Sebők Dávid

Diplomaterv

Budapesti Műszaki és Gazdaságtudományi Egyetem

Gépészmérnöki Kar

Épületgépészeti és gépészeti eljárástechnika Tanszék



Diplomatervek

Budapesti Műszaki és Gazdaságtudományi Egyetem

Gépészmérnöki Kar

épületgépészeti és gépészeti eljárástechnika Tanszék

Sebők Dávid

Diplomaterv

Távhőrendszer tervezése optimalizációs módszerrel

|  |  |
| --- | --- |
| Konzulens:  *Csirmaz István*  Direkt Kft. | Témavezető:  *Dr. Jasper Andor*  Tudományos segédmunkatárs |

Budapest, 2018

Szerzői jog © Sebők Dávid, 2018.

Szerzői jog © Dr. Jasper Andor, 2018.

Ide kell befűzni az eredeti feladatkiírási lapot!

Nyilatkozatok

*Beadhatósági nyilatkozat*

A jelen tervezési feladat az üzem által elvárt szakmai színvonalnak mind tartalmilag, mind formailag megfelel, beadható.

Kelt, 2018.05.08.

Az üzem részéről:

*üzemi konzulens*

*Elfogadási nyilatkozat*

Ezen tervezési feladat a Budapesti Műszaki és Gazdaságtudományi Egyetem Épületgépészeti és Gépészeti Eljárástechnika Tanszéke által a Diplomatervezési és Szakdolgozat feladatokra előírt valamennyi tartalmi és formai követelménynek maradéktalanul eleget tesz. E tervezési feladatot a nyilvános bírálatra és nyilvános előadásra alkalmasnak tartom.

A beadás időpontja: 2018.05.08.

*témavezető*

*Nyilatkozat az önálló munkáról*

Alulírott, *Sebők Dávid* (JIIKQT), a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója, büntetőjogi és fegyelmi felelősségem tudatában kijelentem és sajátkezű aláírásommal igazolom, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomaterv feladatomban csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Budapest, 2018.05.08.

*szigorló hallgató*

Tartalomjegyzék

[Előszó viii](#_Toc208583190)

[Jelölések jegyzéke ix](#_Toc208583191)

[1. Bevezetés 1](#_Toc208583192)

[1.1. Célkitűzések 1](#_Toc208583193)

[1.2. Áttekintés 2](#_Toc208583194)

[1.2.1. Elvek és módszerek 2](#_Toc208583195)

[1.2.1.1. Első módszer 2](#_Toc208583196)

[1.2.1.2. Második módszer 2](#_Toc208583197)

[1.2.2. Alapfeltevések 2](#_Toc208583198)

[2. Szakirodalmi áttekintés 3](#_Toc208583199)

[2.1. Az első főfeladathoz 3](#_Toc208583200)

[2.2. A második főfeladathoz 3](#_Toc208583201)

[3. A feladat kibontása/Első főfeladat 4](#_Toc208583202)

[3.1. Első részfeladat 4](#_Toc208583203)

[3.2. Második részfeladat 4](#_Toc208583204)

[3.3. N-edik részfeladat 4](#_Toc208583205)

[4. Második főfeladat 5](#_Toc208583206)

[4.1. Első részfeladat 5](#_Toc208583207)

[4.2. Második részfeladat 5](#_Toc208583208)

[4.3. N-edik részfeladat 5](#_Toc208583209)

[5. Összefoglalás/Eredmények értékelése 6](#_Toc208583210)

[5.1. Eredmények 6](#_Toc208583211)

[5.2. Javaslatok/Következtetések/Tanulságok 6](#_Toc208583212)

[6. Felhasznált források 7](#_Toc208583213)

[7. Summary 8](#_Toc208583214)

# Előszó

Az egyetemi pályafutásom első évétől kezdve kiemelkedően sok figyelmet szenteltem a programozásra. Úgy érzem, itt illik elővennem az anekdotát, miszerint 2009 nyarán fontosabbnak tartottam javítani a programtervezés 4-es érdemjegyemet, mint foglalkozni a tanterv gerincét képző tárgyakkal. A javítás sikerült, az utóbbi tárgyakból viszont többet kellett ismételnem. Büszke vagyok rá, hogy gépészmérnök lehetek és nem bántam meg, hogy ebbe az irányba tanultam tovább. Hálás vagyok, hogy a mérnöki szemléletmód elsajátítása mellett volt lehetőségem több tantárgynál, a szakdolgozatomnál és a jelenlegi diplomamunkámnál is programozási eszközökkel dolgoznom. Amíg az alapképzés végén egy magasabb szintű nyelvvel oldottam meg a szakdolgozatom tárgyát, úgy jelenleg egy alapvető és közismert nyelvben tudtam kiteljesedni. Számomra a mesterképzés célja, hogy a hallgató megtanuljon rendszerben gondolkodni, illetve egyre magasabb absztrakciós szinteket használni. A Távhőellátás, illetve Csőhálózatok hidraulikája tantárgyak során oktatóim erős alapokra helyezve minden lényeges tudást átadtak, amire nagy szükségem volt a jelen feladatkör elvégzésére és a program megalkotására. A közös munkának köszönhetően egy stabil és elég magas absztrakciós szintet tudtam elérni, ami elengedhetetlen volt ahhoz, hogy genetikus algoritmust alkalmazzak optimalizálásra.

\* \* \*

Nehéz lenne minden embert felidézni, aki hozzásegített az egyetem elvégzéséhez. Elsősorban a családomnak szeretném kifejezni hálámat, akik mindvégig támogattak és nem adták fel a belém vetett hitüket. Köszönöm a rengeteg segítséget csoporttársaimnak, kollégáimnak, akikből mostanra jó barátok váltak. Hálás vagyok sok tanárnak, de különösen Dr. Garbai Lászlónak, aki hitet és motivációt adott a mesterképzéshez. Köszönettel tartozom Csirmaz Istvánnak, aki esélyt adott arra, hogy az iparban is bizonyítsak, illetve amiért időt tudott szakítani a szakdolgozatom elbírálására. Végül, de nem utolsó sorban Dr. Jasper Andornak szeretném megköszönni oktatását, tanácsait, megértését és rengeteg türelmét, ami nélkül most nem írhatnám e sorokat.

Budapest, 2018.01.08.

Sebők Dávid

# Jelölések jegyzéke

A táblázatban a többször előforduló jelölések magyar és *angol* nyelvű elnevezése, valamint a fizikai mennyiségek esetén annak mértékegysége található. Az egyes mennyiségek jelölése – ahol lehetséges – megegyezik hazai és a nemzetközi szakirodalomban elfogadott jelölésekkel. A ritkán alkalmazott jelölések magyarázata első előfordulási helyüknél található.

*A táblázatokat ABC rendben kell feltölteni, először mindig a kisbetűvel kezdve. Ha egyazon betűjelnek több értelmezése is van, akkor mindegyiket külön sorban kell feltüntetni. Konstansok esetén az értéket is a táblázatba kell írni. Dimenzió nélküli mennyiségek mértékegysége* 1 *és nem: – ! A jelölésjegyzékben csak SI vagy SI-n kívüli engedélyezett mértékegységeket szabad feltüntetni. Egy dokumentumon belül az SI és pl. az angolszász mértékrendszer nem keverhető! Ezt a bekezdést a végső szövegváltozatból törölje ki!*

Latin betűk

|  |  |  |
| --- | --- | --- |
| Jelölés | Megnevezés, megjegyzés, érték | Mértékegység |
| *g* | gravitációs gyorsulás (9,81) | m/s2 |
| *p* | nyomás | bar |
|  | szélprofil-kitevő | 1 |
| *s* | fajlagos entrópia | J/(kg·K) |

Görög betűk

|  |  |  |
| --- | --- | --- |
| Jelölés | Megnevezés, megjegyzés, érték | Mértékegység |
|  |  |  |
|  |  |  |

Indexek, kitevők

|  |  |
| --- | --- |
| Jelölés | Megnevezés, értelmezés |
| *i* | általános futóindex (egész szám) |
| nom | névleges (nominális) érték |
| opt | legkedvezőbb (optimális) érték |
|  |  |

# Bevezetés

## Célkitűzések

Magyarországon jelenleg újra fellendült az építőipar és ezzel együtt a távhőellátási beruházások is megsűrűsödtek. A piaci verseny kiélezettsége miatt és a jó hírnév, illetve referenciák eléréséhez elengedhetetlen, hogy egy cég megbízható számításokkal dolgozzon, valamint tervezés során a lehető legközelebb kerüljön az optimumhoz.

A kapcsolódó tantárgyaknak köszönhetően minden olyan számítással találkoztam, ami biztosítja, hogy egy rendszer pontosan működjen és ne teljesítsen alul. Mesterképzésen a Távhőellátás c. tárgyunk során pedig sokat foglalkoztunk optimum-kereséssel. Mátészalka távhőellátó rendszerét vizsgáltuk: az előadások során kiderült, hogy a hálózat tervezői csupán rendszerfüggetlen optimummal számoltak és nem aknázták ki azokat a lehetőségeket, mellyel a beszabályozás során is komoly összeget lehet megtakarítani.

Rendszerfüggetlen optimummal számolva a rendszer éves összköltsége 19 906 722 Ft lenne. Ez egy ideális érték, ami gyakorlatilag akkor valósulhatna meg, ha minden egyes vezetékhez egy minimalizált, egyedi hőforrás csatlakozna. Ennek a fölöslegessége egyértelmű – már csak azért is, mert ez önmagában igen nagy mértékben megnövelné a beruházási költséget. Maradjunk a rendszernél és a hozzá tartozó egy darab hőforrásnál. Ebben az esetben ehhez az összeghez és paraméterezéshez egy olyan hálózat tartozna, amely nincs beszabályozva, így ugyan az összesített térfogatárama megfelel a tervezettnek, de az egyes fogyasztókhoz már egyenként közel sem a szükséges térfogatáram, ill. hőáram érkezne.

Beszabályozás után ugyanennek a hálózatnak az éves összköltsége 22 745 306 Ft. Annak idején a tervezők itt álltak meg. Az egyik lehetőség a fojtási optimalizálásban rejlik, amelynek lényege, hogy fojtáskor ellenőrizzük, hogy az adott szakasz nyomásesése elért-e egy akkora értéket, ami megengedi a fojtás kiváltását átmérőcsökkentéssel. Vannak esetek, amikor több, mint egy mérettel lehet redukálni egy szakasz vezetékeit. Minden alkalommal, amikor ezt meg tudjuk tenni, tisztán csökkentjük a szakasz, s ezzel az egész rendszer beruházási költségét.

Ezt a műveletet dinamikus programozással tanácsos végrehajtani. A folyamat során a hidraulikai végpontból (melynek változatlanul hagyjuk az átmérőjét) kell elindulni, majd szakaszról szakaszra haladva a fojtást és az átmérők csökkentését egyszerre érdemes elvégezni. Ezek után már egy gazdaságilag kedvezőbb, viszont ugyanolyan hatékonyan működő rendszer áll rendelkezésünkre.

A fojtási optimalizálás után a rendszer tervezett éves összköltsége 22 197 061 Ft lett. Ez 548 244 Ft értékű, illetve 2,4%-os csökkenés – ami jelentős mértékű, főleg, hogy könnyen algoritmizálható, így alig igényel extra munkát.

Az elméleti (elérhetetlen) minimum és ez eddigi legalacsonyabb költség között maradt 2 290 340 Ft. A fogyasztókhoz tartozó hőigények, ergo térfogatáramok és a hálózat kialakítása miatt az optimálásban maradt teljes potenciál ezt biztosan nem éri el. Viszont úgy gondolom, hogy a költségeken még lehet faragni. Ez ihlette a diplomamunkámat, emiatt írtam az optimum-kereső programomat. A kulcs a rendszerben való gondolkodás: egy vagy több meghatározó jelentőségű szakasz eddigi optimális értékének elmozdításával ugyan helyi költségnövekedés fog fellépni, de az egész rendszer összköltsége ezzel együtt is csökkenni fog. Részletezni fogom ennek az okát, illetve bemutatom az algoritmust, melyet megalkottam és amivel a munkám során dolgoztam.

## Áttekintés

A bevezetést, illetve ezt a fejezetet néhány nevezetesebb optimalizálási eljárás követi: ismertetem a dinamikus programozás, a Newton-Raphson módszer, illetve a genetikus algoritmusok alapelveit és tulajdonságait. Ezek után részletesen bemutatom a genetikus algoritmusok felépítését és működését az Utazó ügynök (Traveling Salesman Agent) példáján keresztül. Ezt követi a feladat során vizsgált távhővezeték-rendszer tervezése: a hálózat struktúrájának bemutatása, az alap- és kiindulási adatok felsorolása, a számításokhoz szükséges képletek, diagramok és elméletek leírása, illetve nyomásesések, rendszerfüggetlen optimum, inverz feladat, beszabályozás és a fojtási optimum kiszámolása. Továbbiakban az általam fejlesztett genetikus algoritmust fogom ismertetni, amely minden egyes iterációval újratervezi a rendszert pontosan úgy, ahogy azt a korábbi fejezet bemutatja. Az utolsó fejezetkben magyar és angol nyelven közlöm a következtetéseimet, illetve azokat az irányokat, amin mentén még érdemes foglalkozni a feladattal. A diplomamunka legvégén találhatóak a források, illetve a függelék, ami további ábrákat, illetve a minden egyes felhasznált algoritmus forráskódját tartalmazza.

# optimalizálási módszerek

## Dinamikus programozás

Egy döntési folyamat döntések sorozatából, döntési fokozatokból áll. Egy döntési folyamatnak az a jellegzetessége, hogy a megelőző döntések sorozata egyértelműen meghatározza az aktuális döntés előtt a rendszer állapotát. Következésképpen minden döntés függ az őt megelőző döntésektől, és befolyásolja az őt követő döntéseket.

Azt a matematikai módszert, amely egymástól függő döntések sorozatát optimálja, dinamikus programozásnak nevezzük. A döntések sorozatában minden döntés számára a döntés kritériuma a dinamikus programozás optimalitási elvének kielégítése.

Az optimalitás elve eredeti megfogalmazásában kimondja, hogy:

„Egy optimális döntéssorozat mindig rendelkezik azzal a tulajdonsággal, hogy bármilyen legyen is a kezdeti állapot és a kezdeti döntés, a további döntés, a további döntések optimális döntéssorozatot alkotnak az első döntés és a kezdeti állapot eredményeképpen kialakult új állapothoz képest”.

Tehát az optimalitás elvének értelmében az egyes fokozatokban a döntéseknek úgy kell létrejönniük, hogy figyelembe vesszük az egész döntési folyamat lehetőségeit, és a rendszer adott állapotához képest ennek optimálisnak kell lennie. Döntéseink matematikai értelemben általánosságban egy rekurzív függvényegyenletet elégítenek ki. A függvényegyenlet – az úgynevezett soros rendszerre – az általános döntési folyamat jelöléseivel a következő:



illetve az adott fokozatot jellemző állapottranszformáció bevezetésével:



A függvényegyenlet az állapottranszformáció alkalmazásával már csak a kapcsolatváltozó (*z*m) és a döntési változó (*u*m) függvénye. Az O(*z*m) az úgynevezett optimális függvény, amely minden *z*m kapcsolatváltozó értékhez tartalmazza a döntési fokozatokhoz tartozó optimális döntéseket.

**A soros rendszerek optimalizációja**

A soros rendszerek (serial system) absztrakt döntési modelljét az 1. ábra mutatja. A döntési modellt white box modellnek is nevezik. A modell tartalmazza a döntési változókat (*u*m), a döntési eredményeket (*f*m), és a bemenetek-kimenetek közötti összefüggéseket (többnyire mérlegegyenletek) (*g*m). Az optimalizáció történhet visszafelé (backward), illetve előrehaladó (forward) rekurzióval. Leggyakrabban a visszafelé haladó rekurzív optimalizálást alkalmazzuk, tekintve, hogy a rendszer kimenete, amely általában valamely fogyasztói igény, például hőigény, adottnak tekinthető A visszafelé haladó rekurzióban a függvényegyenletet rekurzíven, az utolsó fokozattól kezdve, az állapottranszformációk backward analízisével fokozatonként megoldjuk. Vagyis az adott fokozatban, a *z*m kapcsolatváltozó függvényében, a megvizsgált fokozatok összességére nézve az optimális megoldások halmazát állítjuk elő az *U*m döntési változó alkalmas megválasztásával.

A rekurzív függvényegyenlet fokozatról-fokozatra történő felírásával és megoldásával a feladatot részfeladatok sorozatára bontjuk.

**Soros rendszer optimalizációja visszafelé haladó rekurzióval**

A célfüggvény:



Az optimalizáció tulajdonképpen feltételes szélsőérték-számítás, ahol a feltételi egyenleteket a boxokba írt transzformációs összefüggések képezik. A független változók a döntési változók, amelyek alkalmas megválasztásával a célfüggvényt minimalizálhatjuk.

**Első lépés:**

Felírjuk a függvényegyenletet az utolsó tagtól kezdve:



Fennáll, hogy:



A függvény inverzének segítségével kifejezzük *u*n-1-et:



Ezt behelyettesítjük a (4) egyenletbe, és így kapjuk:



Ez után a lépés után a (7) célfüggvényünk optimuma csak a bemeneti és a kimeneti változótól függ. A *z*n viszont általában adott, hiszen *z*n a kielégítendő igény.

**Második lépés:**

A soron következő boxra is felírjuk az optimalizációs egyenletet, amiben már szerepel az előző optimumfüggvény is:



A boxban lévő egyenlet felhasználásával:



Tehát:



Mivel az egyenletünket *u*n-2 értékre minimalizáljuk, az *u*n-2 optimális értékének behelyettesítésével csak *z*n-2 marad meg paraméternek. Az optimalizációt tovább folytatva meghatározzuk *O*m értéket a fent bemutatott módszer segítségével, majd azt követően eljutunk *O*1 értékhez, ami az egész rendszer optimumát fogja megadni.

A soros kapcsolás megoldható előrehaladó rekurzió használatával is. Ennek a rekurziónak a használatával kiküszöbölhető *u*1, de *z*1, *z*2 megmarad. Ha *z*1 értékét ismerjük, akkor az első lépés után elhagyhatjuk, ha ismeretlen, az egész optimalizációs folyamat során szerepelni fog a függvényeinkben, mint ismeretlen.

**Nem-soros rendszerek vizsgálata és optimalizációja**

A nem-soros rendszerek optimalizációja általában felépíthető a soros rendszerekre bemutatott rekurzív függvényegyenletek optimalizációjából. Ugyanis az egyszerűbb nem-soros rendszerek felbonthatóak (felvághatóak) soros rendszerű ágakra.

## Newton-Raphson módszer

Az angol nyelvű wikipedia lap [] részletesen ismerteti a módszert, a hivatkozást azért érdemes megtekinteni, mert tartalmaz egy animációt, amelynek segítségével másodpercek alatt megérthető optimalizáció elmélete.

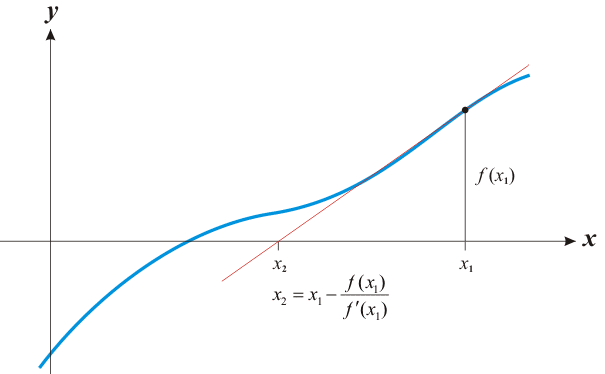
Az iterálási módszert 1 dimenzióra a -es egyenlet tartalmazza:



A módszer megállapítja a kezdeti feltételként megadott helyen a megadott függvény értékét és meredekségét (hely szerinti deriváltját), a kettő hányadosa pedig a kezdeti hely (*x*n) és az érintő x tengellyel való metszéspontjának (*x*n+1) a különbségével egyezik meg. Erre mutat példát a -es és -as egyenlet, valamint a 3.1. ábra. Ha a függvény értéke és meredeksége is pozitív, vagy ha mindkettő negatív, akkor a következő lépés helye (*x*n+1) balra / negatív irányba lesz a kezdeti helytől, ha viszont előjelük ellentétes, akkor jobbra / pozitív irányba.







3.1. ábra: Az 1 dimenziós Newton-Raphson módszer egy iterációs lépése [33]

A lépések addig folytatódnak, amíg az algoritmus nem éri el a kilépési feltételt. (Egy apró módosítással bármilyen érték kereshető, de módszernek elsősorban nem ez a célja). Több megoldás esetén az optimalizáció csak 1-et talál meg, és nem feltétlenül a kezdeti értékhez (geometriailag) legközelebbit. Mivel ez csak egy közelítő módszer, ezért az esetek többségében végtelen lépés lenne szükséges a zérushely pontos helyének megállapításához. Az algoritmus 2 feltétel függvényében hagyja abba az iterálást: egyrészt meg kell határozni a maximális iterációs lépések számát, másrészt a zérushely távolságának toleranciáját. Érdemes megemlíteni, hogy rossz kezdeti feltétel megválasztása esetén (pl.: 0 értékű a görbe adott pontjához tartozó meredekség, így a végtelenben metszené az x tengelyt az érintő) a program nem talál megoldást. Egy program tartalmazhat olyan sorokat, melyek ezt a problémát is kiküszöbölik, de mivel igen ritka eset, ezért nem létfontosságú.

A -es egyenlet alapján a Newton-Raphson módszer felírható több dimenzióra is:







Ahol: - **x**n+1: a függvények változóinak helyeit tartalmazó vektor az n+1-edik lépésben.

* *x*j,n+1: a j-edik változó értéke az n+1-edik lépésben.
* **x**n: a függvények változóinak helyeit tartalmazó vektor az n-edik lépésben.
* *x*j,n: a j-edik változó értéke az n-edik lépésben.
* **J**(**x**n): az egyenletrendszer Jacobi mátrixa az n-edik lépésben.
* **J-1**(**x**n): az egyenletrendszer Jacobi mátrixának inverze az n-edik lépésben.
  + : az i-edik függvény (i-edik egyenlet nullára rendezve) j-edik változó szerinti deriváltja.
  + : az egyenletek értékeit tartalmazó vektor az n-edik lépésben.
  + : az i-edik függvény (i-edik egyenlet nullára rendezve) értéke az n-edik lépésben.

#### A genetikus algoritmusok elve

Mi az a genetikus algoritmus?

Az emberiség tudományos fejlődésében mindig is szerepet játszottak az empirikus

megfigyeléseken alapuló elméletek. Ha valami készen kapva működik a nagyvilágban, azt miért ne lehetne felhasználni saját céljainkra, esetleg egy kicsit elvonatkoztatva, általánosítva? Rengeteg matematikai probléma, és ezen keresztül új szakterület alakult ki igen egyszerű hétköznapi problémák megfigyelésével. Nincs másképp ez a genetikus ill. evolúciós algoritmusokkal sem: itt a példát a biológia (szebben mondva az anyatermészet) adta. Alapja, Darwin: „A fajok eredete” című könyve 1859-ben jelent meg, és már az első napon elfogyott minden példánya. Azóta is erősen megosztja mind a tudós, mind a laikus társaságokat.

Természetesen nem célom ezzel kapcsolatban állást foglalni, csak az elmélet gyökereire kívánok rávilágítani. A genetikus algoritmusok potenciális megoldás gyűjteményekkel dolgoznak, különböző stratégiák során csoportosítják, majd a „jobb” megoldásokat kiválogatva igyekeznek mindig új állományokat kreálva eljutni egy elfogadható (esetleg az optimális) megoldáshoz. A megoldások gyűjteményét populációnak hívjuk, a benne lévő lehetséges megoldásokat egyedeknek. Az egyedekből különböző szelekciókkal és mutációkkal új egyedeket képzünk (születnek), és igyekszünk kiválogatni közülük azokat, amelyek számunkra jobb tulajdonságokkal rendelkeznek, azaz „rátermettebbek”. Több generáción keresztül ismételve ezt a folyamatot, egyre jobb, pontosabb lehetséges megoldásokat kapunk. Véges lépés után, vagy egy kellően pontos megoldás esetében az algoritmus leáll.

Hogyan alakult ki?

Genetikus algoritmusok ötlete egyidős a számítógép ill. a számítógépes programozás megjelenésével. Az alapkoncepció már megjelent az 1950-es években, természetes evolúciókutatással foglalkozó biológusok kívánták elsőként az elméleti eredményeket minél pontosabb szimulációval vizsgálni. Akkor még nem merült fel bennük, hogy a kidolgozott módszerek szélesebb körben is használhatóak. A 60-as évek elején többen egymástól függetlenül kidolgozták az evolúció által inspirált algoritmusaikat, főként függvényoptimalizálásra és gépi tanulásra, de ezek a munkák kevés érdeklődésre tettek szert.

1965-ben Ingo Rechenberg publikálta az evolúciós stratégiák alapjait, ami a genetikus algoritmusok egy rokon ága. Az elméletből még hiányoztak alapvető ma használt fogalmak, de a terület fejlődésének lökést adott. Nem volt még keresztezés, csak önmagukat reprodukáló egyedek, viszont itt vetődött fel először a populáció fogalma.

1966 egy új irányág, az evolúciós programozás születését hozta. A lehetséges megoldásokat ebben a verzióban véges állapotú automaták reprezentálták, egy automata állapotátmenet mátrixának véletlenszerű mutációja hozott létre egy új egyedet, majd az algoritmus a kettő közül jobbnak ítéltet tartotta meg. A keresztezés jelentőségét itt sem ismerték még fel.

John Holland és csoportja volt az első, akik hivatalosan bevezették a keresztezés fogalmát, rávilágítva annak előnyeire. A hatvanas évek elejétől dolgozott tanuló rendszereken, eredményeit csak 1975-ben publikálta. Ez tekinthető a genetikus algoritmusok ma is használt elméleti alapjainak. A 80-as évek közepére az evolúciós programozás különböző válfajai széles körben használtak voltak. A számítási nagyság növekedésével és az Internet rohamos terjedésével jelentősége és használhatósága széles körben kiteljesedett. Míg az első időkben főként elméleti, napjainkban az élet számos terén alkalmazzák gyakorlati problémák megoldására.

# Szakirodalmi áttekintés/Előzmények

## Genetikus algoritmusok bemutatása az utazó ügynök problémán keresztül

A genetikus algoritmus

A genetikus algoritmus (továbbiakban: GA) egy globális optimalizációs eljárás. Mindenhol alkalmazható, ahol a feladat sok lehetséges megoldás közül a legjobbat megkeresni. A genetikus algoritmus lehetséges megoldások egy populációját tartja fenn, azaz egyszerre több megoldáskezdeménnyel dolgozik. A lehetséges megoldások minőségét egy értékelő-, vagy más néven rátermettségi függvény adja meg. Az aktuális populációból minden lépésben egy új populációt állít elő úgy, hogy a szelekciós operátor által kiválasztott legrátermettebb elemeken (szülőkön) alkalmazza a rekombinációs- és mutációs operátorokat. Az alapgondolat az, hogy mivel általában minden generáció tagjai az előzőnél rátermettebbek, a keresés folyamán egyre jobb megoldások állnak rendelkezésre.

Az utazó ügynök probléma

Az utazó ügynök problémája egy [kombinatorikus optimalizálási](https://hu.wikipedia.org/w/index.php?title=Kombinatorikus_optimaliz%C3%A1l%C3%A1s&action=edit&redlink=1) [probléma](https://hu.wikipedia.org/wiki/Probl%C3%A9ma). Kiváló példa a [bonyolultság-elmélet](https://hu.wikipedia.org/w/index.php?title=Bonyolults%C3%A1g-elm%C3%A9let&action=edit&redlink=1) által [NP-nehéznek](https://hu.wikipedia.org/w/index.php?title=NP-neh%C3%A9z&action=edit&redlink=1) nevezett problémaosztályra. Az utazó ügynök problémájához kapcsolódó [matematikai](https://hu.wikipedia.org/wiki/Matematika) feladatokkal először [Sir William Rowan Hamilton](https://hu.wikipedia.org/wiki/William_Rowan_Hamilton) és [Thomas Penyngton Kirkman](https://hu.wikipedia.org/w/index.php?title=Thomas_Kirkman&action=edit&redlink=1) foglalkoztak az [1800-as években](https://hu.wikipedia.org/wiki/1800-as_%C3%A9vek). Hamilton és Kirkman ezen korai munkájáról szóló értekezés megtalálható a Graph Theory [1736](https://hu.wikipedia.org/wiki/1736)-[1936](https://hu.wikipedia.org/wiki/1936) című munkában.

Adva van n város, illetve az útiköltség bármely két város között, keressük a legolcsóbb utat egy adott városból indulva, amely minden várost pontosan egyszer érint, majd a kiindulási városba ér vissza. (n-1)!/2 út közül kell választanunk, ez ugyanis a [Hamilton-körök](https://hu.wikipedia.org/wiki/Hamilton-k%C3%B6r) száma az n pontú [teljes gráfban](https://hu.wikipedia.org/wiki/Teljes_gr%C3%A1f).

A legkézenfekvőbb megoldás az összes [permutáció](https://hu.wikipedia.org/wiki/Permut%C3%A1ci%C3%B3) végignézése, és a legkisebb súlyú kiválasztása lenne, de tekintve, hogy ez n! darab permutációt jelent (ahol n a városok száma), nagyobb n-ekre ez a megoldás kivitelezhetetlen. Dinamikus programozási technikák segítségével a probléma megoldásának lépésszáma felülről becsülhető n2 · 2n-nel. Ez még exponenciális függvénye n-nek, de sokkal jobb, mint az n! lépést használó brute force („nyers erő”) módszer.

Egyed, egyedek reprezentációja

GA esetében a különböző lehetséges megoldásokat egyedeknek (species) nevezzük. Az algoritmus megvalósítása szempontjából az első lényeges dolog, az egyedek ábrázolása. Leggyakoribb esetek a bitsorozattal, vektorokkal vagy permutációval történő ábrázolás, esetleg ezek valamely kompozíciójából alkotott struktúra. Az egyedeket nagy E-vel jelöljük.

Bitsorozat

Bitsorozattal reprezentáljuk az egyedeket, ha valamely tulajdonsághalmazzal írjuk le azokat, és mindössze arra vagyunk kíváncsiak, hogy az adott tulajdonság jellemzi-e a példányt vagy sem. Előnye alacsony tárigénye, illetve, hogy könnyű számításokat végezni vele.

Vektor

Másik gyakori ábrázolás a (valós) vektoros ábrázolás. Ezt olyan esetekben érdemes használni, ha fontos az egyedek egyes tulajdonságainak mennyiségi értéke is. Vektoros megoldásokkal nagyon sok minden könnyen leírható, és a számítások is jól elvégezhetőek alapvető lineáris algebrai ismeretekkel. Vektorokkal bármely olyan egyedtípus ábrázolható, amelyet fix darabszámú mennyiséggel/költséggel jellemezhető tulajdonság ír le.

Permutáció

Sok esetben egy összesség bejárására kell sorrendet adnunk (gráfalgoritmusok, pl. utazó ügynök probléma), ilyenkor a megoldás az elemek egy permutációja lesz. Az ábrázolás itt is valós vektorral történik, a vektor elemei az alapelemek sorszámai lesznek a felsorolás sorrendjében.



*Egy permutációval ábrázolt kromoszóma. Az egyes biteket a biológiával analóg módon alléleknek is nevezik. Jelen esetben egy 15 városból álló utazó ügynök probléma egyik egyede.*

A klasszikus genetikus algoritmusok bitsorozattal, vagy valós vektorral megvalósított genotípusos ábrázolást használnak, de az egyedi probléma ismeretében lehetséges egy másfajta, hatékonyabb megoldás is. Mivel a véges bitsorozatok tekinthetők azonosnak a {0,1} alaphalmaz feletti vektorterekből vett vektoroknak, továbbiakban csak a valós vektoros ábrázolással foglalkozunk.

Populációk

Az egyedek egy adott időbeli összegségét nevezzük populációnak (population). Egyszerű lenne azt mondani, hogy ez a lehetséges megoldások egy halmaza, de ez nem igaz, mert egy adott tulajdonságokkal rendelkező egyed többszörösen benne lehet. A populációt P-vel jelöljük. Ezen kívül használatos még a populáció méretére a *µ* = |P| jelölés (az elemszám ebben az esetben számításba veszi a többszörös elemeket is).



*Az általam írt algoritmus 4 darab véletlenszerűen generált egyedből alkotja meg a kezdeti populációját.*

Az algoritmus során a könnyebb számolás miatt gyakori az egyedek egyfajta kódolása. Egy ilyen kódolás adja az egyed genotípusát (genotype). Műveleteket az egyed genotípusával végzünk, a folyamat végén ebből dekódoljuk az optimális megoldást. Az egyedeket leíró tulajdonságok megjelenésének összességét fenotípusnak (phenotype) nevezzük. Felfogható úgy, hogy a fenotípus az egyed valóságban vett absztrakt vagy hétköznapi megjelenése, a genotípus ennek egy leképezése valamely az algoritmus során használható adattípusba. Ezt a leképezést kódoló függvénynek, a kimeneti változóit géneknek (gene), azok adott egyednél felvett értékeit alléloknak (allele) nevezzük. Ezek szerint, a kódoló függvény az egyes egyedek tulajdonságait képezi le génekbe.

Jelen esetben szükségtelen a genotípusok alkalmazása, annyira egyszerű a probléma, illetve annak leképzése. Az egyedek a városok sorszámaiból alkotott permutációt tárolják. A távhőrendszer optimálása során már alkalmazható lett volna a genotípusok használata. Adott esetben egy egyed fenotípusát egy olyan lista írja le, ahol az elemek (allélok) a különböző szakaszokhoz tartozó csőméreteket jelentik. Ebben az esetben a vonatkozó genotípus egy olyan lista lehetett volna, ahol az elemek (gének) a csőméretek sorszámát jelzik. Programozási szempontból ez praktikus lett volna, de ezzel együtt is fölöslegesnek éreztem a genotípusok alkalmazását, és a forráskód megírása során kihagytam őket az algoritmusból.

A GA minden egyes iterációs lépése újabb és újabb populációkat állít elő. Az iterációs lépések során keletkező populációkat generációknak (generation) hívjuk. Az algoritmus minden generáció során új populációt állít elő az aktuális felhasználásával. Az új egyedek beépülnek a populációba lecserélve annak néhány, vagy akár az összes tagját. Utóbbi esetben generációs (generational) algoritmusról beszélhetünk. Ha a populáció mindössze egy elemű (*µ* = 1), akkor helybeni (steady state) az algoritmus. Sok esetben szokták az aktuális populáció adott számú, százalékú legjobb példányát automatikusan, változtatás nélkül bepakolni az újba. Az ilyen megvalósítást elitistának (elitist) hívjuk.

Az utazó ügynök probléma során elitista szelekción alapuló populációkat alkotott a program minden generációban. A kezdeti 4 elem mellé különböző módszerekkel képzett további 8-at és a 12 elemből választotta ki a 4 legjobb tulajdonságút, majd vette azt a következő populáció alapjául. Az iterációk végén egyre inkább jellemző, hogy folyamatosan ugyanaz a 4 elem jut tovább a következő generációba.

Fitneszfüggvény

A megoldások értékeit egy rátermettségi- vagy szokásos terminológiával fitneszfüggvény (fitness function) segítségével adjuk meg. Ennek a függvénynek kell heurisztikusan keresni a globális optimumát. Így érvényesül a darwini „legrátermettebb túlélése” elv, mivel minél rátermettebb egy elem, annál jobban megközelíti a fitneszfüggvény értéke a globális szélsőértéket. A fitneszfüggvény megválasztása lehet a legnehezebb feladat, és egyben a legfontosabb is, hiszen ennek segítségével mérjük az egyedek teljesítményét, alkalmasságát.

Kiszámolása az algoritmus szempontjából sok időt vesz igénybe, ezért megválasztása jelentős a futási idő szerint is. Ha kromoszóma n bit hosszúságú, akkor a rátermettségi függvény a keresési téren egy n-dimenziós rátermettségi tájképet (fitness landscape) határoz meg. Valós vektorok esetén ez a függvény egy n-dimenziós folytonos valós függvény lesz. A GA megoldása ekvivalens a rátermettségi tájkép globális szélsőértékének keresésével.

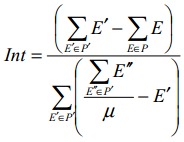
Az utazó ügynök probléma esetén egyértelmű a fitneszfüggvény kérdése: amelyik egyed permutációjához rövidebb körbejárási út tartozik, az lesz a rátermettebb. Ehhez alkottam egy 15x15-ös szimmetrikus mátrixot, melyet feltöltöttem a különböző városok közötti távolságokkal. Véletlenszám generátort használtam, így természetesen a síkban nem lenne kivitelezhető a városok elrendezése (csak egy 14 dimenziós térben). Szerencsére a genetikus algoritmus helyes működése független attól, hogy valóságban a 15 város leképezhető távolságokra helyezkedik-e el egymástól. Röviden: a fitneszfüggvény az egyed első elemét betölti egy „sor” változóba, a másodikat pedig egy „oszlop” változóba. Ezek együtt fogják kiadni a távolság-mátrix megfelelő elemét. Ezután a „sor” változó a második elem lesz, az „oszlop” változó pedig a harmadik. Eközben a távolságokat folyamatosan összeadja a program egy „fitnesz” változóba. Végül az egyed utolsó és első eleme fogja meghatározni az összesített távolsághoz adandó utolsó értéket a mátrixból.

Szelekció

Alapvető fontossággal bír az algoritmus szempontjából, hogy miként jutunk új megoldásokhoz. A meglévő egyedeken végzett műveleteket, amelyek új lehetséges megoldásokat állítanak elő kereső operátoroknak hívjuk. Háromféle operátort különböztetünk meg, melyek mindegyikét, vagy egy részét használjuk az eljárás során. Ezek: a szelekció, rekombináció, és a mutáció.

Az egyedkiválasztás, vagy szelekció (selection) operátor az aktuális populációból alkalmas szülőket (párokat) választ, utódnemzés céljából. Ez azért is fontos, mert a módszer használata során nem kívánunk „légből kapott” adatokkal kisérletezni, hanem szeretnénk az addig elért eredményeket felhasználva tökéletesíteni azokat. A művelet során a szelekciós állományból (selection pool) választunk bizonyos feltételek szerint egyedeket, és helyezzük a szülői állományba (mating pool). A szelekciós állomány kezdetben általában megegyezik a teljes populációval. A szülői állományt addig kell növelni, amíg az új egyedek létrehozásához elegendő szülő nem kerül bele, ez a GA esetében klasszikusan megegyezik a populáció méretével.

Az egymástól különböző szelekciós metódusok működését összehasonlíthatjuk, értékelhetjük többféle tulajdonsággal. A szelekciós intenzitás (selection intensity) a populáció átlagos rátermettségének változását mutatja a szelekció hatására:



Ahol P΄a szelekcióval létrehozott populációt jelöli. Vagyis az intenzitás az új és régi populációk átlag rátermettségének különbsége, osztva az új populáció szórásával.

A változatosság elvesztése (loss of diversity) azon egyedek aránya, amelyek nem kerültek kiválasztásra. Azaz:



Ezen tulajdonságokkal jól vizsgálhatjuk az algoritmusunk szelekciós működését. A szelekciós intenzitás megmutatja, hogy az algoritmus milyen mértékben választja ki a legrátermettebb egyéneket. Ha a változatosság elvesztése túl magas, félő, hogy a keresési tér beszűküléséhez és korai, nem feltétlenül globális optimumhoz való konvergenciához vezet, mivel az új populáció túl sok azonos egyedből áll.

Véletlen kiválasztásos szelekció

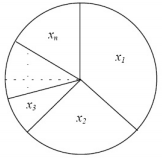
A legegyszerűbb, ámde legkevésbé hatékony szelekció. Gyakorlatilag az aktuális populációból véletlenszerűen választunk szülőket. Lehet ismétléses vagy ismétlés nélküli. Előbbinél a szelekció során minden egyes lépésnél minden egyed azonos eséllyel kiválasztható, míg az utóbbi esetben a már kiválasztott egyedek nem vesznek részt a további kiválasztásban. Legnagyobb hátránya az, hogy nem veszi figyelembe azt a darwini alapelvet, miszerint a rátermettebb egyedek nagyobb eséllyel érvényesülnek az egyedlétrehozásban.

Rátermettség-arányos szelekció

A rátermettség-arányos szelekció (fitness proportionate selection) esetében egy egyed kiválasztásának valószínősége annál magasabb, minél nagyobb a rátermettsége a populáció átlagához viszonyítva. Visszatevéses szelekció, azaz minden lépésnél mindegyik példány (újra) kiválasztható.

Megvalósítása általában a rulett módszerrel történik, ami az egyik legrégebbi, és leginkább használt szelekciós operátor. Az algoritmus analógiája egy rulettkerék: a kerületén felvesszük az egyedeket p(Ê) hosszarányú körcikkekkel. Ahol a „golyó” megáll, az egyed kiválasztásra kerül.

Előnye, hogy könnyen megvalósítható, és figyelembe veszi a szülők rátermettségét. Hátránya, hogy egy nagy rátermettségű egyed aránytalanul sokszor bekerülhet a szülők közé, ezáltal beszűkülhet az algoritmus keresési tere. Ennek elkerülésére szokták a fitnessz függvényt skálázni (scaling). Például, ha a fitnessz függvény exponenciális, akkor egy megfelelő g(x) = c - log(x) függvény lineárissá teheti.



Rulett szelekció szemléltetése kördiagrammal. Az egyes egyedek kiválasztási esélye a körcikkely méretével arányos

Sztochasztikus univerzális mintavétel

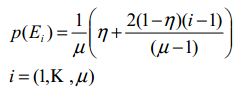
A sztochasztikus univerzális mintavétel (stochastic universal sampling, SUS) fitnessz alapú szelekció. A rulett szelekció olyan módosított változata, amely minimalizálja a művelet során az azonos egyedek többszörös kiválasztását. Itt úgy osztjuk fel a rulettkereket, hogy figyelembe vesszük az egyedek várhatókiválasztásainak számát:



A rulett kerületén *µ* darab mutatót helyezünk, azt egyenlő részekre felosztva. „Pörgetés” után azt az egyedet választjuk ki, amelyik a leközelebbi mutatóhoz esik pörgésirány szerint.

Lineáris rangsorolás

A rangsorolásos szelekció (ranking, linear ranking) szintén a fitnesszérték alapú módszereknél előforduló szórásbeli hátulütőket kívánja kiküszöbölni. A populáció egyedeit sorbarendezzük rátermettség szerint, kezdve a legrosszabbtól a legrátermettebbig. A sorszámozás egyedi, tehát azonos fitnessz értékű egyedek sorszáma eltérő. Az egyes egyedek kiválasztásának valószínősége lineárisan függ azok sorszámától:



Ahol *η* a legrosszabb rátermettségű példány kiválasztási valószínősége (0 ≤ *η* ≤ 1). A legrátermettebb egyed szelekciós esélye 2-*η* lesz, a köztes elemek valószínűségei lineárisan eloszlanak a két szélsőérték között.

Pár-verseny szelekció

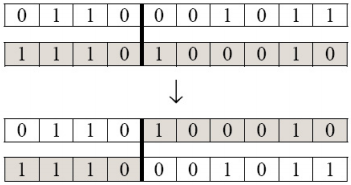
A versengő, vagy pár-verseny szelekció (binary tournament selection) nem fitnessz arányos szelekció. A módszer *µ* lépéses ciklusból áll. Minden lépésben előre rögzített T (tour) darab elemet választunk ki véletlenszerűen a populációból. Majd az így kapott elemek közül a legrátermettebbet választjuk a szülők közé. Pár-versenyről valójában T=2 esetén beszélhetünk, ami a leggyakrabban használt paraméterérték. Előnye, hogy kisebb eséllyel veszít a változatosságból, cserébe általánosan kisebb a szelekciós intenzitás, mint a rátermettség arányos metódusoknál.

Rekombináció

A rekombináció (recombination) operátor két egyed (szülők) reprezentációjából generál új egyedet, egyedeket. Ez a klasszikus biológiai utód létrehozásának felel meg. A GA alapvető kereső operátora a rekombináció, az új egyedek létrehozásában döntő szerepet játszik.

Egypontos keresztezés

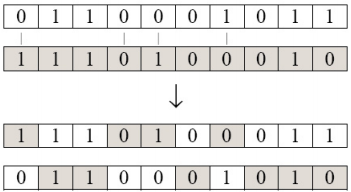
Egypontos keresztezésnél (1-point crossover) az egyedeket egy véletlenszerűen kiválasztott i. bitnél (1 ≤ i < n) elvágjuk. Az új egyed az egyik szülő tulajdonságait 1,...,i bitig, a másik szülő tulajdonságait (i+1),...,n-ig örökli. Lehetséges két új egyed (testvérek) létrehozása is: a második keletkező egyed, analóg módon a szülők kromoszómáinak másik feléből áll össze, ahogy azt az alábbi ábra is mutatja.



*Egypontos keresztezés. A keresztezési pont (crossover point) a 4. génnél van.*

Egyenletes keresztezés

Egyenletes keresztezés (uniform crossover, UX) használatával az utód egyes génjei a szülők azonos génjeinek valamelyike lesz. Az i. utódgén 0,5 eséllyel az egyik vagy a másik szülő azonos génje lesz. Ez alapján átlagosan a gének fele cserélődik ki a szülők között.



*Egyenletes keresztezés. A két utód 4-4 génben tér el a szülőktől.*

Formálisan leírva:



Ahol  az utód, ,  a szülők megfelelő génjeit jelöli, *a* pedig minden génre véletlenszerűen választott együttható {0,1} halmazból.

Köztes keresztezés

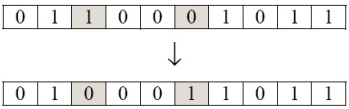
A köztes keresztezés (intermediate recombination) az egyenletes keresztezés kissé módosított változata. A különbség annyiban rejlik, hogy az előbb megadott formulában *a* értéke [-h, 1+h] intervallumban mozoghat (h ≥ 0). Tehát az utód génjeinek értéke szüleik génjeinek egy függvénye lesz, ami azt jelenti, hogy nem közvetlenül a szülői tulajdonságokat örökli.

Mutáció

A mutáció (mutation) operátor meglévő egyedeket módosít úgy, hogy véletlenszerűen megváltoztatja azok bizonyos tulajdonságait. A mutáció segít szélesíteni a keresési teret, viszont óvatosan szabad csak alkalmazni, mert elronthatja a gondosan kiválasztott szülői tulajdonságokat. Éppen ezért a mutáció jelentősége a GA folyamán csekély, egy tulajdonság megváltoztatásának esélye általában  ≤ . Ezt úgy kell érteni, hogy a mutáció valószínűsége  minden egyed, minden génjére. Az alábbiakban összefoglaljuk az általánosan használt mutáció megvalósításokat.

Véletlen génenkénti mutáció

A mutáció legegyszerűbb és legkézenfekvőbb formája, hogy az egyes géneket bizonyos valószínűséggel megváltoztatjuk. Bitsorozat esetén szimplán negáljuk a megfelelő értéket, valós vektoroknál az eredeti értéket egy véletlen értékkel helyettesítjük, ügyelve természetesen arra, hogy az új érték értelmezhető legyen az adott pozícíóban. Egy másik verzió szerint találomra veszünk n ·  darab gént, és azokat változtatjuk meg.



*Véletlen bitenkénti mutáció. A mővelet során az egyed 2 génje negálódik.*

Véletlen elemi permutáció

Egy másik könnyen megvalósítható mutációs algoritmus a véletlen elemi permutáció. Igazából akkor van jelentősége, ha az egyedek permutációkkal vannak ábrázolva, hiszen az esetben eme tulajdonságot nem szabad a mutáció során sem elrontani. Ilyenkor az aktuális permutációt megszorozhatjuk (i j) transzpozícióval (1 ≤ i, j ≤ n), aminek eredménye egy olyan permutáció lesz, amiben az i. és j. érték helyet cserél. A mutáció valószínűségét egyedenként vizsgáljuk, egy kiválasztott egyednél akár több elemi permutációt is végrehajthatunk.

# A feladat kibontása/Első főfeladat

## Első részfeladat

# Második főfeladat

## Első részfeladat

Főfeladatból lehetőleg háromnál több ne legyen!

## Második részfeladat

## N-edik részfeladat

# Összefoglalás/Eredmények értékelése

## Eredmények

Az összefoglaló értékelés a három oldalt lehetőleg ne haladja meg! Az elvégzett munka és eredményeinek bemutatása egyes szám első személyben fogalmazva.

## Javaslatok/Következtetések/Tanulságok

A feladat elkészítése során levont tanulságok összefoglalása. Javaslattétel, továbbfejlesztési lehetősége bemutatása, előretekintés a jövőbe stb.

# Felhasznált források

1. Szerző(k) Neve (1999): *Írásművének címe*. Kiadó, Kiadó székhelye. Egyéb azonosítók
2. http://oldweb.reak.bme.hu/

# Summary

Az elvégzett munka rövid, másfél oldalt meg nem haladó, de legalább 2/3 oldalnyi terjedelmű angol nyelvű összefoglalása.

Keywords: *keyword1, keyword2, keyword3*