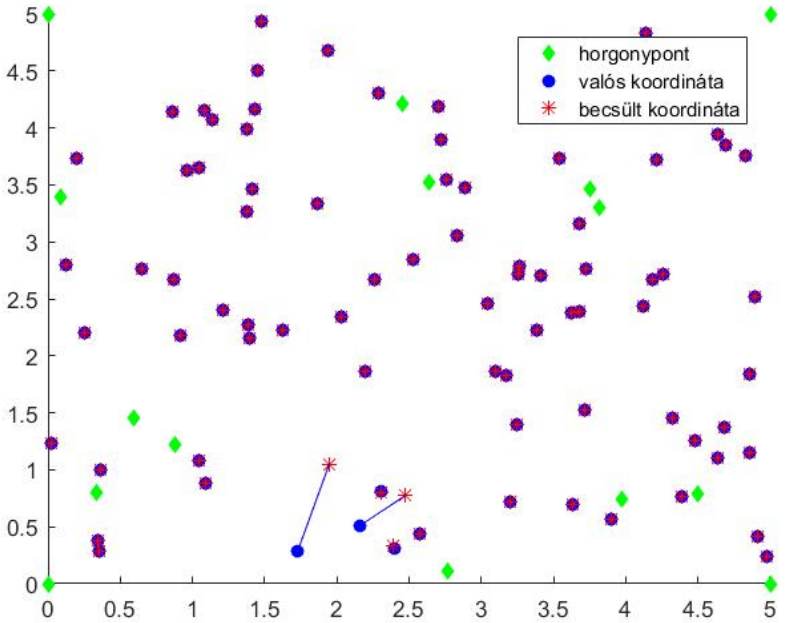
Ha egy szenzorra megoldottuk a fenti egyenletrendszert, akkor ezentúl őt is horgonypontnak tekintjük. A lépést addig iteráljuk, amíg van olyan csúcs, mely legalább három horgonyponttal szomszédos. Nézzük mit tehetünk, ha egy szenzornak háromnál kevesebb horgonypont szomszédja van.

Egy csúcsot közreműködőnek nevezünk ha horgonypont, vagy legalább három közreműködő szomszédja van.

Jelölje B a horgonypontok halmazát és legyen U az ismeretlen koordinátájú közreműködő pontok halmaza. Célunk

$$f(x_u; y_u) = d_{iu} - \sqrt{(\hat{x}_i - \hat{x}_u)^2 + (\hat{y}_i - \hat{u})^2}$$

minimalizálása $\forall u \in U$, ahol $i \in B$ vagy $i \in U$ és $u \in U$, valamint i és u szomszédosak és ha $i \in B$, akkor, x_i és y_i ismert. Az optimalizáláshoz én a Levenberg-Marquardt [31] algoritmust használtam. Az algoritmus eredménye 100 szenzor esetén a 2. ábrán látható.



2. ábra. Az AHLoS algoritmus eredménye 100 egyenletesen elhelyezett szenzor esetén. A horgonypontok aránya 15%. Látható, hogy az algoritmus kis eltéréssel határozza meg a szenzorok tényleges koordinátáit.

4.2. Kereszt-entrópia módszerén alapuló algoritmus

Azim és társai által kidolgozott algoritmus [23] centralizált számítási szerkezetű, és horgonypontokat használ a lokalizációhoz. Az algoritmus a kereszt-entrópia módszerén [24] alapszik, amely egy globális optimalizáló algoritmus. Nevét a Kullback-Leibler-féle kereszt entrópia [25] után kapta, amely két valószínűségi mérték közti távolság mérésére szolgál. A módszert kezdetben ritka események valószínűségének becslésére használták komplex sztochasztikus hálózatokban, de hamar rájöttek, hogy jól alkalmazható kombinatorikus optimalizálási problémák megoldásához is.

A kereszt-entrópia módszere a kombinatorikus optimalizálás, a Monte-Carlo módszer és a gépi tanulás elemeit ötvözi. Egy iteratív eljárás, melynek során először véletlen mintákat generálunk, majd a paramétereket a legjobb egyedek alapján úgy frissítjük, hogy a következő iterációs lépés során „jobb”

egyedeket kapjunk. A módszer előnye abban rejlik, hogy a frissítési szabályok általában egyszerűek (és így gyorsan megvalósíthatóak).

A lentebb bemutatásra kerülő algoritmusnál [23] kezdetben véletlen koordinátákat, várható értékeket és szórásokat rendelünk a szenzorokhoz. A várható értékek és szórások segítségével mintákat veszünk, majd a mintákat értékeljük egy korábban bevezetett költségfüggvény alapján. Ha találunk olyan mintát, amelynek lokalizációs hibája kisebb, mint a jelenlegi becsült koordinátához tartozó, akkor frissítjük a szenzor koordinátáit, a becsült koordináta a legjobb minta lesz. Az új várható értéket és szórást a legjobb M minta alapján számoljuk. A lépést addig iteráljuk, amíg az ismeretlen helyzetű szenzorokoz rendelt szórás elég kicsi nem lesz. Azonban ennél a módszernél a szenzorok helyzetének jó becsléséhez szükséges, hogy a szenzorok nagy része horgonypont legyen.

4.2.1. Az algoritmus

A megfigyelt területen N darab szenzor található, melyek közül A darab horgonypont. Az inicializálás során minden szenzor egy listát készít a szomszédairól, és tőlük vett távolságokról. Ezt a listát elküldi egy központi számítógépnek, amely a kereszt-entrópia módszerét alkalmazva meghatározza a szenzorok helyzetét.

Az algoritmus futása során célunk, hogy minimalizáljuk a lokalizációs hibát. A lokalizációs hiba:

$$cost = \sum_{i=A+1}^{N} \sum_{j \in n_i} (\hat{d}_{ij} - d_{ij})^2,$$

ahol n_i az i. szenzor szomszédainak halmaza.

Kezdetben a lokalizációs algoritmus minden ismeretlen helyzetű i szenzorhoz véletlenszerűen generál egy $(\hat{x}_i, \hat{y}_i)$ párt, valamint minden szenzorhoz hozzárendel egy véletlen $\mu_i = (x\mu_i, y\mu_i)$ várható értéket, és $\sigma_i = (x\sigma_i, y\sigma_i)$ szórást. Ezekhez a kezdeti koordinátákhoz tartozó lokalizációs hibát a $BestCost$

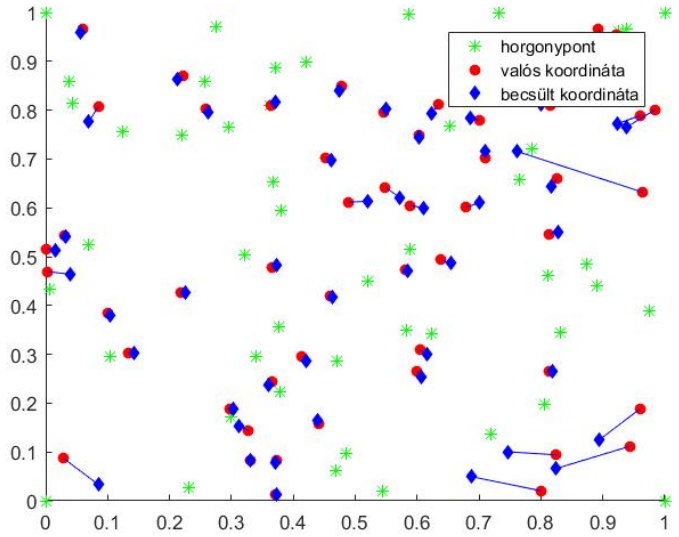
változóban tároljuk.

Az iteratív lépés során μ várható érték és σ szórás segítségével előállítunk S számú mintát mind az $\hat{x}_i$ mind a $\hat{y}_i$ koordinátához ($i \in \{1, 2, \ldots, N - A\}$). Ezek közül kiválasztjuk a legjobbat, azaz azt az x_i^* és y_i^* koordinátát, melyre a lokalizációs hiba a legkisebb. Ha ez a *bestcost* érték kisebb, mint az eddigi *BestCost*, akkor a továbbiakban a szenzor feltételezett helyzete a legjobb minta lesz, azaz x_i és y_i koordinátákat x_i^*-ra, illetve y_i^*-ra cseréljük. A lokalizációs hibát is módosítjuk, *bestcost* lesz az új értéke.

A szenzorhoz rendelt várható érték és szórás értékeit is frissítjük, ehhez a legjobb M mintát használjuk. Legyen $x\mu best_i$, illetve $x\sigma best_i$ a legjobb M minta első koordinátájának átlaga, illetve szórása. Ekkor az új $x\mu_i$ várható érték az $x\mu_i$ és $x\mu best_i$, $x\sigma_i$ az $x\sigma_i$ és $x\sigma best_i$ egy konvex kombinációja lesz. A konvex kombináció együtthatóit a várható érték esetén az α, a szórás esetén a β tanulási ráta határozza meg. Analóg módon számolható ki $y\mu_i$ és $y\sigma_i$. Ezek az új értékek segítenek egyre jobb, kisebb hibájú generációk létrehozásában. Ha egy lépés során a legjobb minta hibája nagyobb, mint *BestCost*, akkor a mintákat figyelmen kívül hagyjuk. A következő lépésben a minták egy újabb halmazát generáljuk.

Ezt a lépést addig iteráljuk, amíg a szórások maximuma egy kívánt γ érték alá nem csökken. Ekkor megkaptuk a szenzoroknak az algoritmus által becsült koordinátáit.

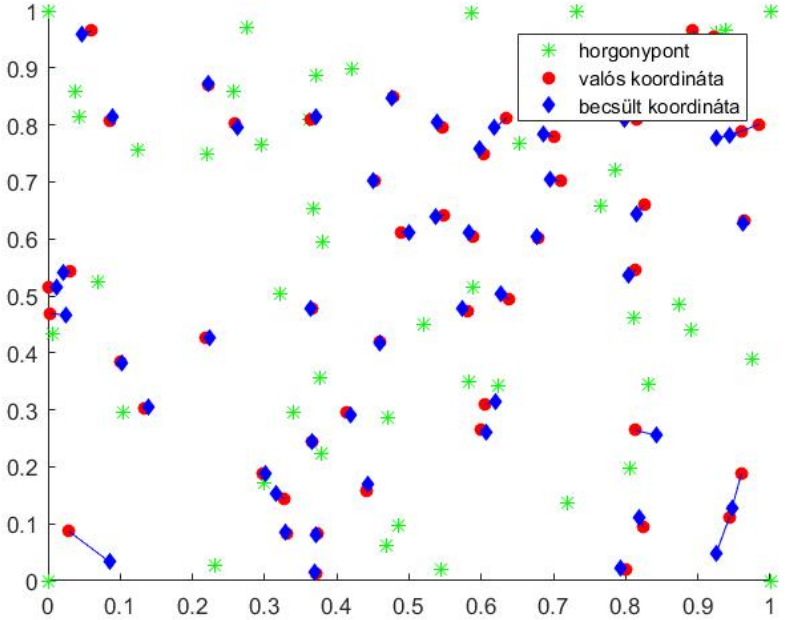
Az algoritmus hátránya, hogy a megfelelő (kellően kis hibájú) működéséhez szükséges, hogy a szenzorok nagy arányban horgonypontok legyenek. Sajnos a gyakorlatban nem mindig áll fenn ez a helyzet. Emellett ha egy szenzorral szomszédos horgonypontok (közel) kollineárisak, akkor az algoritmus által meghatározott helyzete elég távol lehet a valódi helyzetétől.



3. ábra. Az algoritmus eredménye 100 egyenletesen elhelyezett szenzor esetén, melyek fele horgonypont. Látható, hogy az algoritmus nagyon kis eltéréssel meghatározza a szenzorok tényleges koordinátáit, kivéve azon szenzorok esetén, melyeknek horgonypont szomszédaik kevesen vannak vagy (majdnem) kollineárisak.

4.2.2. Hogyan módosíthatunk az algoritmuson?

1. Az algoritmus során a szenzorok feltételezett helyzete folyamatosan változik. Egy szenzor hibája nem csak a saját, hanem a szomszédainak becsült koordinátáitól is függ. Ha a fenti hibafüggvény kiszámítása során ha nagyobb (pl. kétszeres) súllyal számítjuk a horgonypontoktól vett távolságot, akkor ezzel nagyobb hangsúlyt adunk azoknak a távolságoknak, melyek csak a szenzortól függnek. Ezzel csökken a lokalizációs hiba, hiszen mivel a horgonypontok helyzete nem változik, így az ezektől vett távolságokban jobban bízhatunk, mint a minden iteráció során helyzetet változtató szenzorok egymástól vett távolságában.



4. ábra. A módosított algoritmus eredménye 100 egyenletesen elhelyezett szenzor esetén, melyek fele horgonypont.

Az algoritmus hibájának mérésére sokféle metrika létezik, én a valós és becsült pozíciók különbségének átlagát tekintettem, azaz a lokalizációs hiba:

$$error = \frac{\sum_{i=A+1}^{N} \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}}{N - A}.$$

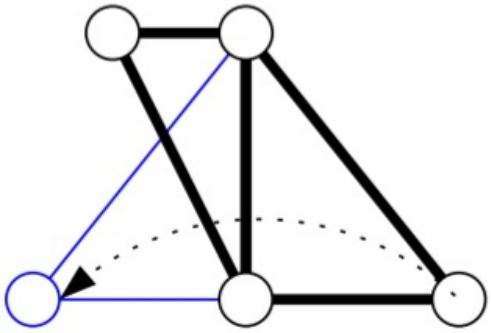
Ez a függvény természetesen ebben az esetben csak azért használható, mert az algoritmus futtatásakor én határoztam meg (egyenletes eloszlással, véletlenszerűen) a szenzorok helyzetét. A gyakorlatban azonban, mivel nem ismerjük a szenzorok pontos koordinátáit (kivéve a horgonypontokét), ezért az algoritmus hibájának besclésére a fenti *cost* függvényt használhatjuk.

Az algoritmus átlagos lokalizációs hibája az 1. táblázatban látható, melyet 100 szenzor 10 különböző (egyenletes eloszlással való) elhelyezéséből kapott eredményekből számoltam, $\gamma = 0.01$ paraméter mellett.

Hibafüggvény	Eredeti algoritmus	Módosított algoritmus
error	0.032	0.023
cost	0.183	0.139

1. táblázat. lokalizációs hibák

Mint említettem az algoritmus egyik hátulütője, hogy ha egy szenzor szomszédai (majdnem) kollineárisak, akkor előfordulhat, hogy a szenzornak az algoritmus által meghatározott helyzete a valódi helyzetének a szomszédai által alkotott egyenesre, mint tengelyre vett tükörképe (lásd 5. ábra).



5. ábra.

Azim és társai 2015-ben megjelent cikkükben [26] ennek a problémának a megoldását célozták meg. Ehhez a *cost* költségfüggvényt módosították. Bevezettek egy büntetőfüggvényt, amely egyre növekvő súllyal bünteti azokat a csúcsokat, melyeknek az aktuális pozíciója olyan, hogy léteznek olyan szomszédai, melyek nem szerepelnek a csúcs szomszédsági listáján, vagy épp ellenkezőleg, vannak olyan szenzorok, melyek ugyan rajta vannak a szomszédsági listáján, de az aktuális becsült helyzetük olyan, hogy a szenzor kommunikációs tartományán kívül esnek.

Büntetőfüggvényként a logaritmikus barrier-függvényt választották. Tehát

$$\psi_i = \sum_{j \in n_i} (\hat{d}_{ij} - d_{ij})^2$$

$$\hat{d}_{ip} \leq R \qquad \forall p \in n_i$$

$$\hat{d}_{iq} > R \qquad \forall q \in \bar{n}_i$$

megkötések helyett logaritmikus barrier-függvényt használtak:

$$\psi_i = \sum_{j \in n_i} (\hat{d}_{ij} - d_{ij})^2 + r_k \left(- \sum_{j \in n_i} ln(-(\hat{d}_{ij} - R)^2 \right) +$$

$$+ r_k \left(- \sum_{j \in \bar{n}_i} ln(-(\hat{d}_{ij} - R)^2 \right),$$

ahol R a szenzorok kommunikációs sugara, n_i, illetve $\bar{n}_i$ az i csúccsal szomszédos, illetve nemszomszédos csúcsok halmazát jelöli, r_k pedig az iterációs lépések számától függő monoton növő függvény.

Szimulációjukból kiderül, hogy a módosított költségfüggvénnyel az algoritmus sokkal jobban teljesít a lokalizációs hiba tekintetében.

5. Horgonypont nélküli algoritmusok

A következőekben bemutatásra kerülő algoritmusok csak a szenzorok relatív helyzetét határozzák meg, nem szükséges ismernünk egyik szenzor pontos koordinátáit sem. A gyakorlatban azonban csak ritkán elegendő a relatív helyzetüket ismernünk, ugyanis tudnunk kell pontosan honnan érkeznek a szenzorok által mért adatok. Ahhoz, hogy a relatív helyzetük alapján a tényleges elhelyezkedésüket is meg tudjuk határozni, legalább három szenzor koordinátáit ismernünk kell. A 6. fejezetben bemutatom hogyan lehet meghatározni azt a transzformációt, amely az algoritmusok által eredményként kapott pozíciókat úgy forgatja, nyújtja és tükrözi, hogy azok a relatív helyzetüket megtartják, és a horgonypontoknak a transzformációval kapott koordinátái megegyeznek a valódi koordinátáikkal.

5.1. MDS

A többdimenziós skálázás (multidimensional scaling, továbbiakban MDS) [22] módszerével egy adathalmazt ábrázolhatunk a $q-$dimenziós térben, mégpedig úgy, hogy a pontok közötti euklideszi távolságok az adatpontok hasonlóságát tükrözik. Esetünkben a szenzorok közötti távolságokat ismerjük, és úgy szeretnénk beágyazni őket a síkba, hogy a közöttük lévő távolságok minél jobban megközelítsék valós távolságukat, azaz azokat az $x_1, \ldots, x_n \in \mathbb{R}^2$ vektorokat keressük, melyre $\|x_i - x_j\| \approx d_{ij}$, ahol d_{ij} jelöli az i. és j. szenzor távolságát.

5.1.1. Az algoritmus

Vegyük észre, hogy nem csak egy megoldás létezik, hiszen ha $\mathbf{X} = [x_1, \ldots, x_n]$ megoldás, akkor $\mathbf{X}^* = \mathbf{X} + c, \quad c \in \mathbb{R}^q$ szintén teljesíti az

$$\|x_i^* - x_j^*\| = \|(x_i + c) - (x_j + c)\| = \|x_i - x_j\| = d_{ij}$$

egyenletrendszert. Ezért válasszuk úgy az x_i vektorokat, hogy teljesüljön

$$\sum_{i=1}^{n} x_{ik} = 0 \qquad \forall k = 1, \ldots, n\text{-re.} \tag{1}$$

Legyen $\mathbf{B} = \mathbf{X}^\top \mathbf{X}$. Ekkor

$$d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij}, \tag{2}$$

hiszen

$$\|x_i - x_j\|^2 = x_i^\top x_i + x_j^\top x_j - 2x_i^\top x_j.$$

Ekkor az (1) miatt:

$$\sum_{i=1}^{n} b_{ij} = \sum_{i=1}^{n} \sum_{k=1}^{q} x_{ik} x_{jk} = \sum_{k=1}^{q} x_{jk} \sum_{i=1}^{n} x_{ik} = 0 \qquad j = 1, \ldots, n.$$

Felírhatóak a következő összefüggések:

$$\sum_{i=1}^{n} d_{ij}^2 = tr(\mathbf{B}) + nb_{jj}$$

$$\sum_{j=1}^{n} d_{ij}^2 = tr(\mathbf{B}) + nb_{ii}$$

$$\sum_{j=1}^{n} \sum_{i=1}^{n} d_{ij}^2 = \sum_{j=1}^{n} (tr(\mathbf{B}) + nb_{ii}) = n \cdot tr(\mathbf{B}) + n \cdot tr(\mathbf{B}) = 2n \cdot tr(\mathbf{B}). \tag{3}$$

A (2)-ből és a (3)-ból következik:

$$b_{ij} = -\frac{1}{2}\left(d_{ij}^2 - -b_{ii} - b_{jj}\right), \text{ azaz}$$

$$\mathbf{B} = -\frac{1}{2}\mathbf{J} D^{(2)} \mathbf{J},$$

ahol $\mathbf{J} = \mathbf{I} - \frac{1}{n} 1 1^\top$, $\mathbf{I}$ jelöli az $n \times n$-es egységmátrixot és $\mathbf{D}^{(2)} = [d_{ij}^2]$.

Innen $\mathbf{X}$ megoldásmátrix $\mathbf{B}$ spektrálfelbontásából következik. Legyen $\mathbf{E\Sigma E}$ a $\mathbf{B}$ mátrix spektrálfelbontása, ahol $\mathbf{E}$ a $\mathbf{B}$ sajátvektoraiból álló mátrix, $\mathbf{\Sigma}$ pedig a $\mathbf{B}$ sajátértékei által alkotott diagonális mátrix. Ekkor

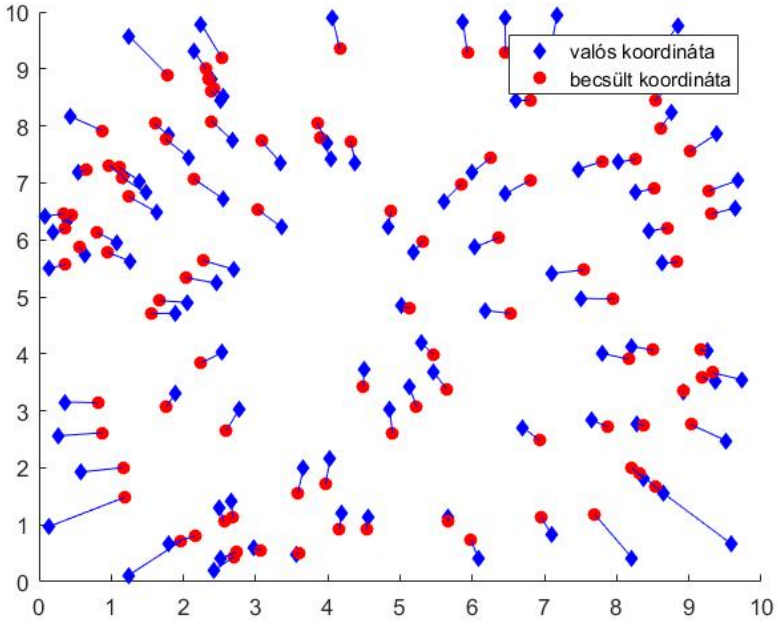
$$\mathbf{X} = \mathbf{\Sigma}^{\frac{1}{2}} \mathbf{E}^\top$$

.

Azonban mi $\mathbb{R}^2$-beli megoldást keresünk. Ennek megtalálásához elegendő $\mathbf{X}$ első két sorát vennünk, hiszen a lineáris dimenziócsökkentő eljárások közül ez adja a távolságokat leginkább megörző megoldást. Tehát a keresett mátrixunk

$$\mathbf{X}_2 = \mathbf{\Sigma}_2^{\frac{1}{2}} \mathbf{E}_2^\top,$$

ahol $\mathbf{\Sigma}_2$ a $\mathbf{\Sigma}$ bal felső 2×2-es almátrixa, és $\mathbf{E}_2$ az $\mathbf{E}$ mátrix első két oszlopa által alkotott mátrix [28].



6. ábra. Az MDS algoritmus eredménye ismert páronkénti euklideszi távolságok esetén.

5.2. Az AFL algoritmus

Priyantha és társainak AFL (Anchor-Free Localization) algoritmusa [32] során a szenzorok a kezdetben kijelölt koordinátáik és kizárólag a szomszédságukban lévő szenzoroktól mért távolságaik alapján külön-külön határozzák meg helyzetüket.

Az algoritmus két részből áll. Az első részben beágyazzuk a gráfot a síkba, majd a második részben finomítunk az így kapott koordinátákon. Ez utóbbi lépés során a gráfban a csúcsokat összekötő éleket rugóknak képzeljük, melyek nyugalmi helyzetben olyan hosszúak, mint a végpontjaiknak megfelelő szenzorok távolsága. Ha a szenzorok pillanatnyi becsült helyzete kisebb, mint a valós euklideszi távolságuk, akkor a rugó igyekszik távolabb tolni őket egymástól, ha azonban a valóságban közelebb vannak egymáshoz, mint becsült koordinátáik, akkor a rugó megpróbálja őket közelebb húzni egymáshoz. A rugók által kifejtett erő nagyságának és irányának megfelelően fogjuk eltolni a gráf csúcsait egy-egy vektorral.

5.2.1. Az algoritmus

Priyantha és társai A GER (Global Energy Ratio) metrikát vezették be az algoritmus hibájának mérésére.

$$\mathrm{GER} = \frac{\sqrt{\sum_{i,j,i<j} \hat{e}_{ij}^2}}{n(n-1)/2}, \text{ahol} \qquad \hat{e}_{ij} = \frac{\hat{d}_{ij} - d_{ij}}{d_{ij}}.$$

Ez a metrika nem csak a szenzorok valódi és becsült helyzetének távolságát, hanem a gráf topológiáját is jól írja le, hiszen nem csak a szomszédos csúcsok távolságát veszi figyelembe, hanem az egymástól távol lévőket is.

Ha a csúcsoknak egy tetszőleges síkbeli beágyazásából indulunk ki, akkor a rugó módszer gyakran a fenti hibafüggvényre nézve lokális minimumot talál. A cikk írói megfigyelték, hogy ez általában akkor fordul elő, ha a csúcsok egy csoportja egy vagy több élre tükrözve van (lásd 5. ábra), azaz a gráf nem hajtásmentes (fold-free). Ezért fontos, hogy olyan kezdeti beágyazást keressünk, mely hasonlít a szenzorok valódi elhelyezkedéséhez, azaz az ál-

talunk meghatározott, síkba beágyazott gráf minden körének csúcssorrendje megegyezzen a szenzorok által alkotott gráf megfelelő körének csúcssorrendjével. A következő heurisztikus eljárás ugyan nem garantálja, hogy a fenti tulajdonság teljesül, de ez a motivációs elv mögötte.

Feltesszük, hogy minden szenzor rendelkezik egy egyedi azonosítóval. Először választunk öt referenciacsúcsot. Ezek közül n_1-t, n_2-t, n_3-t és n_4-et úgy választjuk, hogy azok a gráf szélén helyezkedjenek el, és az (n_1, n_2) pár nagyjából merőleges legyen (n_3, n_4)-re. Az n_5 csúcs a gráf közepén fog elhelyezkedni.

Választunk egy tetszőleges n_0 csúcsot (például amelyik azonosítója a legkisebb). Ezután úgy választjuk n_1-et, hogy h_{01} maximális legyen. Az n_2 csúcsot úgy választjuk, hogy az az n_1-től a lehető legnagyobb hop távolságra legyen. n_3 választásakor a $|h_{13} - h_{23}|$-t minimalizáló csúcsok közül azt választjuk, melyre $h_{13} + h_{23}$ maximális. Így n_3 az n_1 és n_2 közel azonos távolságra, és tőlük távol fog elhelyezkedni. Hasonlóan n_4-et úgy választjuk, hogy $|h_{14} - h_{24}|$ minimális legyen. Ennek a feltételnek eleget tevő csúcsok közül pedig azt választjuk, amelyre h_{34} maximális. Így egy olyan csúcsot kapunk, amely n_1-től és n_2-től közel azonos távolságra, és az n_3 csúcstól a lehető legtávolabb van. Végül n_5-öt a $|h_{15} - h_{25}|$-öt minimalizáló csúcsok közül úgy választjuk, hogy $|h_{35} - h_{45}|$ minimális legyen. Így n_5 körülbelül a terület közepén fog elhelyezkedni.

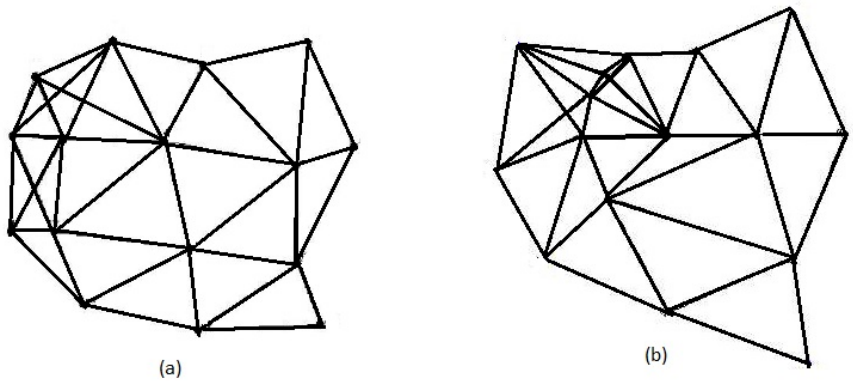
A többi n_k csúcs (ρ_k, θ_k) polárkoordinátáit a következő képlettel számolhatjuk ki:

$$\rho_k = h_{5k} \times r$$
$$\theta_k = tan^{-1}\left(\frac{h_{1k} - h_{2k}}{h_{3k} - h_{4k}}\right),$$

ahol r jelöli a szenzorok maximális kommunikációs sugarát. Ha $h_{1k} = h_{2k}$ és $h_{3k} = h_{4k}$, akkor a k csúcs a terület közepén helyezkedik el, ezért kezdeti koordinátái megegyeznek n_5 kezdeti koordinátáival.

Az így kapott koordináták elég jól közelítik a szenzorok valós elhelyezkedését, főleg azoknak a szenzoroknak az esetében, melyek a terület szélén helyezkednek el. Mivel r-et a maximális kommunikációs sugárnak választottuk,

ezért olyan gráfot kaptunk, mely nagyobb területet fed le, mint az eredeti. Ez a tulajdonság segít elkerülni, hogy lokális minimumot találjunk.



7. ábra. Az (a) ábrán látható 16 csúcsú gráf a szenzorok valódi helyzetét mutatja. A (b) ábra mutatja a csúcsok kezdeti beágyazását a síkba.

A fenti módon kiszámolt Euklideszi koordinátákon finomítunk a második részben. Az alább ismertetett rugós optimalizációs lépést minden szenzor párhuzamosan, egy időben végzi el. Egy szenzor meghatározott időközönként elküldi becsült helyzetét szomszédainak, hogy azok saját és szomszédaik becsült koordinátáinak ismeretében pontosabb becslést tudjanak végezni.

Jelölje az i csúcs valódi koordinátáit x_i és y_i, becsült koordinátáit $\hat{x}_i$ és $\hat{y}_i$, i és j valódi és becsült helyzetének távolságát d_{ij}, illetve $\hat{d}_{ij}$.

Legyen $\vec{v}_{ij}$ az $\hat{x}_i$-ből $\hat{x}_j$-be mutató egységvektor. Ekkor a $\vec{v}_{ij}$ irányú erő

$$\vec{F}_{ij} = \vec{v}_{ij}(\hat{d}_{ij} - d_{ij}),$$

az i csúcsra ható eredő erő pedig

$$\vec{F}_i = \sum_j \vec{F}_{ij}.$$

Az i csúcs energiája:

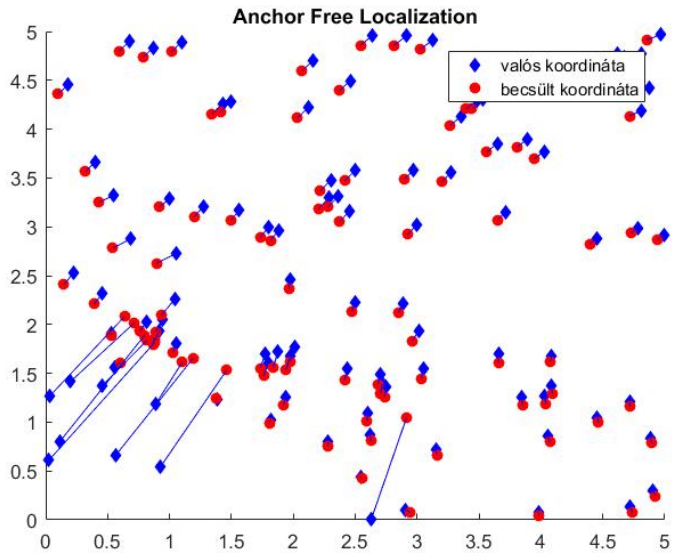
$$E_i = \sum_j E_{ij} = \sum_j (\hat{d}_{ij} - d_{ij})^2.$$

A rendszer energiája:

$$E = \sum_i E_i.$$

Az n_i csúcsok energiája csökken, ha $\vec{F_i}$ erő irányába egy kicsit elmozdítjuk őket. Az, hogy pontosan mennyivel, két szempontból is fontos. Egyrészt azt szeretnénk, hogy a csúcsok energiája az új pozíciójukban kevesebb legyen, másrészt szeretnénk elkerülni, hogy a GER metrikára nézve lokális minimumot találjunk. Az első feltétel teljesülését azzal garantálhatjuk, hogy előre kiszámítjuk mennyi lenne az adott csúcs energiája, ha elmozdítanánk, de csak akkor helyezzük át az új pozícióba, ha energiája így csökken. Az eltolásvektor nagyságát Priyantha és társai szimulációk futtatása után $\frac{|\vec{F_i}|}{2m_i}$-nek választották, ahol m_i jelöli az i csúcs szomszédainak számát.

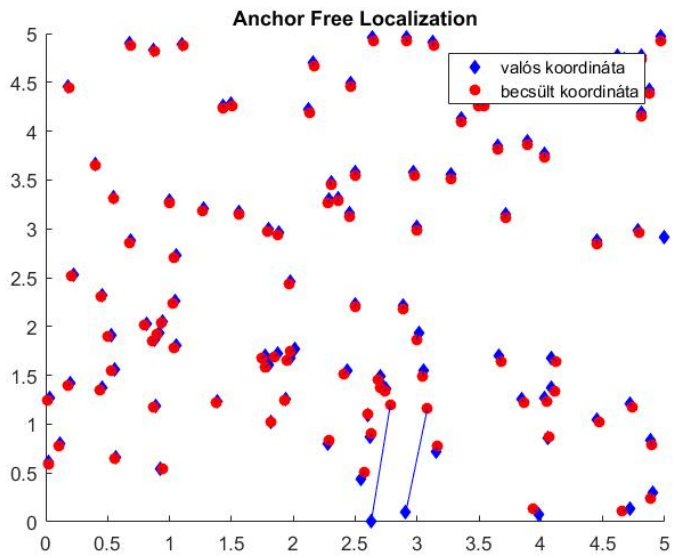
Addig folytatjuk az eljárást, míg a lokalizáció hibája kellően kicsi nem lesz. Sajnos ehhez elég sok iterációra van szükség.



8. ábra. Az AFL algoritmus eredménye 100 egyenletesen elhelyezett szenzor esetén 1000 iterációs lépés után.

5.2.2. Hogyan módosíthatunk az algoritmuson?

1. Természetesen ha ismerjük a szenzorok egymástól való távolságát, akkor nem szükséges a hop távolsággal számolnunk, alkalmazhatjuk helyette a valós távolságadatokat. Ez nem meglepő módon pontosabb eredményhez vezet.



9. ábra. Az módosított AFL algoritmus eredménye 100 egyenletesen elhelyezett szenzor esetén 1000 iterációs lépés után.

Az algoritmus hibájának méréséhez én ebben az esetben is a valós és becsült pozíciók különbségének átlagát tekintettem, azaz a lokalizációs hiba:

$$\frac{\sum_{i=1}^{n} \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}}{n}.$$

Az algoritmus átlagos lokalizációs hibája a 2. táblázatban látható, melyet 100 szenzor 10 különböző (egyenletes eloszlással való) elhelyezéséből kapott eredményekből számoltam.

Eredeti algoritmus	Módosított algoritmus
0.191	0.058

2. táblázat. lokalizációs hiba: valós és becsült pozíciók különbségének átlaga

5.3. MDS-MAP(P)

Egy másik, az MDS módszeren alapuló algoritmus az MDS-MAP [29], mely a szenzorok közötti kommunikációs kapcsolat meglétét használja a lokalizációhoz.

Első lépésként kiszámoljuk a legrövidebb utat minden szenzorpár között (mivel csak a kommunikációs kapcsolat meglétét használjuk, ezért ez egyenlő a szenzorok hop-távolságával). Ezek segítségével konstruáljuk az MDS algoritmushoz szükséges **D** távolságmátrixot. Az MDS módszer segítségével megkapjuk a szenzorok leképezését a síkba, úgy, hogy az így kapott pontok közötti távolságok megközelítőleg megegyeznek a szenzorok közötti távolságokkal. Végső lépésként a kapott pontokat egy megfelelő lineáris transzformáció segítségével úgy forgatjuk, és toljuk el (és esetleg tükrözzük), hogy a horgonypontok a valós helyzetükbe kerüljenek.

A fenti módszer a pontpárok közötti legrövidebb utat használja, ami nem minden esetben ad jó közelítést a tényleges távolságokra. Ha a szenzorhálózat nem elég sűrű vagy a szenzorok nem elég egyenletesen vannak elhelyezve, akkor a szenzorok hop-távolsága jelentősen eltérhet valós (euklideszi) távolságuktól.

Ennek a problémának a kiküszöbölésére alkotta meg Yi Shang és Wheeler Ruml az MDS-MAP(P) algoritmust [30], mely az MDS-MAP módszer egy továbbfejlesztett változata. Algoritmusuk lényege, hogy a vizsgált területet több kisebb átfedő területre bontják, ezeken végzik el az MDS-MAP módszert, majd az így kapott „térképeket” illesztik össze.

Ezzel a módszerrel kiküszöbölhető a távoli pontok távolságának az őket összekötő legrövidebb út hosszával való becslése, a kisebb területeken pedig a hop-távolság elég jól becsli a tényleges távolságot. Ezenkívül a módszer előnye, hogy nem követel centralizált számítást, így nagyobb szenzorhálózatokban is jól alkalmazható.

5.3.1. Az algoritmus

Elöször minden szenzor egy lokális térképet (egy részgráfot és annak egy beágyazását a síkba) hoz létre a közelében lévő szenzorok segítségével. Ehhez először ki kell számolnunk az adott v szenzortól legfeljebb R_{lm} hop távolságra lévő szenzorok közötti legrövidebb utak hosszát, ahhoz, hogy megkapjuk az MDS algoritmushoz szükséges $\mathbf{D}_v$ távolságmátrixot.

Az MDS módszer által kapott lokális térképen tovább finomíthatunk. Jelölje h_{ij} az i és j szenzor hop távolságát. A v-től legfeljebb R_{ref} hop távolságra lévő szenzorok segítségével oldjuk meg az alábbi optimalizálási feladatot:

$$\min_{x_k,y_k} \sum_{i,j,i\neq j} w_{ij}(d_{ij} - h_{ij})^2 \qquad k = 1, \ldots, n,$$

ahol w_{ij} értékeket úgy választjuk, hogy az egymáshoz közel lévő szenzorok súlya nagyobb legyen, mint az egymástól távolabb lévőké, hiszen minél kevesebb két szenzor hop távolsága, az várhatóan annál jobban becsli a szenzorok euklideszi távolságát.

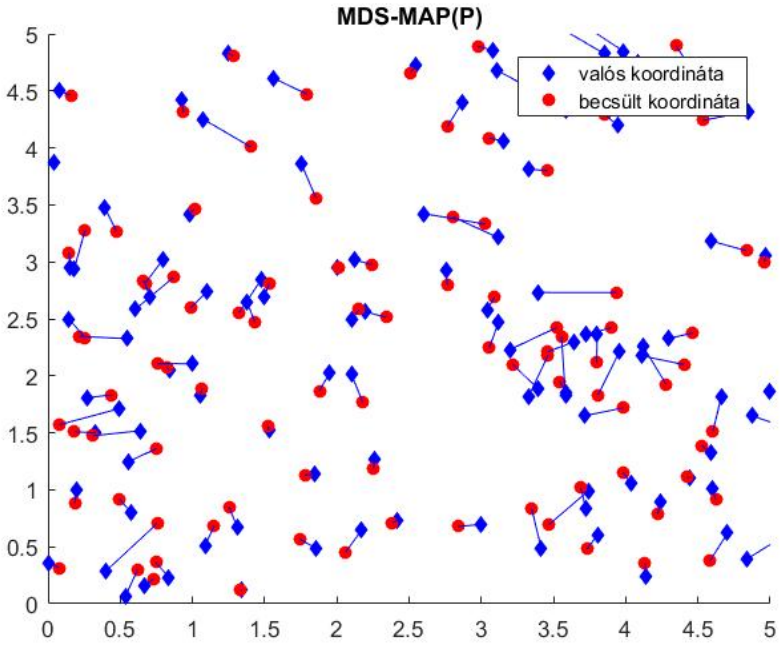
Shang és Ruml szimulációikban $R_{lm} = 2$, R_{ref}=2 paraméterek mellett $w_{ij} = 1$ súlyt adtak az 1 hop távolságra lévő szenzoroknak és $w_{ij} = \frac{1}{4}$ súlyt a 2 hop távolságra lévőknek. Az optimalizálási feladat megoldásához a Levenberg-Marquardt módszert [31] alkalmazták 10 iterációs lépéssel.

A fenti finomítási lépés opcionális, alkalmazása ugyan pontosabb eredményt szolgáltat, azonban kb. két nagyságrenddel lassabb, mint az MDS módszer.

A kapott részgráfokat össze kell illesztenünk, egyetlen nagy gráfot hozva létre, melyben minden szenzor megfelel egy-egy csúcsnak. Tetszőlegesen választunk egy szenzort. Ennek $G_0 = (V_0, E_0)$ részgráfjához fogjuk illeszteni a többit, mégpedig úgy, hogy mindig azt a G_1 részgráfot választjuk, melynek legtöbb közös pontja van vele. Ehhez a $G_1 = (V_1, E_1)$ geometriai gráfot úgy transzformáljuk, hogy a $V_0 \cap V_1$-beli pontok G_0 és G_1-beli példányainak távolságösszege minimális legyen (a következő fejezetben bemutatom ez hogyan

lehetséges), majd egy $v \in V_0 \cap V_1$ csúcshoz az így kapott koordináták átlagát rendeljük. Ha $v \in V_0 \setminus V_1$ vagy $v \in V_1 \setminus V_0$, akkor a G_0-hoz, illetve G_1-hez tartozó koordinátáit rendeljük. Az így összeillesztett $V_0 \cup V_1$ csúcshalmazú gráfra iteráljuk a fenti lépést.

Az így kapott gráfra szintén elvégezhetjük a lokális térképekre alkalmazott finomítási lépést.



10. ábra. Az MDS-MAP(P) algoritmus eredménye 100 egyenletesen elhelyezett szenzor esetén.

5.3.2. Hogyan módosíthatunk az algoritmuson?

1. Természetesen ha ismerjük a szenzorok egymástól való távolságát, akkor nem szükséges a hop távolsággal számolnunk, alkalmazhatjuk helyette a valós távolságadatokat. Ez nem meglepő módon pontosabb eredményhez vezet.

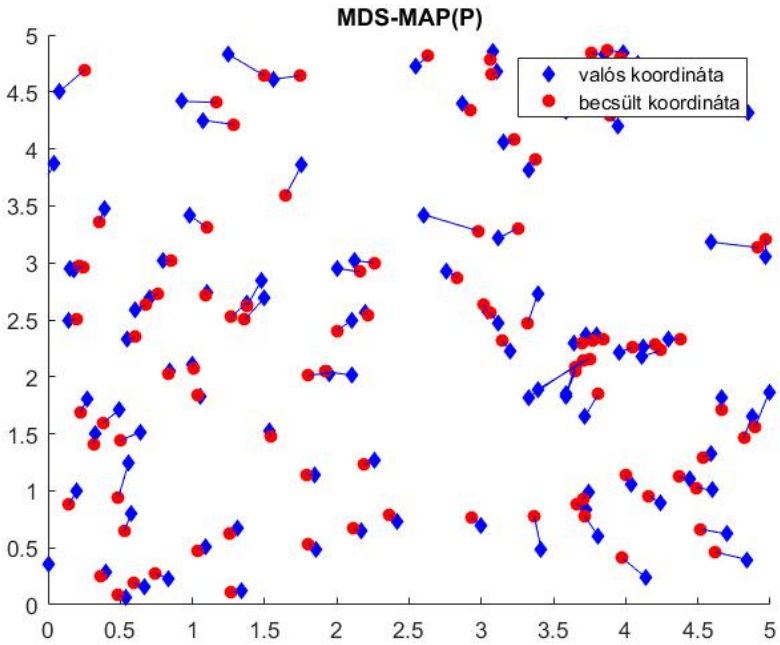
Eredeti algoritmus	Módosított algoritmus
0.239	0.162

3. táblázat. lokalizációs hiba: valós és becsült pozíciók különbségének átlaga

Az algoritmus hibájának mérésére sokféle metrika létezik, én a valós és becsült pozíciók különbségének átlagát tekintettem, azaz a lokalizációs hiba:

$$error = \frac{\sum_{i=1}^{n} \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}}{n}.$$

Az algoritmus átlagos lokalizációs hibája a 3. táblázatban látható, melyet 100 szenzor 10 különböző (egyenletes eloszlással való) elhelyezéséből kapott eredményekből számoltam.



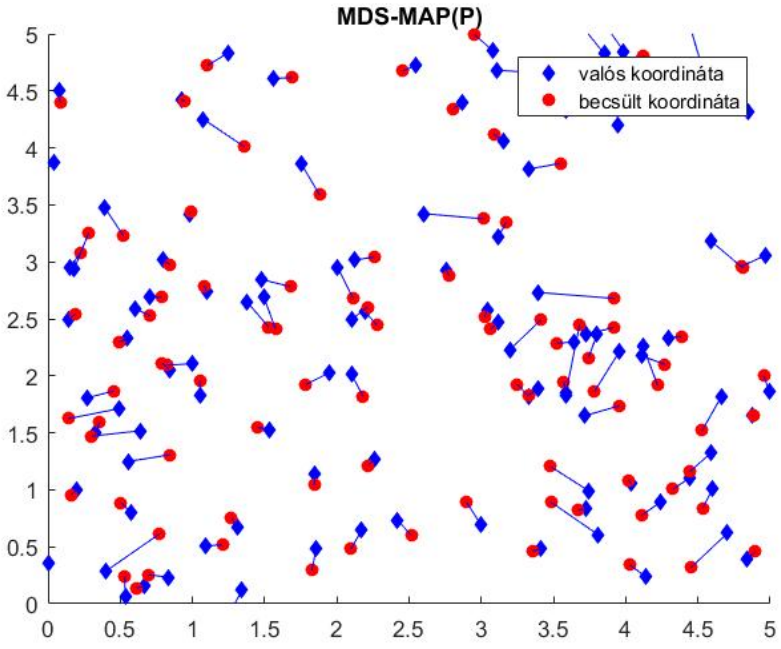
11. ábra. Az 1. módosított MDS-MAP(P) algoritmus eredménye 100 egyenletesen elhelyezett szenzor esetén.

2. Ha sűrű a hálózatunk, akkor a G_v részgráfok nagyban fedik egymást, egy szenzor sok lokális térképen szerepel. Ekkor megtehetjük, hogy nem

dsas	Eredeti algoritmus	Módosított algoritmus
hiba	0.19	0.21
futásidő (s)	18.41	5.14

4. táblázat. Az eredeti és a 2. algoritmus lokalizációs hibája és futásideje k=4 esetén

minden részgráfot használunk az egyesítés során, hiszen a nagy átfedések miatt általában nem kapnánk sokkal rosszabb eredményt, azonban így csökkenthetünk az algoritmus futási idején. Például megtehetjük, hogy egy csúcshoz tartozó részgráfot eldobunk, ha az adott csúcs már legalább k olyan részgráfban szerepel, melyet már összeillesztettünk. A 3. táblázat és a 8. ábra mutatja hogyan alakul a lokalizációs hiba és az algoritmus futásideje $k = 4$ esetén. Látható, hogy az algoritmus futásideje jelentősen csökken, hiszen 100 részgráf helyett mindössze 29-et illesztünk össze, azonban a lokalizációs hiba csak kis mértékben nőtt.



12. ábra. A 2. módosított MDS-MAP(P) algoritmus eredménye 100 egyenletesen elhelyezett szenzor esetén.

6. Ortogonális Prokrusztész probléma

Az MDS-MAP(P) algoritmus egyik lépése során a $G_1 = (V_1, E_1)$ geometriai gráfot szeretnénk úgy transzformálni, hogy a $V_0 \cap V_1$-beli pontok G_0 és G_1-beli példányainak távolságösszege minimális legyen. Azonban eddig nem tértem ki rá, hogyan tehetjük ezt meg. Szintén hasonló problémába ütközünk, ha szeretnénk a fenti, horgonypontot nem használó algoritmusok hibáját mérni. Ahhoz, hogy a szenzorok helyzetét meg tudjuk határozni, szükségünk van legalább három horgonypontra. Ha háromnál kevesebb szenzor koordinátáit ismerjük, akkor a szenzoroknak csak az egymáshoz viszonyított helyzetét tudjuk meghatározni. Ezért az algoritmus hibájának méréséhez az algoritmus eredményeként kapott koordinátákat úgy kell transzformálnunk, hogy azok minél közelebb legyenek valós megfelelőjükhöz, majd az így kapott koordináták segítségével számítjuk ki a hiba nagyságát.

Legyen $\mathbf{A} \in \mathbb{R}^{2\times n}$ a szenzorok koordinátáinak mátrixa, és legyen $\mathbf{B} \in \mathbb{R}^{2\times n}$ az algoritmus eredményeként kapott koordináták mátrixa. Szeretnénk meghatározni azt a transzformációt leíró $\mathbf{Q} \in \mathbb{R}^{2\times 2}$ ortonormált mátrixot, melyre $\mathbf{A} - \mathbf{QB}$ Frobenius-normája minimális. Tehát a probléma az alábbi alakba írható:

$$min\|\mathbf{A} - \mathbf{QB}\|_F^2 \tag{4}$$

$$\mathbf{Q}^\top\mathbf{Q} = \mathbf{I} \tag{5}$$

A problémát először Peter H. Schönemann oldotta meg 1964-ben készült szakdolgozatában, majd 1966-ban egy pszichológiai lapban publikálta [33].

6.1. Állítás. *A fenti feltételeket teljesítő mátrix:*

$$\mathbf{Q} = \mathbf{UV}^\top,$$

ahol $\mathbf{Q} = \mathbf{U\Sigma V}^\top$ *az* $\mathbf{AB}^\top$ *mátrix szinguláris érték szerinti felbontása.*

Az alábbi bizonyítás [34] alapján készült.

Bizonyítás:

$$\|\mathbf{A}-\mathbf{QB}\|_F^2 = tr\left[(\mathbf{A}-\mathbf{QB})(\mathbf{A}^\top - \mathbf{B}^\top\mathbf{Q}^\top)\right] = \tag{6}$$

$$= tr(\mathbf{AA}^\top) - 2tr(\mathbf{AB}^\top\mathbf{Q}^\top) + tr(\mathbf{QBB}^\top\mathbf{Q}^\top) = \tag{7}$$

$$= tr(\mathbf{AA}^\top) - 2tr(\mathbf{AB}^\top\mathbf{Q}^\top) + tr(\mathbf{B}^\top\mathbf{Q}^\top\mathbf{QB}) = \tag{8}$$

$$= tr(\mathbf{AA}^\top) - 2tr(\mathbf{AB}^\top\mathbf{Q}^\top) + tr(\mathbf{B}^\top\mathbf{B}) \tag{9}$$

A (7) és a (8) közötti egyenlőség abból következik, hogy $tr(\mathbf{XY}) = tr(\mathbf{YX})$, a (8) és a a (9) közötti egyenlőség pedig abból, hogy $\mathbf{Q}$ ortonormált mátrix. Mivel $tr(\mathbf{AA}^\top)$ és $tr(\mathbf{B}^\top\mathbf{B})$ konstans érték, ezért a továbbiakban a feladatunk $tr(\mathbf{AB}^\top\mathbf{Q}^\top)$ maximalizálása.

Legyen $\mathbf{Z} = \mathbf{V}^\top\mathbf{Q}^\top\mathbf{U}$. Ekkor

$$tr(\mathbf{AB}^\top\mathbf{Q}^\top) = tr(\mathbf{U\Sigma V}^\top\mathbf{Q}^\top) = tr(\mathbf{\Sigma V}^\top\mathbf{Q}^\top\mathbf{U}) \tag{10}$$

$$= tr(\mathbf{\Sigma Z}) = \sum_{i=1}^{n} \mathbf{Z}_{i,i}\mathbf{\Sigma}_{i,i} \tag{11}$$

Figyeljük meg, hogy $\mathbf{Z}$ ortogonális, mert ortogonális mátrixok szorzata. $\mathbf{\Sigma}$ diagonalitása miatt $\sum_{i=1}^{n} \mathbf{Z}_{i,i}\mathbf{\Sigma}_{i,i}$ akkor maximális, ha $\mathbf{Z}_{i,i} = 1 \quad (i = 1 \ldots n)$. Ekkor tehát $\mathbf{Z} = \mathbf{I}$, így $\mathbf{Q} = \mathbf{UV}^\top$. ■

7. Összefoglalás

Szakdolgozatomban ismertettem a szenzorlokalizációs problémát, és bemutattam néhány algoritmus típust, amelyekkel próbálják megoldani ezt a problémát. Öt algoritmust részletesebben is bemutattam. Ezek közül kettő horgonypontokat használ, a többi három azonban kizárólag a szenzorok távolságadataira hivatkozva próbálja meghatározni azok relatív helyzetét.

Bemutattam az ortogonális Prokrusztész problémát és annak Peter H. Schönemanntól származó megoldását, mely hasznos eszköz a hiba kiszámítására a csak relatív helyzetet meghatározó algoritmusoknál.

Három algoritmus esetén javaslatot tettem az algoritmus módosítására, és az általam készített MATLAB program segítségével konkrét példákon mutattam be hogyan változik az algoritmus futásideje és/vagy hibája.

8. Hivatkozások

[1] Le Dinh, Tuan, et al. "Design and deployment of a remote robust sensor network: Experiences from an outdoor water quality monitoring network." *Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on.* IEEE, 2007.

[2] Mainwaring, Alan, et al. "Wireless sensor networks for habitat monitoring." *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications.* Acm, 2002.

[3] Kumar, Vijay, Daniela Rus, and Sanjiv Singh. "Robot and sensor networks for first responders." *IEEE Pervasive computing*, 2004, 3.4: 24-33.

[4] Li, Shen, et al. "Efficient Emergency Rescue Navigation with Wireless Sensor Networks." *J. Inf. Sci. Eng.*, 2011, 27.1: 51-64.

[5] https://newsroom.accenture.com/subjects/technology/new-accenture-offering-uses-sensor-technology-to-help-companies-deploy-wireless-applications.htm

[6] Yu, Liyang, Neng Wang, and Xiaoqiao Meng. "Real-time forest fire detection with wireless sensor networks." *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on.* Vol. 2. IEEE, 2005.

[7] Girod, Lewis, and Deborah Estrin. "Robust range estimation using acoustic and multimodal sensing." *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on. Vol. 3.* IEEE, 2001.

[8] Prorok, Amanda, Phillip Tomé, and Alcherio Martinoli. "Accommodation of NLOS for ultra-wideband TDOA localization in single-and multi-robot systems." *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on.* IEEE, 2011.

[9] Bahl, Paramvir, and Venkata N. Padmanabhan. "RADAR: An in-building RF-based user location and tracking system." *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE.* Vol. 2. Ieee, 2000.

[10] Rong, Peng, and Mihail L. Sichitiu. "Angle of arrival localization for wireless sensor networks." *Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on. Vol. 1.* IEEE, 2006.

[11] Chen, Hongyang, et al. "Novel centroid localization algorithm for three-dimensional wireless sensor networks." *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on.* IEEE, 2008.

[12] Jichun, Wang, et al. "A Novel Range Free Localization Scheme Based on Voronoi Diagrams in Wireless Sensor Networks [J]." *Journal of Computer Research and Development*, 2008, 1: 014.

[13] Han, Guangjie, Deokjai Choi, and Wontaek Lim. "Reference node placement and selection algorithm based on trilateration for indoor sensor networks." *Wireless Communications and Mobile Computing*, 2009, 9.8: 1017-1027.

[14] Sugano, Masashi, et al. "Indoor localization system using RSSI measurement of wireless sensor network based on ZigBee standard." *Target*, 2006, 538: 050.

[15] Bulusu, Nirupama, John Heidemann, and Deborah Estrin. "GPS-less low-cost outdoor localization for very small devices." *IEEE personal communications*, 2000, 7.5: 28-34.

[16] Xiao, Ling, Renfa Li, and Juan Luo. "Sensor localization based on non-metric multidimensional scaling." STRESS, 2006, 2: 1.

[17] He, Tian, et al. "Range-free localization schemes for large scale sensor networks." *Proceedings of the 9th annual international conference on Mobile computing and networking.* ACM, 2003.

[18] Niculescu, Dragos, and Badri Nath. "DV based positioning in ad hoc networks." *Telecommunication Systems* 22.1-4 (2003): 267-280.

[19] Tian, Shuang, et al. "A selective anchor node localization algorithm for wireless sensor networks." *Convergence Information Technology, 2007. International Conference on.* IEEE, 2007.

[20] Savarese, Chris, Jan M. Rabaey, and Jan Beutel. "Locationing in distributed ad-hoc wireless sensor networks." *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on.* Vol. 4. IEEE, 2001.

[21] Moore, David, et al. "Robust distributed network localization with noisy range measurements." *Proceedings of the 2nd international conference on Embedded networked sensor systems.* ACM, 2004.

[22] Shang, Yi, et al. "Localization from mere connectivity." *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing.* ACM, 2003

[23] Azim, Mohammad Abdul, et al. "Localization in wireless sensor networks by cross entropy method." *International Conference on Ad Hoc Networks.*Springer Berlin Heidelberg, 2012.

[24] Rubinstein, Reuven Y., and Dirk P. Kroese. "The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning." Springer Science & Business Media, 2013.

[25] Kullback, Solomon, and Richard A. Leibler. "On information and sufficiency." *The annals of mathematical statistics*, 1951, 22.1: 79-86.

[26] Azim, Mohammad Abdul, et al. "Constrained cross entropy localization technique for wireless sensor networks." *International Journal of Distributed Sensor Networks* 11.5: 267369, 2015.

[27] Savvides, Andreas, Chih-Chieh Han, and Mani B. Strivastava. "Dynamic fine-grained localization in ad-hoc networks of sensors." *Proceedings of the 7th annual international conference on Mobile computing and networking.* ACM, 2001.

[28] http://www.stat.pitt.edu/sungkyu/course/2221Fall13/lec8_mds_combined.pdf

[29] Shang, Yi, et al. "Localization from mere connectivity." *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing.* ACM, 2003.

[30] Shang, Yi, and Wheeler Ruml. "Improved MDS-based localization." *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. Vol. 4.* IEEE, 2004.

[31] Marquardt, Donald W. "An algorithm for least-squares estimation of nonlinear parameters." *Journal of the society for Industrial and Applied Mathematics*, 11.2: 431-441, 1963.

[32] Priyantha, Nissanka B., et al. "Anchor-free distributed localization in sensor networks." *Proceedings of the 1st international conference on Embedded networked sensor systems.* ACM, 2003.

[33] Schönemann, Peter H. "A generalized solution of the orthogonal Procrustes problem." *Psychometrika*, 1966, 31.1: 1-10.

[34] Wang, Jianfeng. "Proof of the Orthogonal Procrustes Problem.", 2015.

[35] Hu, Zhen, et al. "Localization in wireless sensor networks using a mobile anchor node." *Advanced Intelligent Mechatronics, 2008. AIM 2008. IEEE/ASME International Conference on.* IEEE, 2008.

[36] Koutsonikolas, Dimitrios, Saumitra M. Das, and Y. Charlie Hu. "Path planning of mobile landmarks for localization in wireless sensor networks." *Computer Communications* 30.13: 2577-2592, 2007.

[37] Ou, Chia-Ho, and Kuo-Feng Ssu. "Sensor position determination with flying anchors in three-dimensional wireless sensor networks." *IEEE Transactions on Mobile Computing* 7.9: 1084-1097, 2008.