Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Szörfi Jázmin

Napi szélfarm-karbantartási és -javítási műveletek ütemezése és optimalizálása

KONZULENSEK
Dr. Kolumbán Sándor
Benedek Zoltán
Dr. Juhász Sándor

TDK Dolgozat
2018

1. Tartalomjegyzék

2. Absztrakt

Napjaink megújuló energiamixében jelentős szerep jut a tengeri szélerőműparkoknak. Ezek üzemeltetése, karbantartása és javítása költséges művelet, amit hajókon szállított technikuscsapatok végeznek. A napi menedzsment feladatok ellátásának segítésére már léteznek számítógépes eszközök, melyek a feladatok adott sorrendű elvégzését tudják szimulálni, és ezzel azok költségét megbecsülni. Az előre meghatározott végrehajtási sorrend azonban nem biztos, hogy optimális, komplexebb napi feladatok esetén az operátorok messze nem optimális feladatsorrendet is specifikálhatnak. Munkám elsődleges célja az volt, hogy ezt a manuális ütemezési folyamatot optimalizáló modullal támogassam, ami a paraméterek ismeretében automatizáltan állít elő minél kisebb költségű ütemtervezetet a feladatok elvégzéséhez.

Az általam készített modul matematikai optimalizálási feladatokat fogalmaz meg az ütemtervezet közelítésére, ezek megoldásaiból készítek munkamenet-javaslatokat. A probléma feldolgozása a modell határainak meghúzásánál kezdődik, hiszen a teljes üzemeltetési munkafolyamat túl sok paramétertől függ.

Már maga az időjárás számos dolgot befolyásolhat: a szélerősség a termelt energiát, a hajók sebességéét, a tengerszint magassága a különböző műveletek biztonságosságát. A meghibásodások javítását különböző képesítésű csapatok végezhetik, adott szállítmányhoz megfelelő típusú hajóval, feladatfüggő végrehajtási idővel

Először számba vettem a lehetséges paramétereket, majd egy kellően egyszerű, de még az eredeti feladat főbb jellemvonásaival rendelkező feladatot fogalmaztam meg. Innen folytattam tovább a modellezést iterációnként újabb paraméterek hozzáadásával, a teljes feladat minél több vonatkozását megragadva.

Az optimalizálási feladatok megoldásait különböző Matlab toolboxokat használva kerestem, melyek különböző sikerrel teljesítettek. A modelleket kétféleképpen parametrizáltam, az egyik változatban nem lineáris egyenlőtlenségekkel dolgoztam, melynél a megoldások mindig értelmesek voltak, de szűk volt a keresési tér, így a toolboxok hajlamosak voltak lokális minimumban terminálni a globális szélsőérték helyett.

A másik megoldásban csak lineáris egyenlőtlenségek voltak, lazább feltételekkel, így a numerikus optimalizáló eljárások könnyen tudtak kis módosításokkal a jó megoldás felé tendálni. Az elvárások nem explicit módon a kényszerekben voltak megfogalmazva, a célfüggvényben jelentek meg oly módon, hogy az optimális megoldás garantáltan teljesítse a követelményeket.

A különböző algoritmusokat kiértékeltem és összehasonlítottam egymással és egy heurisztikán alapuló mohó algoritmussal. Az algoritmusok által javasolt ütemterveket a szimulátorral validáltam, összevetve az egyszerűsített modellek által becsült költségeket a komplexebb, több paraméterrel operáló szimulátor számításaival. A szimulátorral való integrációt .NET környezetben, C# nyelven valósítottam meg.

3. Részletes problémaleírás

3.1 A feladat felmerülése

Napjaink egy fontos kérdése a növekvő energiaigény kiszolgálása, melyben egyre nagyobb szerepet kap a környezettudatos gondolkodásmód. A megújuló energiaforrások egyre nagyobb arányban jelennek meg a hagyományos energiaforrások mellett[1], közülük is a legjelentősebb – a vízenergia után – a szélenergia, mely 2011 és 2017 között több, mint megnégyszereződött [2]. Léteznek onshore, azaz szárazföldre telepített szélturbinák, és offshore, azaz nem partra telepítettek. Az utóbbiak kevésbé elterjedtek Európában, de jelentőségük egyre nő [3]. Az offshore erőművek nagyobb teljesítményre képesek, a nagyobb vízfeletti szélerősségből következően és nagyobb méretük miatt is, de karbantartásuk és javításuk megszervezése bonyolultabb.

Több erőmű meghibásodása esetén figyelembe kell venni a kikötőtől és egymástól vett távolságukat, az egyes meghibásodások súlyosságát, az értintett erőművek teljesítményét, hogy megalkothassuk azt a javítási tervet, mely minimális költséggel jár. Egy ilyen terv nem csak az erőművek megjavításának sorrendjét tartalmazhatja; ha a javító társaságnak rendelkezésre áll több hajója és több technikuscsapata, akik alkalmasak a munkálatok elvégzésére, akkor meg kell adni, hogy mely csapatok mely hajókkal utaznak, és mely hajók melyik erőművekhez fognak elhajózni. A meghibásodások különbözőek lehetnek (például egész lapát cseréje szükséges, vagy csak meg kell húzni egy csavart); ezek fajtáitól függően különböző méretű és felszereltségű hajókra és különböző képesítésű szakemberekere lehet szükség.

A szélerősség befolyásolja az erőművek termelését[4], a hajók haladási sebességét, a munkálatok elvégzésének biztonságosságát. Az utóbbi kettőt a tengerszint magassága is befolyásolhatja. Ezek a hatások szintén nem hagyhatók figyelmen kívül egy javítási ütemterv elkészítésénél.

A fent felsoroltakon kívül lehetségesek karbantartási műveletek, melyeket bizonyos időközönként javasolt és/vagy kötelező elvégezni, de a periódusidő letelte előtt szabad kezet kapnak az ütemezők a munkák beütemezésére. Ebből több módon is lehet költséget minimalizálni. Összevárhatjuk az egy periódusba eső erőműveket, hogy ne kelljen többszörös hajózási költséget fizetni. Az is lehetséges, hogy a karbantartási munkálatok elvégzése befolyásolja egy meghibásodás fellépésének esélyét, így megfelelően sűrű időközt kell választani, hogy csökkentsük egy súlyos hiba fellépésének esélyét/valószínűségét, de megfelelően ritkát is, hogy ne legyen többletköltség a gyakori javítás, holott lehetséges, hogy fel se lépne meghibásodás.

Látható, hogy egy ilyen terv elkészítése nem egyszerű feladat, rengeteg paraméter van hatással egymásra és a végső költségre. Papír-ceruza módszerrel nehéz és hosszadalmas egy ilyen feladat megoldása, ezért íratott egy egy erőművekkel foglalkozó cég egy szimulátor[15], melyben az operátor megadhat egy tervet. A tervben különböző hajókhoz különböző csapatokat lehet rendelni, és minden hajónak saját listája van a meglátogatandó erőművekről.

A szimulátor számol időjárással; megadhatunk ebédszünetet a munkásoknak; ha egy csapat végzett, a hajót automatikusan elindítja feléjük, és felveszi őket. Adhatunk rakodási időt; szabályozhatjuk, hogy a kikötőből mikor lehet be- és kihajózni, ha például a zsiliprendszer megköveteli ezen időpontok diszjunkt létét. A szimuláció elvégzi mindezen paraméterekkel a számításokat a felhasználó helyett, így annak csak különböző terveket kell kigondolnia, majd összehasonlítania őket, akár költségtényezőnként egyenként.

Ez már önmagában nagy segítség egy terv elkészítésében, de még hasznosabb lenne, ha a tervkészítés is automatikus lenne, vagy az operátor segítségére automatikusan fel lehetne ajánlani tervjavaslatokat, melyeket aztán belátása szerint módosíthat vagy eldobhat. Ennek megvalósulásának elősegítése volt dolgozatom célja.

3.2 Feladat elhelyezése

A 4. fejezetben felsorolt paraméterekhez hasonló szerkezetű optimalizáló keretrendszert készítettek 2009-ben [10]. A projektben a hangsúlyt a megelőző karbantartási munkálatok ütemezésére tették, melyet hosszú és rövid távon is lehetett tervezni. A megoldásban lineáris egészértékű optimalizást használtak, az algoritmust egy a repülőgépiparban használt opportunista karbantartási optimalizáló modell inspirálta. Azonban az időjárás hatásaival nem számoltak, jövőbeni munkálatként jelölték meg egy olyan modell felállítást, amivel már a valós világban is lehet számolni.

A feladat komplexitását az is jól mutatja, hogy 2011-ben született egy cikk [11], mely csak egyetlenegy turbinával foglalkozik, melynek csak egyetlenegy modulja van. Így nyilvánvalóan nem sorrendiségi optimalizálás történik, hanem valószínűségi modell és döntési fa építésének segítségével végeznek számításokat a megfelelő megelőző karbantartások elvégzésének időpontjaira, hogy minimalizálják a hibák előfordulását.

Megelőző karbantartások ütemezésének egy továbbfejlesztett 2016-os modelljében [12] már több turbina több moduljával is számolnak, bevezetve az öregedést a modulokra, mely növeli a meghibásodás esélyét; javítással ez a valószínűség csökken, míg cserével 0-ra csökken adott idő elteléséig.

A fenti munkák mind megelőző, tervezett munkálatok ütemezésével, illetve azoknak a tényleges meghibásodásokra gyakorolt hatásaival foglalkoznak, míg én a ténylegesen fellépő hibák ütemezését helyeztem középpontba.

3.3 Megoldás menete

Kezdetnek összegyűjtöttem a lehetséges paraméterek egy nagy halmazát, majd rendeztem őket aszerint, hogy melyik objektumnál léphetnek fel; melyek konstansok, melyek nem; hogyan hatnak egymásra. Így kaptam egy átfogó képet a problémáról. Ezen listát adom meg a 4. fejezetben.

Ezután áttekintettem, milyen különböző módokon lehet vizsgálni a problémát, ha csak bizonyos típusú paramétereket engedünk meg. Redukáltam a problémát a lehető legtöbb paraméter elhagyásával, de még megoldásra érdemes problémaszint eléréséig. Így kaptam egy olyan problémát, ahol minden erőmű azonosan termel, csak a távolságaik különbözőek, egyetlen hajónk és csapatunk van a bejárásra. Ezzel a leegyszerűsített problémával foglalkozom a továbbiakban. Elővettem a legalapvetőbb heurisztikát, azaz mindig a legközelebb lévő erőműhöz küldjük a csapatot. Ennek az optimálisságára találtam ellenpéldát, így ennek módosítását, más heurisztikák és algoritmusok keresését kezdtem el. Ezt a gondolatmenetet írom le a 5. fejezetben.

Saját algoritmus fejlesztése helyett abba az irányba indultam el, hogy már létező, általános célú optimalizáló programokat fogok felhasználni. Ezek használatához a természetes nyelven megadott problémákból modelleket kellett alkotnom, melyeket már az algoritmusok is értenek és képesek megoldani. Egy bonyolultabb modellnél a nagyszámú paraméter és egymásra hatás miatt nehéz megalkotni a jó modellt, ezért a legegyszerűbb esetből kiindulva megalkottam több egyszerűbb problémaesetet. Ezek modelljeit elkészítettem, így egy bonyolultabb modellkészítéshez csak össze kell válogatni a megfelelő alapeseteket, és a hozzájuk tartozó függőségeket kell behúzni az új modellbe. Az 6. fejezetben adom meg a készített modelleket.

A készített modellek általánosak, használatukhoz át kell írni őket a használt optimalizáló eszköz nyelvére. Több eszközt is kipróbáltam, ezeket felhasználói szempontból hasonlítom össze a 7. fejezetben; mivel fekete dobozként használtam őket, nem térek ki belső felépítésükre.

Egy adott modell lefordítása egy eszközre nemcsak a szintaktika miatt más (esetemben a szintaktika nem is más, mivel mindegyik általam kipróbált eszköz használható MATLAB környezetben), hanem mert különböző típusú kényszereket ismernek, például lineáris és nem lineáris kényszereket. Egy adott

eszközre is többféleképpen meg lehet fogalmazni ugyanazt a problémát, ezt meg is tettem a beépített MATLAB Genetic Algorithm esetén. Az alkalmazott modellátírások különbségeiről és hasonlóságairól írok a 8. fejezetben.

Ezek után a kapott eredményeket kiértékeltem, összehasonlítva a különböző eszközök és leírások képességeit. Mivel az optimális megoldás nem ismert, lévén hogy pont annak a megtalálása a cél, referenciaként egy heurisztikán alapuló algoritmussal is lefuttattam a teszteket, ezek eredményeit tartalmazza a 9. fejezet.

A fentiekkel kapcsolatos eredményeket és tapasztalatokat, és mivel a dolgozat keretein belül csak a probléma felszínét lehetett érinteni, a lehetséges folytatásokat írom le a 10. fejezetben.

4. Lehetséges paraméterek

A problémában részt vevő objektumokat sorolom fel, különválasztva konstans változóikat és számított változóikat, melyeknek kiszámítási módját meg is adom a többi paraméter függvényében. A paraméterek könnyebb és tisztább értelmezéséhez és viszonyaiknak leírásához szükségszerűek mértékegységek.

Mivel az itt felsorolt paramétereknek csak egy részhalmazával foglalkozom a további fejezetkben, az itt leírt precíz mértékegységeknek csak egy részét fogom használni, többet leegyszerűsítek (például a modellekben nem számolok benzinárfolyammal, sem -fogyasztással, a hajó költségét EUR/km-ben rögzítem). A pontos mértékegységeket egy jövőbeni több paraméterre kiterjedő modell esetén lehet hasznosítani.

Szintén abból következve, hogy nem használtam fel minden paramétert, vannak nem véglegesítettek, melyeket több lehetséges értelmezési módon is megadtam (például munkás bére), jövőbeni felhasználás és algoritmus függvényében le lehet rögzíteni őket.

Világ

Ezek a paramétek egyetlen modellezett objektumhoz sem csatlakoznak, úgymond külső globális változók.

Energiaárfolyam $\left[\frac{EUR}{kW}\right]$: a világban egy kW óra ára euróban, hogy az erőművek teljesítményveszteségét pénzben tudjuk kifejezni; így összehasonlíthatóvá válik a munkások bérével, a hajók bérleti és benzinköltségével, mivel azonos mértékegységben lesznek mérve.

Biztonságos szélerősség $\left[\frac{km}{h}\right]$: ha e fölötti az aktuális szélerősség, nem dolgoznak a technikusok, a hajók lehorgonyoznak, egész addig, amíg ez alá nem csökken az aktuális szélerősség.

Aktuális idő [ÉÉÉÉ/HH/NN/ÓÓ/PP]: a világidő, többnapos ütemterv készítéséhez, vagy egy napon belül a munkaidő végéig hátralévő részének mérésére.

Üzemanyag ár $\left[\frac{EUR}{l}\right]$: a világban egy liter üzemanyag ára, hogy a hajók fogyasztását a közös mértékegységre, azaz euróra hozzuk a többi költséggel.

Kikötő

Innen indulnak el a hajók minden nap a technikusokat szállítva az erőművekhez, és ide is kell visszatérniük.

x, y koordináták [1 egység 1 m] : a kikötő elhelyezkedése.

Szélerőmű

A farmhoz tartozó szélerőművek, ezeket kell karbantartani.

Meghibásodások: az erőműhöz tartozó meghibásodások azonosítói.

Minszélerősség $\left[\frac{km}{h}\right]$: ez alatt nem forognak a lapátok, nem termel energiát az erőmű, ezt a szélerősséget elérve viszont bekapcsolódnak a turbinák.

Maxszélerősség $\left[\frac{km}{h}\right]$: ezen szélerősség felett veszélyes lenne működtetni a turbinákat, mivel káreset történhetne, ezért ezt a szélerősséget elérve lekapcsoljuk őket.

Prefszélerősség $\left[\frac{km}{h}\right]$: ezen a szélerősségen érik el az erőművek maximális teljesítményüket, nagyobb szél esetén sem tudnak több energiát termelni.

Maxtermelés$\left[\frac{kW}{h}\right]$: ennyit termel az erőmű a preferált szélerősségen.

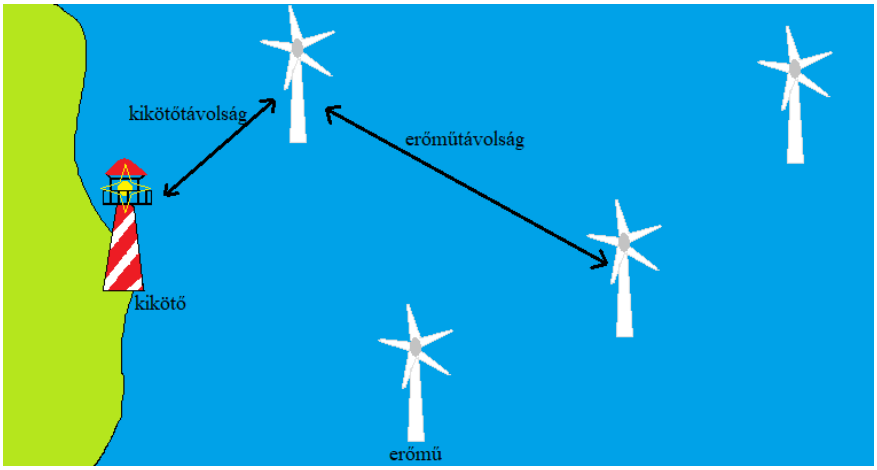
x, y koordináták $[1\ egység\ 1\ m]$: az erőmű koordinátái.

Javítás alatt $[igen/nem]$: javítás alatt kikapcsolt állapotban vannak az erőművek, így nem termelnek energiát.

Számított értékek:

Kikőtőtávolság $[m]$: egy vízi útvonal a kikötő és erőmű között, az egyszerűség kedvéért kikötő távolsága az erőműtől légvonalban; feltételezzük, hogy nincs akadály a kettő között, és nincsenek kijelölve követendő vízi sávok, melyekben közlekedni kell.

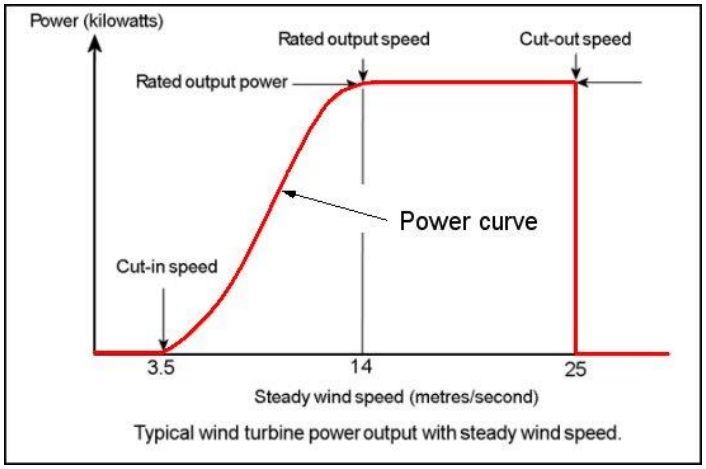
Erőműtávolság $[m]$: másik erőműtől vett légvonalbeli távolság



1. ábra: Kikötőtávolság és erőműtávolság légvonalban mérve

Termelés (t) $[kW]$: az erőmű pillanatnyi termelése az idő függvényében. A közvetlen időfüggés teszi közvetve az időjárásnak és aktuális állapotának függvényévé a termelését.

Ha adott időpillanatban javítás alatt áll, akkor 0 a pillanatnyi termelése. Ha szélerősség(t) < minszélerősség(t), akkor is 0, hiszen még nem kapcsolódtak be a turbinák. Ha minszélerősség < szélerősség(t) < prefszélerősség, akkor már elindult a termelés, de még nem maximumteljesítményen vagyunk.



2. ábra: Szélturbina termelése a szélerősség függvényében

A fenti görbén látható köztes részt lineárisra egyszerűsítettem.

$$\frac{szélerősség\ (t) - minszélerősség}{prefszélerősség - minszélerősség} \cdot maxtermelés \cdot \Delta t - tényleges\ kiesés \cdot \Delta t$$

A $\Delta t[h]$ egy kellően kis időintervallum, amire kiértékeljük az aktuális termelést, órában mérendő, hogy a $maxtermelés$-sel beszorozva helyes mértékegységet kapjunk.

A tényleges kiesés$\left[\frac{kW}{h}\right]$ a meghibásodás jellemzője, ami az időmúlásával súlyosbodhat, és teljesítményvesztést okoz a turbinánál. Jelen modellben időjárásfüggetlen a hiba súlyossága.

Ha prefszélerősség $<$ szélerősség(t) $<$ maxszélerősség, akkor maximálisan termel az erőmű, csak a meghibásodás vehet le ebből:

$$maxtermelés \cdot \Delta t - tényleges\ kiesés\ \cdot \Delta t$$

Ha maxszélerősség $<$ szélerősség(t), akkor 0 a termelés, biztonsági okokból ki vannak kapcsolva az erőművek.

Meghibásodás

Kialakulás ideje [ÉÉÉÉ/HH/NN/ÓÓ/PP]: a hiba kialakulásának kezdte.

Időtartam $[h]$: a megjavításhoz szükséges idő, ez a hiba súlyosbodásával nőhet.

Súlyosbodás $\left[\frac{1}{h}\right]$: megadja, hogy az idő múlásával súlyosbodik-e a hiba, és ha igen, menyivel. Egynél nagyobb vagy egyenlő, mivel különben enyhülne a hiba.

Kiesés$\left[\frac{kW}{h}\right]$: a hiba által okozott alapvető kiesés, a kialakulásának pillanatában, ez az érték súlyosbodhat az idő múlásával.

Szükséges képességek a megjavításhoz: a hiba fajtájából adódóan különböző képesítésű technikusokra lehet szükség.

Számított értékek:

Elvégzett $[h]$: ennyi órányi javítási munkálatot már elvégeztek rajta. Ha $szélerősség(t) > biztonságos\ szélerősség$, akkor 0-val nő az érték, egyébként a munkások által ott töltött idő.

Tényleges időtartam $[h]$: a javítás elvégzéshez szükséges idő, amibe már a hiba súlyosbodása is beleszámít, a következőképpen: $időtartam \cdot súlyosbodás^{aktuális\ idő\ -\ kialakulás\ idje}$

Hátralévő $[h]$: a még hátralévő igényelt javítási idő, ami a következőképp számolandó: $tényleges\ időtartam\ - elvégzett$

Tényleges kiesés(t) $\left[\frac{kW}{h}\right]$: a meghibásodás által okozott tényleges energiatermelési kiesés. azaz a súlyosbodással számolt kiesés: $kiesés \cdot súlyosbodás^{aktuális\ idő\ -\ kialakulás\ idje}$

Hajó

Maxhullámmagasság $[m]$: ennél nagyobb hullámok esetén nem közlekedik biztonsági okokból.

Maxsebesség $\left[\frac{km}{h}\right]$: az elérhető maximális sebessége a hajónak.

Széllassulás [%]: ezzel az értékkel súlyozzuk az aktuális szélerősséget, majd levonjuk a sebességéből, ezzel szemléltetve a hajó szél okozta sebességvesztését.

Hullámlassulás[%]: ezzel az értékkel súlyozzuk az aktuális hullámmagasságot, majd levonjuk a sebességéből, ezzel szemléltetve a hajó hullámmagasság okozta sebességvesztését.

Technikuscsapatok: a hajón utazó technikusokból álló csapatok.

Benzinfogyasztás $\left[\frac{l}{100\ km}\right]$: a hajó 100 kilométerenkénti benzinfogyasztása literben.

Használati költség$\left[\frac{EUR}{nap}\right]$: a hajó bérleti díja egy napra, euróban számolva.

Számított értékek:

Tényleges sebesség(t) $\left[\frac{km}{h}\right]$: a hajó tényleges sebessége, beleszámítva az időjárás miatti veszteséget. Számítás: $maxsebesség - (széllasulás \cdot szélerősség(t) + hullámlassulás \cdot hullámmagasság(t))$, ha ez az érték kisebb vagy egyenlő 0, akkor értelem szerint nem mennek a hajók (nincs értelmezve tolatás). Ha $szélerősség(t) > biztonságos\ szélerősség$, akkor is 0 a sebesség, a hajók biztonsági okokból lehorgonyoznak.

Megtett távolság $[km]$: a hajó által megtett összes távolság km-ben mérve.

Tényleges fogyasztás $\left[\frac{l}{100\ km}\right]$: a hajó benzinfogyasztása literben az egész karbantartási procedúra alatt, azaz $\frac{megtett\ távolság}{100} \cdot benzinfogyasztás$.

Technikus

Aki képes megjavítani a meghibásodásokat; csapatokba szervezve működnek, hajókhoz vannak rendelve, amelyek szállítják őket a javítási helyszínhez.

Képességek: a javítási képességek, melyeknek birtokában van az adott technikus.

Munkaidő kezdte $[ÓÓ/PP]$

Munkaidő vége $[ÓÓ/PP]$ Többféleképp is lehet értelmezni: eddigre már a kikötőbe kell érniük a technikusoknak; vagy ekkor hagyják abba a munkát, és visszavisszük őket a kikötőbe; vagy nem visszük őket vissza, a hajón éjszakáznak, és másnap a munkaidő kezdetével folytatják a munkát. (A szimulátor az utolsót alkalmazza.)

Napi bér $\left[\frac{EUR}{nap}\right]$ Lehetne más modellt is alkalmazni a bérezésre, például órabér, különszámolt hajózási bér és javítási munkabér, akár túlórapénz felszámítása (bár valószínűleg minden túlórabér megfizetése jobban megérné, mint még egy napnyi energiatermelés-kiesés bevállalása).

Technikuscsoport

Ez a kapcsolat adja meg, hogy melyik hibát mely technikusok fogják javítani.

Technikusok: akik egy csapatba tartoznak.

Meghibásodások: melyeket az adott csapatnak kell megjavítania.

Időjárás

Az úgymond tényleges értékek, melyekkel a szimuláció fog futni.

Szélerősség(t)$\left[\frac{km}{h}\right]$: az aktuális szélerősség.

Hullámmagasság(t)$[m]$: az aktuális hullámmagasság.

Időjárás-előrejelzés

Ez az az időjárás, amivel az optimalizáló dolgozik, a szimulátor más időjárással futtatandó. Megközelítéstől függően lehet ez egy évekkel ezelőtti időjárás-előrejelzés, a futás során pedig az ahhoz tartozó valós időjárás, lehet ez egy valószínűség, míg a szimulátor időjárása ezen valószínűségekkel generált véletlen érték.

Jelzett szélerősség(t) $\left[\frac{km}{h}\right]$

Jelzett hullámmagasság(t))$[m]$

Bevétel egy napra

$$\sum_{erőmű} \sum_{t} termelés(t) \cdot energiaárfolyam$$

Költség egy napra

$$\sum_{technikus} napi\ bér$$
$$\sum_{hajó} használati\ költség$$
$$\sum_{hajó} tényleges\ fogyasztás$$

5. Lehetséges megközelítések

Miután sorba vettem a lehetséges paramétereket, megoldási módot kellett találnom. Az alábbiakban megemlítek több lehetséges megközelítési módot, mellyel modellezni lehet a problémát. Én ezek közül a legegyszerűbbeket valósítottam meg.

Optimalizálni kétféleképpen lehet: megalkotjuk a saját algoritmusunkat, vagy felhasználjuk másokét. Az első megoldás jobb eredményre vezethet, mivel maga az algoritmus is feladatspecifikus lesz, míg a másodikban általános megoldókat tudunk alkalmazni csak. Elsőnek megtaláltam a legegyszerűbb heurisztika hibáját, majd megpróbáltam általam ismert algoritmusokkal párhuzamot vonva algoritmusötleteket alkotni, ezeket mutatom be az alábbiakban.

5.1 Valószínűségi modellel

Meghibásodás

Vannak karbantartási munkák, amelyeket adott időközönként el kell végezni, mivel hibákat okozhatnak. Ha hamarabb végezzük el őket, kisebb valószínűséggel okoznak hibát; ha később, nagyobbal.

Teszteléséhez készíteni kell mellé egy algoritmust, ami csak akkor indul el javítani egy erőművet, ha a pontos időköz lejár, vagy ha elromlik az erőmű. Ha az első algoritmus túlbuzgó lenne, megvédene a meghibásodásoktól, de cserébe annyiszor kéne javítani „felesleges óvatosságból”, hogy már az is jobb lenne, ha ritkább javítások mellett néha el is romolna az erőmű, akkor a második algoritmussal lebuktathatjuk.

Az időjárást is lehet valószínűséggel modellezni. Felállítunk egy előrejelzést, ami eloszlásokkal adja meg a várható hullámmagasságot és szélerősséget, ezzel számol az algoritmus. Az algoritmus értékelésekor a szimulátorba az eloszlásoknak megfelelő véletlen számokat generálunk, így modellezve, hogy a valós időjárás sem pontosan felel meg a meteorológiai előrejelzéseknek.

Az eddig elkészült modelljeim nem tartalmaznak sem időjárást, sem súlyosbodó meghibásodásokat, így ezen problémák tárgyalását a későbbiekben mellőzőm.

5.2 Offline / Online

Offline

Adott egy kigenerált hibahalmaz, arra optimalizál az algoritmus, ha közben újak jönnek, azokat csak a teljes lefutás után fogja figyelembe venni, teljesen új tervet készít hozzájuk.

Karbantartási munkákat is ütemezhetünk így, ekkor, ha hamarabb elromlik is egy turbina: vagy csak a kijelölt időben megyünk ki, vagy bekerül az ütemezési sorba, és a legközelebbi lefuttatáskor ő is bekerül.

Online

Javítási munkálatok megkezdése után is jelentkezhet új meghibásodás; de az algoritmus kész új ütemtervet készíteni, beilleszti egy jónak ítélt helyre az újabb meghibásodást, nem másítva meg nagyon ezzel a már elkészült és futtatott tervet. Ez utóbbi rosszabb megoldást ad, de kisebb számítási költségű, mint a fenti.

Az offline megoldást alkalmaztam, karbantartási munkálatokat nem modellezve, csak meghibásodásokat.

5.3 Heurisztika halála

Először redukáltam a problémát egészen odáig, amíg csak a távolságok számítanak szinte, és vettem a *legközelebbi erőmű elsőnek* heurisztikát, majd kerestem ellenpéldát, hogy bemutassam, ez miért nem optimális.

Az eset leírása

Van k darab erőmű, melyek termelése azonos, meghibásodásaik is azonosak, az időjárás is konstans. Egyetlen hajónk és technikuscsapatunk van, tehát nekik kell sorban végiglátogatniuk az összes erőművet. Költség csak a kiesett energiaveszteségből van, tehát se a technikusnak nincs bére, se a hajónak költsége. A nap végtelen, azaz megállás nélkül folyamatosan tudnak dolgozni a munkások.

Megoldandó: milyen sorrendben járja be a hajó az egy emberrel az erőműveket, hogy minimális legyen a kiesés. Mivel egyformák, az a lényeg, hogy minél hamarabb, minél több elkészüljön. Mivel állandó az időjárás, az erőművek kikötőtől és más erőművektől való távolságát elég időben mérni.

Mivel állandó a hajó sebessége, ezért a távolságot lehet km helyett időben mérni, így egyből megkapjuk az időt, ameddig az egyes erőművek a javításukra várnak, és mivel azonos a termelésük, elég a leállásukat időben mérni.

A megoldandó probléma

Mivel a hajó, munkás ingyen van, a visszaidővel se kell számolni, elég egy út, amivel bejárjuk az összes erőművet. Veszteségünk nem az út megtételéből van, hanem a kiesett termelésből, azt pedig úgy kaphatjuk meg, ha az erőműkiesését megszorozzuk a meghibásodás fennállásával. Mivel jelenleg minden erőmű azonosan termel, nem kell megkülönböztetnünk egymástól a leállásokat, az összes leállva töltött időt kell minimalizálni. Ez az érték nem fog megegyezni a javítási munkálatok össz igényelt idejével. A meglévő hibák számát súlyozni kell a felmerülésük hosszával, hisz veszteségünk abból van, ha egy erőmű áll.

$\text{összes javítási idő}: n$
$\text{erőművek száma}: k$
$\text{erőművek új termelésbe állásának ideje}: x_1, x_2 \dots x_k \qquad \forall\, 1 \le i \le k \text{ esetén } x_i \le n$

$$\text{minimalizálandó}: \sum_{i=1}^{k} x_i$$

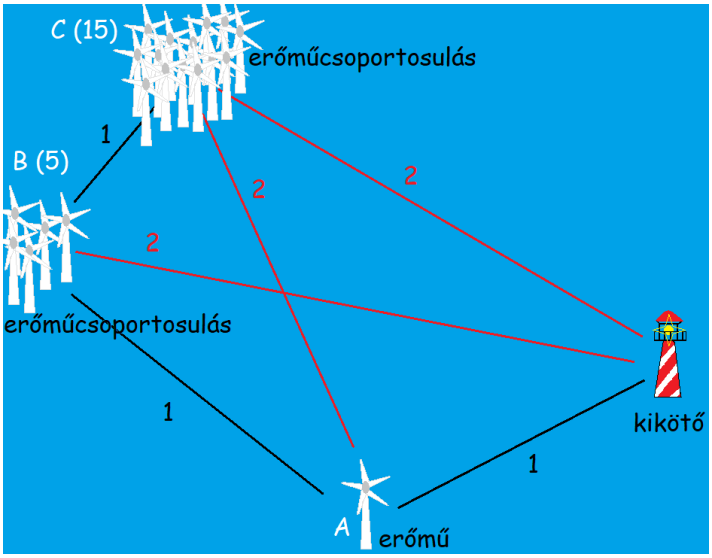
A javítási idő miatt minden esetben egy c konstanssal késik a kész állapot a hajó odaérkezéséhez képest, egy konstans $k \cdot c$ –vel növekszik a nem működési idő minden esetben, ezt nem befolyásolja a választott sorrend, így elhanyagolható.

Ellenpélda minimális összidő helyességére:

Gráfokat használok, ahol csúcsok: erőművek, erőműcsoportosulások (a tagok között lévő távolság elhanyagolható az erőműcsoportok/erőművek közötti távolságokhoz képest, a megjavításukra eső idő ugyanúgy konstans, mint egy erőműnek, csak nagyobb), kikötő.

Az élek jelentik az útvonalakat, minden csúcs között van útvonal, nincsenek vízi akadályok.

Élsúlyok: a távolság időben, a súlyok teljesítik a háromszög-egyenlőtlenséget.



3. ábra: Ellenpélda a legrövidebb idő heurisztikára

Egy erőműnek a kieséssel töltött idejét úgy kaphatjuk meg, ha vesszük az előtte meglátogatott kieséssel töltött idejét, és hozzáadjuk a köztük lévő távolságot.

A veszteséget úgy számoljuk, hogy minden erőműnek megnézzük a kieséssel töltött idejét, azaz egy erőműcsoportosulásra a tagok számával kell felszorozni a csoportra jutó kiesést.

1. táblázat: Legrövidebb időválasztása heurisztika út ideje és költsége

Legrövidebb idő választása			
Sorrend	Idáig eljutási idő f	Leállással töltött idő x	Erőművek száma
A	1	$f(A) = 1$	1
B	1	$x(A) + f(B) = 1 + 1 = 2$	5
C	1	$x(B) + f(C) = 2 + 1 = 3$	15
Összesen	$n = 3$	$veszteség = 1 \cdot 1 + 2 \cdot 5 + 3 \cdot 15 = 56$	

2. táblázat: Legnagyobb csoport elsőnek heurisztika út ideje és költsége

Legnagyobb csoport választása			
Sorrend	Idáig eljutási idő f	Leállással töltött idő x	Erőművek száma
C	2	$f(C) = 2$	15
B	1	$x(C) + f(B) = 2 + 1 = 3$	5
A	1	$x(B) + f(A) = 3 + 1 = 4$	1
Összesen	$n = 4$	$veszteség = 2 \cdot 15 + 3 \cdot 5 + 4 \cdot 1 = 49$	

Ebből látszik, hogy mohó algoritmus nem lesz jó, sem mindig a legrövidebb út választása esetén, sem a legnagyobb csoportosulás választása esetén.

Erre példa a fenti eset, ha C-erőműcsoportosulásban 7 db erőmű van 15 helyett.

3. táblázat: Legrövidebb idő heurisztika nyereségesebb

Legrövidebb idő választása			
Sorrend	Idáig eljutási idő f	Leállással töltött idő x	Erőművek száma
A	1	$f(A) = 1$	1
B	1	$x(A) + f(B) = 1 + 1 = 2$	5
C	1	$x(B) + f(C) = 2 + 1 = 3$	7
Összesen	$n = 3$	$veszteség = 1 \cdot 1 + 2 \cdot 5 + 3 \cdot 7 = 32$	

4. táblázat: Legnagyobb csoport heurisztika veszteségesebb

Legnagyobb csoport választása			
Sorrend	Idáig eljutási idő f	Leállással töltött idő x	Erőművek száma
C	2	$f(C) = 2$	7
B	1	$x(C) + f(B) = 2 + 1 = 3$	5
A	1	$x(B) + f(A) = 3 + 1 = 4$	1
Összesen	$n = 4$	$veszteség = 2 \cdot 7 + 3 \cdot 5 + 4 \cdot 1 = 33$	

Azért itt van a fordulópont, mert a rövid úton 2 idő alatt $1 + 5 = 6$ erőművet, a felső 2 hosszú úton 7 erőművet javítunk meg. Ha 2 időegység alatt a C-ben lévő erőműveket javítottuk meg, akkor onnan még 2 időegység kell, hogy eljussunk a „minden erőmű kész" állapotba. Míg ha 2 egység alatt a B-be mentünk, akkor már csak egyetlen időegység kell, hogy elérjük a „minden erőmű kész" állapotot, amikor is nem termelődik több veszteség.

Odaérkezési idők kifejtése ismert adatokkal

A minimalizálandó képletben már az egyes erőművek termelés nélkül töltött ideje van, ami egy származtatott adat a bejárás sorrendjéből. Ezt visszavezetem olyan adatokra, melyek a feladat leírásában explicit szerepelnek, és nem következtetett adatok.

$csúcsok: c_1, c_2, \dots\ c_k, K(kikötő)$
$él\ súlya\ c_1\ és\ c_2\ között: e_{c_1 c_2}$
$csúcsok\ bejárási\ sorrendje: c_{s_1}, c_{s_2}, \dots c_{s_k}$

Ekkor egy erőműig eljutás ideje az összes előtte lévő él közti távolság, beleszámítva a kikötőtől az első erőműig távolságot:

$$x_j = e_{Kc_{s_1}} + \sum_{i=1}^{i=j-1} e_{c_{s_i} c_{s_{i+1}}}$$

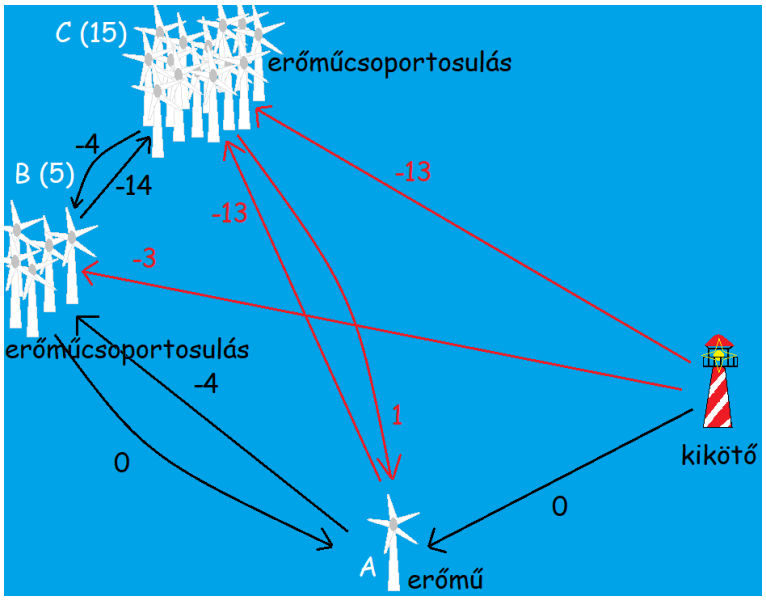
Azaz $e_{Kc_{s_1}}$ távolság minden erőműre fog szerepelni, vagyis k-szor, hasonlóan a $e_{c_{s_1} c_{s_2}}$ az első erőmű kivételével mindenhol számítani fog a késleltetésbe, azaz $k - 1$ –szer:

$$minimalizálandó: \sum_{i=1}^{k} x_i = k \cdot e_{Kc_{s_1}} + (k-1) \cdot e_{c_{s_1}c_{s_2}} + (k-2) \cdot e_{c_{s_1}c_{s_3}} + \cdots e_{c_{s_{k-1}}c_{s_k}}$$

5.4 Megoldási ötletek

Élek súlyozása: a távolságnak és az erőműcsoportosulás méretének kombinálása

Csoportoknál az élsúlyok módosítása: $költség - megjavítható\ erőművek\ száma$. Hogy egyértelmű legyen, mennyivel csökkenne a hibás erőművek száma, minden élet megdupláztam, és irányítottá tettem; a nyíl a hajó haladási irányát jelöli, mindig a fejrésznél lévő erőműszámot kell levonni az útköltségéből, mivel ezek megjavulását nyerjük a haladással.



4. ábra: Élsúlyok módosítva távolság-megjavított erőművek számára

A probléma, hogy lehetnek, és valószínűleg lesznek is negatív körök, így nem fog működni a Bellman-Ford algoritmus[13]. Az ilyen útkereső algoritmusok egy forrás és nyelő között működnek, itt pedig nem meghatározott a nyelő, ráadásul minden csúcsnak szerepelnie kell. Így az se lenne jó, ha pl. egy legrövidebb utat futtatnánk mindig más nyelővel. Egy másik probléma a csoportosítással, hogy intuitívan látszik, mi egy csoport, amelynek tagjait érdemes egyszerre kezelni, de matematikailag nehéz lenne megfogni. Ha mindig a legközelebbieket vesszük egy csoportnak, amelyeket úgymond érdemes egyszerre kezelni, a végén minden bekebeleződne egy naggyá. Ha viszont egy iteráció után leállítjuk, akkor csak a kis csoportokat találjuk meg (~országok), míg az ezekből álló nagyobbak (~kontinensek) ugyanúgy külön lennének, és nem futna jobban az algoritmus, mint csoportok nélkül.

Megj: A valóságban nem ez a feladatok nagyságrendje általában. Egy kikötőből több farm érhető el, amik között azért vannak nagyobb távolságok, mert csak megadott tengermélységig lehet erőműveket telepíteni. Ha egyenetlen a tengerfenék, akkor lesznek foltokban az erőművek.

Hátizsákprobléma[14]

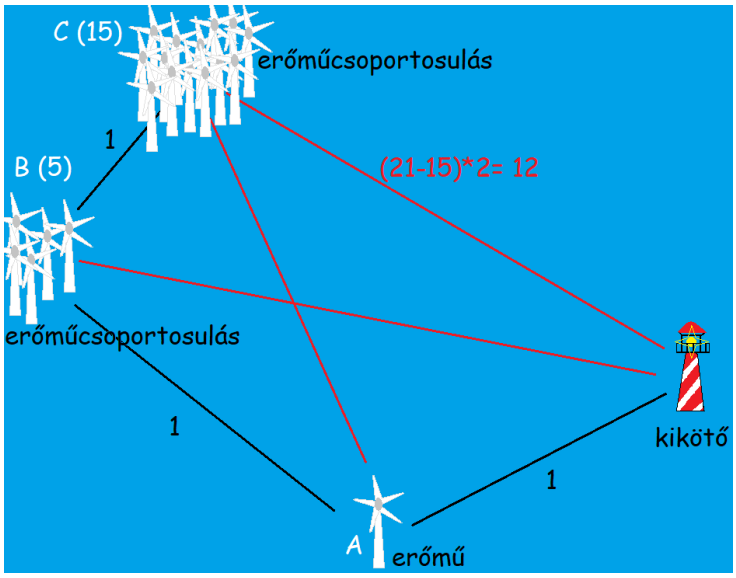
Csoportok helyett visszatérek a turbinák egyenkénti kezelésére, így nem kell a „csúcssúlyokkal” foglalkozni, hiszen minden csúcsra pontosan egy erőmű esik. Dinamikus programozásból a hátizsákproblémára úgy lehet visszavezetni, hogy minden élről eldöntjük, hogy szerepel-e vagy sem.

Itt próbáltam azt a megoldást, hogy korlátoztam az egyszer használatos élek számát, de az nyilván nem jó, hisz mindig adott (k darab él kell, minden erőműbe egy).

Próbáltam úgy lépegetni, hogy mindig adott csoportból lehet éleket válogatni. Itt viszont nem lesz „egyszerűen felső sor két eleméből" eldönthető, hogy az elem bevételével lesz-e kisebb az érték vagy anélkül, mert nem függetlenek egymástól az élek, mint a hátizsákban az értékek, egy él bevételénél azt is kell nézni, hogy lesz-e útvonal is, és hogy akkor melyik másik élet kell kivenni. Itt már nem jobb, mintha az összes lehetséges utat megnéznénk.

Élsúlyok lemondási veszteséggel

Próbáltam az élekbe belekalkulálni, hogy hány másik erőmű megjavításáról és mennyi időre mondok le. Ebben az eseteben viszont csak a kikötőből lehet egy lépésben meghatározni az élek értékét, utána egy él „vesztesége", függ az előtte használt élsorozattól (~összes végigpróbálása).



5. ábra: Egy élsúly lemondási vesztesége

Szummában szereplő értékek élsúlyokkal való kifejezése

Próbáltam egyszerűsíteni a problémát: a „mikor áll szolgálatba" érték gráfalgoritmus esetében nehezen használható, mindig „észben kell tartani" az addigi egész útvonalat, és bármilyen műveletnél újra kell számolni. Átírtam a szummát olyan formára, hogy gráfon is ábrázolható értékek szerepeljenek rajta. Így kaptam a k-val súlyozott összeget.

Erre viszont nem tudtam gráfos algoritmust kitalálni.

$$minimalizálandó\colon \sum_{i=1}^{k} x_i = k \cdot e_{Kc_{s_1}} + (k-1) \cdot e_{c_{s_1}c_{s_2}} + (k-2) \cdot e_{c_{s_1}c_{s_3}} + \cdots e_{c_{s_{k-1}}c_{s_k}}$$

Gráfok helyett analitikusan

Próbáltam a szummát analitikusan nézni, mondván, hogy úgyis minden csúcsból tudunk minden csúcsba menni. De így nem fogunk értelmes útvonalat kapni, csak k darab élet. Kell hozzá a gráf, hogy tényleg mindent csak egyszer érintsünk, és élfolytonos út legyen.

6. Modellek

Az előző fejezetben tárgyalt algoritmusok helyett fekete doboz megoldást választottam. Nem én adom meg az algoritmust, csak a különböző paraméterek estén fennálló esetekről alkotok modelleket. Ezek olyan matematikai műveletekkel vannak megfogalmazva, melyeket értenek a különböző optimalizáló algoritmusok. Ebben a fejezetben csak a matematikai modelleket adom meg, nem térek ki azok szoftveres megvalósítására.

A könnyebb modellezés végett részekre bontottam a problémát, és kevés paraméterrel alkottam modellt, majd ezekből raktam össze bonyolultabb modelleket.

A fekete dobozként használt optimalizáló kimenete a változók értékei, melyeket egyértelműen le lehet képezni egy ütemtervvé. Bemenete a kényszerek és a célfüggvény, melyek a változók függvényei. A kényszereket oly módon kell megadni, hogy azok biztosítsák, hogy az eredmény megfeleljen egy ütemtervnek (például minden erőművet meglátogassunk). A célfüggvény adja meg, hogy egy adott ütemtervnek mi a költsége. Az optimalizáló ezt próbálja meg minimalizálni különböző helyeken történő kiértékelésével, azaz a változók által felvett értékek változtatásaival. Az használt optimalizálók sem első-, sem másodrendű deriváltakat nem használnak.

Mivel offline megoldást választottam, az algoritmus futása során nem keletkezhetnek újabb meghibásodások. A bemeneti erőművek listája csak a rossz erőműveket tartalmazza, sorszámuk értéke semmilyen szemantikát nem tartalmaz, csak a könnyebb hivatkozást segíti. A meghibásodások az algoritmusindulás pillanatában kezdődtek, ha ez nem így lenne, akkor a végső becsült tervhez egy konstanst kell adni, ami az indulás előtti veszteséget tartalmazza, de ez az érték a használt algoritmustól független. A meghibásodások statikusak, nem súlyosbodnak idővel, és mind teljes termeléskieséshez vezetnek. Az időjárástól mint befolyásoló tényezőtől eltekintek. Földrajzi koordináták helyett 2 dimenziós Descartes-koordinátrendszert használok, így jelentős információvesztés nélkül könnyebben számolhatóak a távolságok.

Az esetek leírását a következő struktúrát követve adom meg: először megadom a környezetet, melyben az eset fennáll, ebből következtetéseket vonok le a változók szükségességeire és mértékegységeire. Időpillanat alatt egy ütemtervi időpillanatot értek, azaz egy erőmű vagy a kikötő meglátogatását.

6.1 Alapeset

Az alapesetben minden költséget és azt befolyásoló tényezőt mellőztem, kivéve az erőművek közti távolságot. A későbbi esetekben ezt az esetet fogom bővíteni, ezért kapta az alapeset elnevezést.

Környezet és paraméterek

A legegyszerűbb eset áll fenn: egy hajó és egy technikuscsapat áll rendelkezésre. A hajónak nincsen bérleti díja, sem üzemanyagköltsége, így az utolsó erőműtől a kikötőig tartó útnak nincs költsége. Állandó sebességgel halad, így a távolságot mérhetjük időben, így közelebb kerülve az erőmű állási idejének számításhoz. A technikusok bérétől is eltekintek a költség számításakor. Nincsen munkaidőkorlát, a munkálatok folyamatosan történnek, amíg van meghibásodás.

Minden erőmű azonosan termel, függetlenül az időjárástól, így számolhatjuk közvetlenül euróban a termelést és annak kiesését, így rögtön a veszteség értékéhez jutunk. A meghibások javítási ideje minden erőműre azonos, így az összes veszteséghez egy konstasnként adódna hozzá, ezért elhanyagolható. Így csak a távolságok megtételéhez szükséges idő okoz termelésveszteséget az erőműveknek.

A paraméterek áttekintő leírását az 5. táblázat foglalja össze.

5. táblázat: A legegyszerűbb eset paraméterei

Jelölés	Mértékegység	Indexértékek	Leírás
n	db		meghibásodott erőművek száma
K	$[m;\ m]$		kikötőt jelölő pont (koordinátapár)
e_i	$[m; m]$	$i = 1, 2, \dots n$	az i. erőművet jelölő pont (koordinátapár)
x_i		$i = 1, 2, \dots n$	az i.-nek meglátogatott erőmű
$haszon$	$\frac{EUR}{h}$		az erőművek óránkénti termelése euróban
t_i	h	$i = 1, 2, \dots n$	az x_i erőmű kieséssel töltött ideje
v	$\frac{km}{h}$		a hajó sebessége
$dt(a, b)$	h		a *és* b pont órában mért távolsága, azaz $\frac{d(a,b)}{1000 \cdot v}$

Kényszerek

Az x_i változó által felvehető értékek csak erőművek lehetnek, és egyszerre pontosan egy erőmű értékét veheti fel.

$$\forall\, 1 \le i \le n \textit{ esetén}: \mathbb{1}(\mathrm{x}_i = \mathrm{e}_1) + \mathbb{1}(\mathrm{x}_i = \mathrm{e}_2) + \cdots \mathbb{1}(\mathrm{x}_i = \mathrm{e}_n) = 1 \text{ azaz } \sum_{j=1}^{n} \mathbb{1}(x_i = e_j) = 1$$

A hajó minden erőművet pontosan egyszer látogathat meg.

$$\forall\, 1 \le i, j \le n \textit{ esetén}:$$

$$i \neq j \Rightarrow x_i \neq x_j$$

Költségek számítása

Az első erőmű csak annyi időt fog leállva tölteni, amíg a kikötőből odaér a hajó:

$x_1 = e_j \Rightarrow\ t_1 = dt(K, e_j)$ $\qquad 1 \le j \le n$

Az i.-nek $(i \neq 1)$meglátogatott erőmű annyi ideig van meghibásodva, mint az $(i-1)$.-nek meglátogatott, de az i. még a két erőmű közti távolság megtételéhez szükséges időt is hibásan tölti, azaz amíg oda nem érnek megjavítani.

$$\forall\, 2 \le i \le n, 1 \le j \le n \textit{ esetén}:$$

$$x_i = e_j \Rightarrow\ t_i = t_{i-1} + dt(x_{i-1}, e_j)$$

Célfüggvény

Mivel minden erőmű azonos haszonnal termel, a veszteséget úgy kapjuk, hogy az összes leállási időt[1] megszorozzuk az óránkénti haszonkieséssel.

$$Veszteség\ [EUR] = \sum_{i=1}^{n} t_i \cdot haszon$$

6.2 Üzemanyagköltség figyelembevétele

Az Alapesetet bővítettem a hajó üzemanyagköltségével, ezért extra költségként jelentkezik az út megtétele, és már a kikötőbe visszatérés is költség lesz. A paraméterek megegyeznek az 5. táblázatban megadott paraméterekkel, kiegészítve az üzemanyag figyelembevételéhez szükségesekkel, amelyeket a 6. táblázatban adtam meg.

6. táblázat: Az üzemanyagköltség figyelembe vételéhez szükséges paraméterek

Jelölés	Mértékegység	Leírás
$táv$	m	a hajó által megtett távolság
$üzemanyagköltség$	$\frac{EUR}{km}$	a hajó üzemanyagköltsége 1km-re

Kényszerek

Az x_i változó által felehető értékek csak erőművek lehetnek, és egyszerre pontosan egy erőmű értékét veheti fel.

$$\forall\ 1 \le i \le n\ esetén: \mathbb{1}(\mathrm{x}_i = \mathrm{e}_1) + \mathbb{1}(\mathrm{x}_i = \mathrm{e}_2) + \cdots \mathbb{1}(\mathrm{x}_i = \mathrm{e}_n) = 1 \text{ azaz } \sum_{j=1}^{n} \mathbb{1}(x_i = e_j) = 1$$

A hajó minden erőművet pontosan egyszer látogathat meg.

$$\forall\ 1 \le i,j \le n\ esetén:$$

$$i \ne j \Rightarrow x_i \ne x_j$$

Költségek számítása

Az első erőmű leállási ideje mellett az odáig lévő út megtétele is költség.

$$x_1 = e_j \Rightarrow\ t_1 = dt(K, e_j) \qquad 1 \le j \le n$$

$$táv = d(K, e_j)$$

A többi erőmű kiesésén kívül a köztük lévő út megtétele is költség.

$$\forall\ 2 \le i \le n, 1 \le j \le n\ \ esetén:$$

1 Ami az erőművek egyesével vett leállásidejének súlyozott összege, mivel mindegyik egyesével veszteséget okoz. Tehát például az első erőműig megtett út nem egyszer, hanem n-szer okoz veszteséget, hiszen az az idő minden javítást hátráltat. Ezért nem azonos a teljes távolság megtételéhez szükséges idővel, hanem annak erőművek közti darabjainak még hátralevő erőműszámmal súlyozott összege.

$$x_i = e_j \Rightarrow\ t_i = t_{i-1} + dt(x_{i-1}, e_j)$$

$$táv\ += d(x_{i-1}, e_j)$$

Az utolsó erőműtől a kikötőig való eljutásnak is van költsége az üzemanyagköltség miatt.

$$táv += d(x_n, K)$$

Célfüggvény

Az erőművek termeléskiesésén kívül az összes megtett távolságnak kell a szorzatát venni a km-enkénti üzemanyagfogyasztással. A koordinátarendszer méteres felbontása miatt a távolságot méterben kapjuk, ezért át kell váltani km-re a fogyasztás számítása előtt.

$$Veszteség\ [EUR] = \sum_{i=1}^{n} t_i \cdot haszon + \frac{táv}{1000} \cdot üzemanyagköltség$$

6.3 Változó termelés és javítási idő figyelembevételeAlapeset

Az alapesetben minden költséget és azt befolyásoló tényezőt mellőztem, kivéve az erőművek közti távolságot. A későbbi esetekben ezt az esetet fogom bővíteni, ezért kapta az alapeset elnevezést.

Környezet és paraméterek

A legegyszerűbb eset áll fenn: egy hajó és egy technikuscsapat áll rendelkezésre. A hajónak nincsen bérleti díja, sem üzemanyagköltsége, így az utolsó erőműtől a kikötőig tartó útnak nincs költsége. Állandó sebességgel halad, így a távolságot mérhetjük időben, így közelebb kerülve az erőmű állási idejének számításhoz. A technikusok bérétől is eltekintek a költség számításakor. Nincsen munkaidőkorlát, a munkálatok folyamatosan történnek, amíg van meghibásodás.

Minden erőmű azonosan termel, függetlenül az időjárástól, így számolhatjuk közvetlenül euróban a termelést és annak kiesését, így rögtön a veszteség értékéhez jutunk. A meghibások javítási ideje minden erőműre azonos, így az összes veszteséghez egy konstasnként adódna hozzá, ezért elhanyagolható. Így csak a távolságok megtételéhez szükséges idő okoz termelésveszteséget az erőműveknek.

A paraméterek áttekintő leírását az 5. táblázat foglalja össze.

5. táblázat: A legegyszerűbb eset paraméterei

Jelölés	Mértékegység	Indexértékek	Leírás
n	db		meghibásodott erőművek száma
K	$[m;\ m]$		kikötőt jelölő pont (koordinátapár)
e_i	$[m; m]$	$i = 1, 2, \dots n$	az i. erőművet jelölő pont (koordinátapár)
x_i		$i = 1, 2, \dots n$	az i.-nek meglátogatott erőmű
$haszon$	$\frac{EUR}{h}$		az erőművek óránkénti termelése euróban
t_i	h	$i = 1, 2, \dots n$	az x_i erőmű kieséssel töltött ideje

v	$\frac{km}{h}$		a hajó sebessége
$dt(a,b)$	h		a és b pont órában mért távolsága, azaz $\frac{d(a,b)}{1000 \cdot v}$

Kényszerek

Az x_i változó által felvehető értékek csak erőművek lehetnek, és egyszerre pontosan egy erőmű értékét veheti fel.

$$\forall\, 1 \le i \le n \; esetén: \mathbb{1}(x_i = e_1) + \mathbb{1}(x_i = e_2) + \cdots \mathbb{1}(x_i = e_n) = 1 \text{ azaz } \sum_{j=1}^{n} \mathbb{1}(x_i = e_j) = 1$$

A hajó minden erőművet pontosan egyszer látogathat meg.

$$\forall\, 1 \le i, j \le n \; esetén:$$

$$i \neq j \Rightarrow x_i \neq x_j$$

Költségek számítása

Az első erőmű csak annyi időt fog leállva tölteni, amíg a kikötőből odaér a hajó:

$x_1 = e_j \Rightarrow\ t_1 = dt(K, e_j) \qquad 1 \le j \le n$

Az i.-nek $(i \neq 1)$meglátogatott erőmű annyi ideig van meghibásodva, mint az $(i-1)$.-nek meglátogatott, de az i. még a két erőmű közti távolság megtételéhez szükséges időt is hibásan tölti, azaz amíg oda nem érnek megjavítani.

$$\forall\, 2 \le i \le n, 1 \le j \le n \;\; esetén:$$

$$x_i = e_j \Rightarrow\ t_i = t_{i-1} + dt(x_{i-1}, e_j)$$

Célfüggvény

Mivel minden erőmű azonos haszonnal termel, a veszteséget úgy kapjuk, hogy az összes leállási időt megszorozzuk az óránkénti haszonkieséssel.

$$Veszteség\ [EUR] = \sum_{i=1}^{n} t_i \cdot haszon$$

ÜzemanyagköltségfigyelembevételeAz Üzemanyagköltség figyelembevétele esetet bővítettem azzal, hogy az erőművek különböző haszonnal termelnek, és a javításokhoz is különböző idők szükségesek. A 7. táblázatban találjuk ezen kiegészítéshez szükséges paramétereket, az többi paraméter megegyezik a 6. táblázat és 5. táblázat paramétereivel.

7. táblázat: A különböző javítási időhöz és haszonhoz szükséges paraméterek

Jelölés	Mértékegység	Index értékek	Leírás
$haszon_i$	$\frac{EUR}{h}$	$i = 1, 2, \ldots n$	az i. erőmű óránkénti termelése
jav_i	h	$i = 1, 2, \ldots n$	az i. erőmű megjavításához szükséges idő

Kényszerek

Az x_i változó által felvehető értékek csak erőművek lehetnek, és egyszerre pontosan egy erőmű értékét veheti fel.

$$\forall\, 1 \le i \le n \; esetén: \mathbb{1}(\mathrm{x}_i = \mathrm{e}_1) + \mathbb{1}(\mathrm{x}_i = \mathrm{e}_2) + \cdots \mathbb{1}(\mathrm{x}_i = \mathrm{e}_n) = 1 \text{ azaz } \sum_{j=1}^{n} \mathbb{1}(x_i = e_j) = 1$$

A hajó minden erőművet pontosan egyszer látogathat meg.

$$\forall\, 1 \le i, j \le n \; esetén:$$

$$i \neq j \Rightarrow x_i \neq x_j$$

Költségek számítása

Az elsőnek látogatott erőmű leállással töltött idejéhez hozzáadódik a javításához szükséges idő.

$$x_1 = e_j \Rightarrow\ t_1 = dt\big(K, e_j\big) + jav_j \qquad 1 \le j \le n$$

$$táv = d(K, e_j)$$

A többi erőmű esetében is hozzáadódik a leállási időhöz a javítás. Továbbra is helyes marad i. erőmű idejét az $(i-1)$. erőmű idejét felhasználva számolni. hiszen egy hajónk van; amíg javítják az i. erőművet, addig a többi erőmű vesztegelve áll. A rekurzív összeadással ismételten szerepelni fog minden erőműnek az idejében az őt megelőző összes idő.

$$\forall\, 2 \le i \le n, 1 \le j \le n \;\; esetén:$$

$$x_i = e_j \Rightarrow\ t_i = t_{i-1} + dt\big(x_{i-1}, e_j\big) + jav_j$$

$$táv \mathrel{+}= d(x_{i-1}, e_j)$$

A kikötőbe visszatérés költsége változatlan.

$$táv \mathrel{+}= d(x_n, K)$$

Célfüggvény

Mivel minden erőmű termelése más, így minden erőmű kiesésének idejére vett veszteségét a saját termelésével kell számolni. Mivel t_i az i.-nek meglátogatott erőművet jelenti, ami nem feltétlenül egyezik meg e_i-vel, ezért nem lenne helyes $haszon_i$-vel szorozni. Az i.-nek meglátogatott erőművet x_i jelenti, így $haszon_{x_i}$-vel kell számolni.

$$Veszteség\ [EUR] = \sum_{i=1}^{n} (t_i \cdot haszon_{x_i}) + \frac{táv}{100} \cdot \text{üzemanyagköltség}$$

6.4 Több hajó ütemezése

Az Alapesethez képesti változás, hogy több hajó van, mindegyik azonos sebességű, és csak egy csapat tartozik hozzájuk. A csapat hozzá van rendelve a hajóhoz, más hajóra nem szállhatnak fel.

Az Üzemanyagköltség figyelembevétele nem történik meg, és bérleti díjuk sincs a hajóknak. Nem feltétlen kell minden hajót munkába állítani, és amennyiben egy hajónál többet munkába állítunk, egy hajónak sem kell n erőművet meglátogatnia. A hajók számozásának értéke nem takar szemantikát, csak

megkönnyíti a rájuk való hivatkozást. A Változó termelés és javítási idő figyelembevétele itt is megtörténik.

A paraméterek ennek megfelelően megegyeznek az 5. táblázat és a 7. táblázat paramétereivel, annyi eltéréssel, hogy több hajónak megfelelően a bejárási sorrendet és az erőművekhez tartozó várási időt tartalmazó változót elláttam egy hajóra vonatkozó indexszel. Ezen módosításokat és a szükséges új paramétereket tartalmazza a 8. táblázat.

8. táblázat: A több hajóra ütemezéshez szükséges paraméterek

Jelölés	Mértékegység	Index értékek	Leírás
H	db		hajók száma
h_i		$i = 1,2,\dots H$	az i. hajó
$x_{i_{h_j}}$		$i = 1,2,\dots n$ $j = 1,2,\dots H$	az i.-nek meglátogatott erőmű a j. hajóval
$\emptyset$			üresjárat, $x_{i_{h_j}} = \emptyset$ amikor j. hajó i.-nek nem látogat meg erőművet
$t_{i_{h_j}}$	h		$x_{i_{h_j}}$ erőmű kieséssel töltött ideje

Kényszerek

Minden hajónak van n állomása sorrendben, ami vagy erőmű, vagy semmi ($\emptyset$), de egy időben ezek közül pontosan egyet vehet fel.

$$\forall\, 1 \le i \le n, 1 \le j \le H \; esetén:$$

$$\mathbb{1}\left(x_{i_{h_j}} = \mathrm{e}_1\right) + \mathbb{1}\left(x_{i_{h_j}} = \mathrm{e}_2\right) + \cdots + \mathbb{1}\left(x_{i_{h_j}} = \mathrm{e}_n\right) + \mathbb{1}\left(x_{i_{h_j}} = \emptyset\right) = 1$$

$$\text{azaz } \sum_{k=1}^{n} \mathbb{1}(x_{i_{h_j}} = e_k) + \mathbb{1}\left(x_{i_{h_j}} = \emptyset\right) = 1$$

Ha egy hajó nem erőműhöz ($\emptyset$) megy egyszer, akkor többet nem is mehet erőműhöz, visszatér a kikötőbe.[2] Azaz az összes utána következő állomás $\emptyset$, hiszen annak nincs értelme, hogy két javítás között megálljon a vízen cél nélkül.

$$ha\; x_{i_{h_j}} = \emptyset \Rightarrow \sum_{k=i+1}^{n} \mathbb{1}\left(x_{k_{h_j}} = \emptyset\right) = n - (i+1)$$

Nem mehetnek hajók kétszer ugyanahhoz az erőműhöz, $\emptyset$-be viszont mehetnek többször. Tehát, ha a két vizsgált ütemtervelem nem azonos (azaz különböznek az időpontban vagy a hajóban), és az értékük nem $\emptyset$, akkor nem egyezhetnek meg.

$$\forall\, 1 \le i, j \le n, 1 \le a, b \le H$$

$$\big((i \ne j) \vee (a \ne b)\big) \wedge \left(\left(x_{i_{h_a}} \ne \emptyset\right) \vee \left(x_{i_{h_b}} \ne \emptyset\right)\right) \Rightarrow x_{i_{h_a}} \ne x_{j_{h_b}}$$

[2] Ez nem vesz el optimális megoldást, mivel a háromszögegyenlőtlenség miatt nem lehet rövidebb út A és B között az, ha még K(ikötő) pontot is érinti.

Egyszerűbben: csak akkor egyezhetnek meg, ha ugyanannak a hajónak ugyanarról az állomásáról beszélünk, vagy ha zérus értéke van mindkettőnek.

$$\big((i=j)\ \wedge\ (a=b)\big)\vee\Big(\big(x_{i_{h_a}}=\emptyset\big)\ \wedge\big(x_{i_{h_b}}=\emptyset\big)\Big)\Leftrightarrow x_{i_{h_a}}=x_{j_{h_b}}$$

Költségek számítása

A számítások hasonlóan történnek a fenti esetekhez, annyi különbséggel, hogy minden hajóra egyesével el kell őket végezni. Minden hajónak lesz egy első erőműve, ahová a kikötőből megy.

$$1\le k\le n, 1\le j\le H\ esetén:$$

$$x_{1_{h_j}}=e_k\Rightarrow\ t_{1_{h_j}}=dt(K,e_k)+jav_k$$

Az többi erőműnek már nem kell minden előtte lévőt megvárnia, csak azokat, mivel dedikált hajója van, ezért csak azok hátráltatják a megjavulását, akik a hajójának az ütemtervében előtte szerepelnek.

$$\forall\ 2\le i\le n, 1\le k\le n, 1\le j\le H\ \ esetén:$$

$$x_{i_{h_j}}=e_k\Rightarrow t_{i_{h_j}}=t_{i-1_{h_j}}+dt\left(x_{i-1_{h_j}},e_k\right)+jav_k$$

Célfüggvény

Minden i-re összegezni kell az összes hajónak az i.-nek meglátogatott erőmű veszteségét. Az $haszon$ indexelésének logikája megegyezik az 6.3. fejezet Célfüggvény alfejezetében tárgyaltakkal, annyi kiegészítéssel, hogy most minden hajónak saját erőműsorrendje van.

$$Veszteség\ [EUR]=\sum_{i=1}^{n}\sum_{j=1}^{H}\left(t_{i_{h_j}}\cdot haszon_{x_{i_{h_j}}}\right)$$

6.5 Több hajó ütemezése benzinköltséggel

Ebben az esetben ér össze az összes fent tárgyalt eset. Azaz nem más, mint az Alapeset Üzemanyagköltség figyelembevétele és Változó termelés és javítási idő figyelembevétele mellett Több hajó ütemezése. A hajóknak különböző az üzemanyaköltségük, így mindegyiknek külön ismerni kell a megtett útját, mivel azzal arányosan számítódik a költsége. Ehhez a 6. táblázat paramétereit el kell látni indexekkel, ezt mutatja a 9. táblázat. A többi paraméter azonos az 5. táblázat, a 7. táblázat és 8. táblázat paramétereivel.

9. táblázat Több hajó ütemezéséhez szükséges kiegészítő paraméterek

Jelölés	Mértékegység	Index értékek	Leírás
$táv_i$	m	$i=1,2,\dots H$	az i. hajó által megtett távolság
$üzemanyagköltség_i$	$\frac{EUR}{km}$	$i=1,2,\dots H$	az i. hajó üzemanyagköltsége 1km-re

Kényszerek

Minden hajónak van n állomása sorrendben, ami vagy erőmű, vagy semmi ($\emptyset$), de egy időben ezek közül pontosan egyet vehet fel.

$$\forall\ 1\le i\le n, 1\le j\le H\ esetén:$$

$$\mathbb{1}\left(x_{i_{h_j}} = \mathrm{e}_1\right) + \mathbb{1}\left(x_{i_{h_j}} = \mathrm{e}_2\right) + \cdots + \mathbb{1}\left(x_{i_{h_j}} = \mathrm{e}_n\right) + \mathbb{1}\left(x_{i_{h_j}} = \emptyset\right) = 1$$

$$\text{azaz } \sum_{k=1}^{n} \mathbb{1}(x_{i_{h_j}} = e_k) + \mathbb{1}\left(x_{i_{h_j}} = \emptyset\right) = 1$$

Ha egy hajó nem erőműhöz ($\emptyset$) megy egyszer, akkor többet nem is mehet erőműhöz, visszatér a kikötőbe. Így egyértelmű, hogy az első $\emptyset$ előtti erőmű az utolsó megállója az adott hajónak, ami megkönnyíti a költségfüggvény számítását, mivel annak most már része az utolsó erőműtől a kikötőbe visszatérési üzemanyagfogyasztás.

$$ha\ x_{i_{h_j}} = \emptyset \Rightarrow \sum_{k=i+1}^{n} \mathbb{1}\left(x_{k_{h_j}} = \emptyset\right) = n - (i+1)$$

Nem mehetnek hajók kétszer ugyanahhoz az erőműhöz:

$$\forall\ 1 \le i, j \le n, 1 \le a, b \le H$$

$$\left((i = j)\ \wedge\ (a = b)\right) \vee \left(\left(x_{i_{h_a}} = \emptyset\right)\ \wedge \left(x_{i_{h_b}} = \emptyset\right)\right) \Leftrightarrow x_{i_{h_a}} = x_{j_{h_b}}$$

Költségszámítás

A költségek számítása hasonló a Több hajó ütemezése esethez, annyi különbséggel, hogy hajónként számolni kell a távolságokat a különböző üzemanyagköltség miatt.

Tehát minden hajó első megállója esetén a költség az első erőmű leállással töltött ideje, és a kikötőből az erőműig eljutás költsége.

$$1 \le k \le n, 1 \le j \le H\ esetén:$$

$$x_{1_{h_j}} = e_k \Rightarrow\ t_{1_{h_j}} = dt(K, e_k) + jav_k$$

$$táv_j = d(K, e_k)$$

A többi erőmű esetén annyi változás történik, hogy nem a kikötőtől számítjuk a távolságot, hanem az előző erőműtől, és minden erőmű leállási költsége rekurzívan számolódik a hajójuk útján előtte lévők idejével.

$$\forall\ 2 \le i \le n, 1 \le k \le n, 1 \le j \le H\ \ esetén:$$

$$x_{i_{h_j}} = e_k \Rightarrow t_{i_{h_j}} = t_{i-1_{h_j}} + dt\left(x_{i-1_{h_j}}, e_k\right) + jav_k$$

$$táv_j \mathrel{+}= d\left(x_{i-1_{h_j}}, e_k\right)$$

Minden hajónak az utoljára látogatott erőművétől a kikötőbe visszatérésig is van költsége. Az utolsó erőmű az első $\emptyset$ előtt meglátogatott erőművel fog megegyezni, tehát most nem mindig igaz, hogy az n. indexű sorrend elemet kérjük le, mint egy hajó esetén. De az is előfordulhat, hogy egy hajót használunk az ütemezéshez, ez akkor fordulhat elő, ha olyan magas a benzinköltsége az összes hajónak az erőművek kieső termeléséhez képest, hogy jobban megéri lassabban javítani az erőműveket, mint hajókat indítani. Ebben az estben is tudni kell számolni a kikötőbe való visszatérés költségét, ami ez esetben pontosan az n. erőmű meglátogatása után fog történni.

$$\forall\ 2 \le i \le n, 1 \le j \le H\ \ esetén:$$

$$\left(\left(x_{i_{h_j}} = \emptyset\right) \wedge \left(x_{i-1_{h_j}} \neq \emptyset\right)\right) \Rightarrow\ táv_j \mathrel{+}= d(x_{i-1_{h_j}}, K)$$

$$x_{n_{h_j}} \neq \emptyset \Rightarrow\ táv_j \mathrel{+}= d(x_{n_{h_j}}, K)$$

Amennyiben egy hajót egyáltalán nem használtunk, annak semmilyen költsége sincs, de ez nem okoz problémát, mivel a fenti költségszámítási feltételek csak akkor teljesülnek, ha van nem ∅ állomása egy hajónak.

Célfüggvény

Minden i-re összegezni kell az összes hajónak az i.-nek meglátogatott erőmű veszteségét, és minden hajóra a megtett távolságával arányosan annak üzemanyagköltségét.

$$Veszteség\ [EUR] = \sum_{i=1}^{n}\sum_{j=1}^{H}\left(t_{i_{h_j}} \cdot haszon_{x_{i_{h_j}}}\right) + \sum_{j=1}^{H}\left(táv_j \cdot üzemanyagköltség_j\right)$$

6.6 Több csapat egy hajóval

Üzemanyagköltség figyelembevétele és Változó termelés és javítási idő figyelembevétele mellett egy hajó van, de azon több csapat is utazik. A hajó lerakja a csapatokat különböző erőműveknél, majd javítás végeztével felveszi őket, és viszi őket a csapat által következőnek javítandó erőműhöz.

Ezen cselekvések sorrendjére csak annyi megkötés van, hogy nem vehet fel egy csapatot addig, amíg nem tette le, és nem tehet le több csapatot, mint amennyivel elindult. Ezzel előnyt tesz szert a heurisztikával szemben, bővebben erről a 8.4 fejezetben írok.

A kikötőbe visszatérés az után történik, hogy minden csapat befejezte a javításokat, s a hajó felvette őket. Mivel nem modellezek különböző típusú meghibásodásokat, melyekhez különböző alkatrészeket kellene szállítani és áthelyezni az erőműre, ezért a csapatok erőműnél leszállásnak és felszállásnak ideje azonos mindenen erőműre, így 0-nak tekinthető.

A csapatok felszedéséhez értelemszerűen vissza kell mennünk az erőművekhez, ezeknek az utaknak is lesz üzemanyagköltségük, és a többi erőműnek a leállással töltött idejét is hosszabbítják. Az üzemanyagköltséget egyszerűen lehet számolni, a hajó megtett távolságát meg kell növelni a távolsággal. A leállási idő már bonyolultabb, mivel el is kell jutunk ez erőműig, és az ottani csapatnak is végeznie kell, ezeknek a figyelembevételéről gondoskodik az $idő$ változó.

Az 5. táblázat, a 6. táblázat és a 7. táblázat paramétereit használom, ezen kívül az új paramétereket és a régiek változtatását adja meg a 10. táblázat.

10. táblázat: Több csapat egy hajón esethez szükséges paraméterek

Jelölés	Mértékegység	Index értékek	Leírás
CS	db		csapatok száma
x_i		$i = 1, 2, \dots 2n$	az i. állomása a hajónak, vagy letesz csapatot, vagy felvesz
$idő$	h		az eltelt időt tartalmazza
$állapot_{i\,j}$	$\{R, J, K\}$	$i = 1, 2, \dots n$ $j = 1, 2, \dots 2n$	Az e_i erőmű állapota a j. időpillanatban: Rossz, Javítás alatt, Kész. $j = 0$-ban minden Rossz. Kész állapot nem a javítás befejeztével, hanem a csapat felszedésével következik be.
t_i	h	$i = 1, 2, \dots 2n$	az i.-nek megjavított erőmű leállással töltött ideje, minden erőműre 2-szer szerepel, a felvétel alkalmával 0 értékű

Kényszerek

Az x_i változó által felvehető értékek csak erőművek lehetnek, és egyszerre pontosan egy erőmű értékét veheti fel.

$$\forall\, 1 \le i \le 2n \textit{ esetén}: \mathbb{1}(\mathrm{x}_i = \mathrm{e}_1) + \mathbb{1}(\mathrm{x}_i = \mathrm{e}_2) + \cdots \mathbb{1}(\mathrm{x}_i = \mathrm{e}_n) = 1 \text{ azaz } \sum_{j=1}^{n} \mathbb{1}(x_i = e_j) = 1$$

A hajó pontosan kétszer áll meg minden egyes erőműnél, egyszer, hogy letegye a csapatot, egyszer, hogy felvegye.

$$\forall\, 1 \le j \le n \textit{ esetén}: \sum_{i=1}^{2n} \mathbb{1}(x_i = e_j) = 2$$

Csapatok számánál több munkálat nem lehet egyszerre folyamatban egyetlen időpillanatban sem.

$$\forall\, 1 \le k \le 2n \textit{ esetén}: \sum_{i=1}^{nk} \mathbb{1}\big(\textit{állapot}_{i_k} = J\big) \le CS$$

Költségek számítása

Az elsőnek látogatott erőműnél az $idő$ még egyszerű, csak a kikötő és az első erőmű közti távolság megtételére kellett várni. Az erőmű állási ideje az odáig való eljutással eltelt idő és az erőmű megjavításának ideje, az erőmű állapota „javítás alatt” lesz. A hajó megtett távolsága nő a kikötő és az első erőmű távolságával.

$$1 \le j \le n \text{ és } x_1 = e_j \textit{ esetén}:$$

$$idő = dt\big(K, e_j\big)$$

$$t_1 = idő + jav_j$$

$$táv = d(K, e_j)$$

$$\textit{állapot}_{j_1} = J$$

A többi erőmű esetén a hajó megtett távolsága nő az úttal, és az idő is telik ezen távolságok megtétele közben.

$$1 \le j \le n, 2 \le i \le 2n \text{ és } x_i = e_j \textit{ esetén}:$$

$$idő \mathrel{+}= dt\big(x_{i-1}, e_j\big)$$

$$táv \mathrel{+}= d(x_{i-1}, e_j)$$

Annak függvényében, hogy elsőnek értünk az erőműhöz, vagy már jártunk itt, és a csapat felszedése végett vagyunk itt, megkülönböztetek két esetet.

a) Elsőnek játunk itt. Ekkor javítani vagyunk itt az erőművet, tehát az állapotát „javítás alatt”-ra állítjuk. Mivel több csapat van, nem kell bevárni, amíg az előző erőművek megjavítódnak, a megjavítódásra várakozás tisztán az idáig eltelt idő és az erőmű saját javítására fordított idő.

$$ha\ \textit{állapot}_{j_{i-1}} = R, akkor:$$

$$t_i = idő + jav_j$$

$$\textit{állapot}_{j_i} = J$$

b) Már jártunk itt. Mivel minden időpillanathoz tartozik erőmű várakozási ideje, ezért itt is van egy, melynek értékét 0-ra kell állítani, lévén, hogy ennek az erőműnek a várakozási idejét már

megadtuk, amikor elsőnek jártunk itt. Az erőmű el is készült, így át kell állítani az állapotát, hogy a kint lévő csapatok számolása helyes maradjon. Mennyi idő is telik el a csapat felvétele művelettel? Két eset lehetséges: hamarabb értünk oda, mint ahogy megjavították, ekkor a várakozási idő annyi lesz, amíg be nem fejezik, mivel addig nem indulhatunk tovább. Ekkor az eltelt idő megegyezik az erőmű leállással töltött idejével. A másik eset, hogy már hamarabb befejezték a javítást, csak nem tud a hajó elég gyorsan odaérni, ekkor az eddig eltelt idő a 2 erőmű távolságának megtételéhez szükséges idővel nő, de mivel ez minden esetben szerepel, fent már hozzáadtuk az $időhöz$ ezen értéket, tehát egyszerűen felülírjuk önmagával az érétket. A két esetet egybefoglalva: mindig a nagyobb idő értéket kell választani, hogy mindkettő cselekvés befejeződhessen.

$$ha\ állapot_{j_{i-1}} = J$$

$$t_i = 0$$

$$állapot_{j_i} = K$$

$$idő\ = \max\{t_{x_j}; idő\}$$

Célfüggvény

A veszteség most is minden erőmű leállási idejével és a kiesett termeléséből áll, itt is figyelve az indexelésre, mivel i nem erőművet, hanem látogatási sorrendet jelent, tehát erőművedik hasznot x_i-vel kapunk. 2-szer annyi lépés van a csapatok felszedése miatt, ezért a szumma $2n$-ig megy, de a számítás helyes lesz, mivel csapatfelvételnél t_i 0 értéket vesz fel, azt hiába szorozzuk meg ismételten a haszonnal, 0 értékű lesz. A költséghez tartozik még a hajó üzemanyagköltsége arányosan a megtett távolsággal.

$$Veszteség\ [EUR] = \sum_{i=1}^{2n} \left(t_i \cdot haszon_{x_i}\right) + táv \cdot üzemanyagköltség$$

6.7 Munkaidő figyelembevétele

A feladat annyiban visszaegyszerűsödik, hogy egy hajónk és egy csapatunk van, viszont vissza kell érni munkaidő végeztével a kikötőbe, és a következő napon kell folytatni a javításokat. Nem lehet megszakítani egy javítást; vagy teljesen befejezik, vagy el sem kezdik aznap. Hogy ne lehessen olyan munka, amit emiatt soha nem lehet elvégezni, egyik erőmű javítási ideje és az erőműig a kikötőből való odajutási idő és visszajutási idő nem haladhatja meg egy nap ledolgozható óráit. A meghibásodások az első nap 0:00 időpillanatban léptek fel, de a műszak lehet, hogy csak később kezdődik, ezért a köztes időt is fel kell számolni veszteségnek. Ha egy erőművet nem az első napon javítunk meg, akkor az eddig eltelt napok számával megszorzott 24 órát is hozzá kell adni a kieséséhez, ezen szorzás egyszerűsége végett a napok számozása 0-tól indul.

Ezek mellett tartalmaz mindent, amit az Alapeset, Üzemanyagköltség figyelembevétele, Változó termelés és javítási idő figyelembevétele esetek tartalmaznak. Ennek megfelelően kisebb módosításokkal és új paraméter felvételével – melyeket a 11. táblázat tartalmazza – az 5. táblázat, a 6. táblázat és a 7. táblázat paramétereit használom.

11. táblázat: A munkaidő figyelembevételéhez szükséges paraméterek

Jelölés	Mértékegység	Index értékek	Leírás
t_i		$i = 1, 2, \ldots n$	újítás: i. erőmű hibásan töltött ideje, nem a k.-nak meglátogatotté

T	h		az egy nap ledolgozható maximum idő, a ki- és behajózással együtt
$idő_j$	h	$0 \leq j \leq n-1$	az aktuálisan felhasznált idő a j. napon
x_{j_i}		$1 \leq i \leq n$ $0 \leq j \leq n-1$	i.-nek megjavított erőmű a j. napon. Mivel minden javítás maximálisan egy napig tarthat, elégnek kell lennie n napnak
$kezdés$	h		megadja a műszak kezdésének időeltolását 0:00-tól
$erőműszám_j$	db	$0 \leq j \leq n$	j. napra tett javítandó erőművek száma

Minden hibának egy nap alatt javíthatónak kell lennie:

$$\forall\, 1 \leq k \leq n \; esetén \; jav_k + dt(e_k, K) \cdot 2 \leq T$$

Kényszerek

- Az x_{j_i}változó által felvehető értékek csak erőművek lehetnek, és egyszerre pontosan egy erőmű értékét veheti fel.

$$\forall\, 1 \leq i \leq n, 0 \leq j \leq n-1, 1 \leq k \leq n \; esetén:$$

$$\mathbb{1}\left(\mathrm{x}_{\mathrm{j}_i} = \mathrm{e}_1\right) + \mathbb{1}\left(\mathrm{x}_{\mathrm{j}_i} = \mathrm{e}_2\right) + \cdots \mathbb{1}\left(\mathrm{x}_{\mathrm{j}_i} = \mathrm{e}_n\right) = 1 \text{ azaz } \sum_{k=1}^{n} \mathbb{1}(x_{j_i} = e_k) = 1$$

A hajó minden erőművet pontosan egyszer látogathat meg, mivel nincsen félbehagyott munka. Tehát két ütemtervben felvett erőmű csak akkor egyezhet meg, ha azonos napon, azonos időben vette fel őket a változó.

$$\forall 1 \leq i_a, i_b \leq n, 0 \leq j_a, j_b \leq n-1$$

$$\left((i_a = i_b) \wedge (j_a = j_b)\right) \Leftrightarrow x_{j_{ai_a}} = x_{j_{bi_b}}$$

Nem dolgozhatnak többet a technikusok a műszaknál:

$$\forall\; 0 \leq j \leq n-1 \; esetén: idő_j \leq T$$

Költségek számítása

A számításokat minden napra azonosan kell elvégezni, a számítások az adott napon belül az első, utolsó és közbülső erőművekre különböznek.

Első erőműnél az erőmű kieséssel töltött ideje a szokásos módon az éjféltől a munkaidő kezdéséig eltelt idő, az erőműig eljutási idő, és annak javítása adja, viszont lehetséges, hogy már több napja vár a javításra az erőmű, ezért a javításig eltelt napok egész időtartamát hozzá kell adni a leállási időhöz. A többi paraméter változása értelemszerű.

$$i = 1 \text{ és } x_{j_1} = e_k \; esetén:$$

$$t_k = kezdés + j \cdot 24 + jav_k + dt(e_k, K)$$

$$táv += d(e_k, K)$$

$$idő_j = jav_k + dt(e_k, K)$$

Köztes erőmű esetén a rekurzív számítással történik a leállási idő számítása, az aznapi megelőző erőművek leállással töltött idejének hozzáadásával. A kezdési időt és elmúlt napok kiesését nem kell

ismételten figyelembe venni, mivel azt már tartalmazza az első erőmű, így az összes többi is tartalmazni fogja.

$$ha\ 1 < i < erőműszám_j\ és\ x_{j_i} = e_k$$

$$t_k = jav_k + dt\left(e_k, x_{j_{i-1}}\right) + t_{x_{j_{i-1}}}$$

$$táv += d\left(x_{j_{i-1}}, e_k\right)$$

$$idő_j += jav_k + dt\left(e_k, x_{j_{i-1}}\right)$$

Az utolsó erőmű esetén csak annyi változás van, hogy a kikötőbe visszatérést is bele kell számítani a megtett távolságba és az eltelt időbe is.

$$ha\ i = erőműszám_j\ és\ x_{j_i} = e_k$$

$$t_k = jav_k + dt\left(e_k, x_{j_{i-1}}\right) + t_{x_{j_{i-1}}}$$

$$táv += d\left(x_{j_{i-1}}, e_k\right) + d(e_k, K)$$

$$idő_j += jav_k + dt\left(e_k, x_{j_{i-1}}\right) + dt(e_k, K)$$

Célfüggvény

Mivel itt t_i nem az i.-nek meglátogatott, hanem magát az i. erőművet jelöli, ezért az i. haszonnal kell beszorozni.

$$\sum_{i=1}^{n}(t_i \cdot haszon_i) + táv \cdot üzemanyagköltség$$

7. Használt szoftverek

Széles választék van optimalizáló eszközökből, én ezek közül négyet próbáltam ki. Mivel fekete doboz optimalizálásra használtam őket, így csak a user interfészük érdekelt. Az alábbiakban összehasonlítom a választott négy szoftvert, felhasználói szempontból. A használt eszközök általános optimalizálók, ezért nem rendelkeznek specifikus bementekkel problématípusra.

7.1 MATLAB Ga()

A MATLAB több beépített optimalizáló függvényt is tartalmaz, ezek közül egy a ga()[5], azaz a generic algorithm. A MATLABOT egyetemi licenszen keresztül használtam, telepítése sok helyet foglal és sok időt vesz igénybe, lévén általános matematikaszoftver. A következőkben jellemzem a programot felhasználóként, megemlítek pár általam hibának minősített dolgot, ami kevésbé intenzív és más feladattípusú használat esetén nem merült volna fel.

Felület

Kényelmes, elég programozni, a fordítást, debugot, futtatást támogatja.

Lehet benne a mapparendszert szerkeszteni, csak futtatás során szükséges a megfelelő mappában állni, ha egy fájlt csak olvasni akarok, megnyitja, de nem változik meg az aktuális fordítható mappa helye a megnyitott fájléra.

Ha valamilyen könyvtárcsomag nincs telepítve, a hibaüzenetre kattintva azonnal lehet telepíteni, automatikusan bezárja a MATLABot, majd ugyanott megnyitja a munkamenetet a sikeres telepítés után, és felajánlja a telepített csomag dokumentációját megtekintésre.

A hibaüzenetek hierarchikusak a hívási sor szerint, a hibás sort is meghivatkozza, de azon belül már nehéz eldönteni, pontosan mivel van gondja, mivel karakterindex nincs, és a hibaüzentetek többsége általános: például tömb érvénytelen indexelése, de ha 6 tömb is van egy sorban, nem visz közelebb a megoldáshoz. Ráadásul a hibás változók értékét sem adja meg, és ilyenkor a workspace-t is defaultba helyezi, így onnan sem tudhatjuk meg, milyen értékek mellett következett be a hiba. (Persze break pointtal lehet step-ekkel haladni a baj forrása felé, de ez többszörös ciklusok esetén hosszadalmas, és ha idegességében az ember továbblép, kezdődhet elölről a folyamat, mert nem terminál a hibán a MATLAB, egyszerűen elszáll.)

Kihívások

Csak a scriptfájlokat lehet futtatni, de a futtatás gomb nem érhető el az futtatandó scripthez tartozó épp szerkesztett függvényfájlból. Időveszteség és kényelmetlenség.

Fordítás előtt nem mindig ment, néha még 2-3 alkalommal a módosítás után is a régi scriptet fordítja, sőt, a break pointok helyét sem frissíti. Ez megoldható, ha az ember figyel rá, hogy fordítás előtt manuálisan minden fájlt elmentsen.

Az egyező változó nevekkel több problémám is volt. A különböző scriptekben ugyanúgy neveztem el egy file elérési útvonalát tároló változóját, egy ideig működött a beolvasás, majd egyszer csak tetszőleges régebbi elérési útvonallal írta felül. Megoldásképp minden futtatás után töröltem a workspace-t összes változóját, de ez után is ugyan úgy fennállt a probléma. Mivel nem találtam mega probléma forrását, minden változónak globálisan (olyan értelemben, hogy minden valaha futtatott és futtatandó függvények között két gépújraindítás között) nevet adtam. Ilyenkor is megfigyeltem, hogy betölti a különböző változók értékét, számomra ismeretlen logika szerint választva meg a változót és

időpillanatot. Szerencsére a különböző változónév miatt ez nem vezet értékfelülcsapáshoz és hibás működéshez, csak workspacehely pazarlásához.

Néha nem engedi el a megnyitott fájlokat, így kívülről nem lehet hozzájuk férni, ezen sem a fileclose, sem a futtatásnak a leállítsa nem segít, csak a MATLAB újraindítása. Számomra nem determinisztikus módon megválasztva a nyitva hagyott fájlt: például egy futás alatt megnyitok 6 fájlt, néha futás után csak az első nem elérhető, és a többi igen, néha a negyedik stb.

Több különböző mappában való dolgozásra nem igazán alkalmas, mert minden futtatáskor a mapparendszert átteszi a futtatás helyére. Ha például egy mappában csak le akarjuk futtatni az adatgenerátort, aztán dolgoznánk tovább az algoritmus fájljain, ezt nem tehetjük meg, csak ha visszanavigálunk a munkakönyvtárunkba. Ez lassítja a munkát és roncsolja az idegeket.

Segítség

Könnyű az adódó problémákra megoldást keresni, jól van dokumentálva, különböző fórumok vannak rá, mintakódok. (Google kereséssel bármit meg lehet találni.)

A saját weblapjuk mintakódjai interaktívak, ha az olvasó nem ért egy változót, belekattintva a kódba elvisz a dokumentációjához, ezzel hatalmas könnyebbséget okoz a használóknak.

Algoritmus

Képes lineáris és nem lineáris kényszer kezelésére is; nem lineáris esetben az egyenlőtlenség egyik odalát 0-ra kell rendezni, de természetesen ez semmilyen használatbeli nehézséget nem okoz, csak egy követelménye az algoritmusnak. Többek között meg lehet adni kötelezően egész értékű változókat, melyek tényleg csak egész értékeket fognak felvenni, így lehet velük indexelni. A kényszerek megszegésének toleranciáját is meg lehet adni, de a tesztelés alapján kiderült, hogy ezt a korlátozást nem tartja be. Lehet korlátozni a maximális generációk számát és a futási időt, ez hasznos hosszú futtatások esetén, amikor fontosabb, hogy legyen eredmény, mint hogy optimális legyen.

Kimenetként többek között megkapjuk a változót, a költségfüggvény értékét a változók helyén, és a maximumkényszerszegést.

7.2 NOMAD OPTI

A NOMAD (Nonlinear Optimization by Mesh Adaptive Direct Search) [6] egy optimalizálásra specializálódott ingyenesen használható toolbox. MATLAB támogatással elérhető, így az egész modellt meg lehet adni MATLAB-ban, és az optimalizáló hívása is MATLAB-ban történik. A fájlok lehúzása után MATLAB-ból kell futtatni a telepítőt. Ennek végeztével felvisz a honlap példaoldalára, ami egy meglepő és hasznos tulajdonsága. A hibaüzenetei könnyebben értelmezhetőek, mint a MATLAB saját hibaüzenetei.

Kihívások

Van benne maximális futásiidő-beállítás, ami default 1000 másodperc (~16,6 perc), én beállítottam 3,6 percre, de mindkét esetben volt olyan, hogy egy eset 4,5, 5, 7 órán keresztül futott.

Nem lehet leállítani a futtatást. Ugyan a MATLAB-ban van lehetőség a futás leállítására, de a NOMAD csak szüneteltetni engedi magát, bezárni se engedi a MATLAB-ot, szerencsére a MALTAB-nak van beépítve szoftveres hard reset-je, ami kétszeres rákérdezés után leállítja a MATLAB-ot.

Segítség

Saját wikijük van, példákkal, mind matematikai, mind kódsoros leírásokkal.

A wikin nincsen problémakezelés, csak az adott egyszerű minták vannak, fel sem merül, hogy valami nem működik.

A minta kódok nem hogy nem interaktívak, de a használt kulcsszavak egy része meg sincs említve a kód körül található magyarázó részben. (Ennyire nem lenne szükséges fekete dobozként kezelni, hogy még az interface-t is dobozuljuk.)

Több kódban van „info" visszatérési értéke az optimalizálónak, melyről annyit írnak, hogy *„info a structure with solver run information"*[7], de a pontos dokumentációja sehol sem található.

Alogritmus

A ga()-hoz hasonló be- és kimenettel rendelkezik. Különbségek például a következők: itt a nem lineáris kényszernek is szabadon meg lehet adni mindkét oldalát. A változók nem csak egész, hanem bináris értékűek is lehetnek. Kötelezően meg kell adni tippelt kezdő értéket.

A költségfüggvényem épít arra, hogy az egész kényszer be van tartva, és indexként lehet használni az értékeket. Az egész kényszer toleranciáját 0-ra nem lehet venni (ellentétben a dokumentációval), így csak kicsire vettem, így is kellett élnem a kerekítéssel, mivel a MATLAB az 1.0000-át nem int-nek érzékeli, így nem lehet vele indexelni.

7.3 DFL

A DFL (Derivative-Free Library) [8] egy ingyen használható egyszerű optimalizáló. Több lehetőséget is ad a felhasználónak, C, Phyton, FORTRAN és MATLAB közül választhatunk, így nem kell egy új, specifikus nyelvet megtanulni. Telepítéshez csak a szükséges nyelv kódját kell letölteni, MATLAB esetén csak egy függvényfájt kell betenni a megfelelő saját függvényeink mellé, de lehetőségünk van előre elkészített példakódokat is letölteni.

Segítség:

A dokumentációja inkább az elméletet magyarázza, gyakorlati példák helyett integrálok és deriválok vannak benne. Interneten nem nehéz bármit találni róla. Mivel az egész algoritmus forráskódja látható, nincs is szükség keresésre, az algoritmus képességei pontosan láthatóak.

Algoritmus

Megadhatjuk a minimalizálandó függvényt, alsó és felső korlátot a változóknak, maximális iterációszámot. De semmiféle kényszert nem, így alkalmatlan a feladat elvégzésére.

7.4 APM

Az APM [8] is egy ingyenes optimalizáló, mely szerveren fut. Lehetőség van a modellt weblapjukról feltölteni a szerverükre, vagy MATLAB-os függvényeik és függőségeik projekt mappához adása után MATLAB függvényük hívásával felcsatlakozik a szerverükre, és ott végzi az optimalizálást.

Felület

Az optimalizáló függvény MATLAB-ban futtatható, de a bemenetet egy saját nyelven írt (.apm) fájlban kell neki beadni, melynek sintaxisa nem dokumentált, csak a legtriviálisabb műveletekre képes, mátrixot deklarálni is csak úgy tud, ha soronként indexelve egyesével adjuk meg az elemeket.

A nyelv használhatatlansága miatt C#-ban kódgenerátor írtam hozzá.

Segítség

A NOMAD-hoz hasonló wiki oldal, de inkább csak a terméket reklámozzák, és vagy csak a triviális funkciókat mutatják meg, vagy valóban nem is létezik több.

A hibaüzenetek érdekesek: van egy szervertől jött MATLAB-os error, de a valós hibákat a konzolra írja sajátos struktúrájú üzenet formájában. Ráadásul a hibákat nem kódsorsorrendben találja meg, nem is mindig determinisztikusan, ami nagyban megnehezíti a javítást, és rengeteg felesleges munkát eredményez. A hiba helyét sem írja ki, csak kiemel pár karaktert a sorból, és kiírja, hogy error.

Konklúzió

Az alacsony dokumentáltság, a syntax check- teljes hiánya, a nem determinisztikus hibák és nem felhasználóbarát hibaüzentek miatt végül–számos próbálkozás után–eltekintettem a használatától.

A webes szerverszolgáltatás nem futtatta sem az én kódomat, sem a példakódot, időtúllépés miatt a kapcsolat mindkét esetben lezárult.

8. Modellek megvalósítása

A használt eszközöknek meg kell adni a változók számát, egy ekkora elemszámú vektorban fogják megadni a választ. Az egyes változóinkat ezt a vektor indexelve érhetjük el. Mivel nem használhatunk változóneveket, fokozottan figyelnünk kell az indexelésünk szemantikájára, hogy jól használjuk a költségfüggvényben a változóinkat, és hogy megfelelően értelmezzük a kapott megoldást.

Minden változóra meg kell adni, hogy milyen határok között vehet fel értéket, ga() esetében lehetőség van megadni az egész értékkel rendelkező változókat, NOMAD esetében ezen kívül a binárisakat is megadhatjuk.

Meg kell adnunk a költségfüggvényt, aminek segítségével az optimalizáló ki tudja értékelni az egyes változó-érték hozzárendelések „jóságát". Ebben tetszőleges műveleteket végezhetünk a változóinkkal és egyéb paramétereinkkel, a lényeg, hogy egyetlen szám legyen a visszatérése.

A kényszerek, melyekben leírjuk, hogy milyen tulajdonságokat várunk el az eredménytől (például, hogy minden erőműhöz jusson el a hajó). Egy kényszernek két oldala van, az egyik tartalmazza a változókból álló függvényt, a másik oldal pedig egy konstanst, ami relációban áll a változók függvényével. Ilyen kényszerekből többet is adhatunk meg, ez esetben egy mátrixba kell rendezni őket, amelynek minden sora egy kényszer bal oldala, és egy másik mátrixban, sor-sorrend helyesen tároljuk a hozzájuk tartozó konstansokat. A kényszerek lehetnek lineáris vagy nem lineáris függvényei a változóknak.

Nem lineáris kényszer esetén az összes változóval szabadon bánhatunk, tetszőleg függvényt kreálhatunk belőlük, majd solve() esetén meg kell adnunk egy számot, és hogy ezzel milyen relációban legyen a kifejezésünk. Ga() esetében csak ≤ 0 kényszerre van lehetőségünk, de átrendezéssel és negálással tetszőleges kényszert összerakhatunk.

Lineáris kényszer esetén minden változóra meg kell adnunk, hogy hányszorosan szerepel a kényszerben (0-ha nem, -ha ki kell vonni, + ha hozzá kell adni valahányszor), de ennél többet nem variálhatunk a függvényen. Itt a ga() is megenged bármilyen konstansot a jobb oldalon, hiszen a bal oldalon csak a változók hányszoros tartalmazását tudjuk megadni.

A modellekből háromfajta optimalizálásra alkalmas kényszerrendszert készítettem, és egy negyedik, heurisztikus megoldást is létrehoztam, viszonyítás végett. Ezeket mutatom most be. A Munkaidő figyelembevétele esetet nem valósítottam meg egy eszközzel sem, a Több csapat egy hajóval esetet pedig csak a heurisztikus és nem lineáris módszerrel, ezen kívül a Több hajó ütemezése eset helyett csak a Több hajó ütemezése benzinköltséggel esetet valósítottam meg, mivel az csak egy paraméterrel több.

8.1 Ga NonLinear

Elsőnek a MATLAB Genetic Algorithmot használtam nem lineáris kényszerekkel.

Minden erőművet pontosan egyszer látogatunk meg

x változó egy erőműszámú (n) hosszú vektor, melynek $x(i)$ elem megadja, hogy i.-nek melyik erőművet látogatjuk meg. Mindegyik erőmű rendelkezik egy egyedi sorszámmal, ami azonosítja.

Nem szerepelhet kétszer ugyanaz az erőmű kifejezése egyenlőtlenséggel:

Kiindulás: x minden elemének különböznie kell, azaz:

$$abs\big(x(i) - x(j)\big) \geq 1, i \neq j$$

Mivel a MATLAB nem enged meg csak ≤ 0 relációt, negálás és átrendezés után:

$$-abs\big(x(i)-x(j)\big)+1\leq 0, i\neq j$$

A fenti kényszert minden lehetséges párosra ki kell értékelnünk, ehhez $(n-1)+(n-2)+\cdots 1$ összehasonlítás szükséges, azaz $\frac{n\cdot(n-1)}{2}$

Négy erőmű esetében az alábbi mátrixot kell megszorozni x vektorral, hogy megkapjuk az össze lehetséges kombinációt.

$$\begin{bmatrix}1 & -1 & 0 & 0\\ 1 & 0 & -1 & 0\\ 1 & 0 & 0 & -1\\ 0 & 1 & -1 & 0\\ 0 & 1 & 0 & -1\\ 0 & 0 & 1 & -1\end{bmatrix}$$

Ez a kényszer automatikusan generálja azt is, hogy minden erőműnek legalább egyszer szerepelnie kell, mivel x értékei csak $[1,n]$ tartományból kerülhetnek ki, és különbözniük kell egymástól.

Költségszámítás

A költségfüggvény számolása sem triviális kérdés. Bevezettem egy vektort, aminek első eleme a kikötő-első erőmű távolsága, a második az első-második erőmű, és így tovább. A Heurisztika halála alfejezetben tárgyaltaknak megfelelően a kikötő-első erőmű távolság minden erőműnek kiesést fog okozni, azaz n-szereződik az ott szerzett veszteség (amíg minden erőmű azonosan termel). A súlyok hozzáadásához készítettem egy súlyzó mátrixot:

$$[n \quad n-1 \quad n-2 \quad \ldots \quad 1 \quad 0]$$

A 0 szerepe: az utolsó erőműtől a kikötőig való visszatérés alatt már egyetlen erőműnek sem növeljük a kiesési idejét.

Költségszámítás különböző profittal

A fenti távolságvektort itt is fel lehet használni, de itt már különböző profittal rendelkeznek az egyes erőművek. Ennek kezelésére létrehoztam egy vektort, ami minden erőműre tartalmazza annak leállási idejét, így végezetül csak be kell szorozni a saját profitjával.

A távolságokat tartalmazó vektort:

$$distance[n+1]=\big[d(K,e_{s_1}) \quad d(e_{s_1},e_{s_2}) \quad d(e_{s_2},e_{s_3}) \quad \ldots \quad d(e_{s_{n-1}},e_{s_n})\big]$$

Az egyes erőművek veszteségét tartalmazó vektor:

$$loss(i)=loss(i-1)+\frac{distance(i)}{1000\cdot v}+fix(x(i))$$

Azaz úgy kapjuk meg az i.-nek meglátogatott erőmű kiesési idejét, hogy hozzáadjuk az előtte lévő meglátogatásig eltelt időt és a közte és az előző állomás között lévő távolság megtételéhez szükséges időt, és még hozzáadjuk az aktuális erőmű megszereléséhez szükséges időt.

Több hajó

Itt x egy mátrix, aminek első sora sorra tartalmazza az első hajó állomásait, a második a második hajó állomásait, és így tovább. A modellben leírtakkal ellentétben itt minden hajónak saját sebessége van, saját benzinfogyasztása.

Mivel van integer kényszerem (minden változónak annak kell lennie), így a ga() nem enged meg $=$ kényszert, de azt helyettesítem egy $a \leq 0$ $és$ $egy - a \leq 0$ kényszerrel. A zérust, azaz a hajó nem áll meg erőműnél szimbólumot 0 helyettesíti.

A mátrixos, szummás ne legyen egyező vizsgálat helyett egyszerűen MATLAB függvény segítségével megnézem, hogy szerepel-e minden állomás, és hogy egynél többször ne szerepeljen. A megállókat tartalmazó tömböt állomásszám szerint növekvő sorrendbe rendezem, és kivonom a [0 0 … 0, 1 2 3 … n] mátrixból. 0-ből pontosan $H \cdot n - n$ -nek kell lennie. (Mivel csak n helyen állhat meg összesen, de minden hajónak felvettem n megállót.) Minden 0 után már csak 0 jöhessen, azaz ne térjen vissza a kikötőbe, és onnan menjen erőműhöz.

Az x tömböt hajónként vizsgálom. Ha még nem értem el a következő hajóig, és 0-át tapasztalok, az azt jelenti, hogy az a hajó már nem mehet más állomásra, tehát utána is csupa nullát kell tapasztalnom. Így készítek egy tömböt az állomássorrendek átmásolásával, és a 0-k után csupa 0val. Ezt vonom ki az eredeti X-ből, ha nincs köztük eltérés, akkor az is a szabályok szerint van kitöltve.

Több csapat

A költségszámítás minden erőműre kétszer történik, egyszer az odajutás, egyszer pedig a megjavítás-visszatéréskor (lásd. 6.6. fejezet). Hogy el tudjam dönteni, hogy hányadszorra szerepel egy erőmű, létrehoztam egy 2n hosszú vektort. Megkerestem az ütemtervben minden erőművet, az első előfordulását 1-gyel jelötem, a másodikat -1-gyel, és ezt használtam ki a költségszámításánál.

Itt a kényszerünkben nem minden erőmű pontosan egyszeri, hanem kétszeri meglátogatása szerepel, hisz minden csapatot fel is kell venni az erőművektől. Ezt akartam kihasználni az ellenőrző vektor létrehozásakor, így minden erőműnek kerestem az előfordulását. A ga() megengedi a kényszerek megszegését mind keresések közben, mind végeredményeknél, ezért nem szerepelt minden erőmű az ütemtervben. Így azok keresése 0-át adott vissza, így 0-val indexeltem vektort, amire a MATLAB kivételt dob. A kényszerek maximális megsértésének engedélyezett értékét be lehet állítani egészen kicsit értékre is, ezzel csak annyit érünk el, hogy a MATLAB warningot tesz az olyan eredmények mellé, amik ezt túllépik, de nem kényszeríti ki tényleges betartásukat.

Az állapotjelző helyett egy „bool” tömb van, amiben nem az erőműhöz rendelve, hanem annak minden erőfordulására el van tárolva, hogy első-e vagy második. (1: első, -1: második) Ennek segítségével az egymás után jövő elsők száma nem haladhatja meg a csapatok maximális számát, minden második látogatás „kiüt” egy elsőt. Az lényegtelen, hogy melyik melyiket, elég, hogy számban megegyezzenek. (Helyes algoritmusműködés mellett, ha egy erőmű másodszor szerepelőnek van feltüntetve, akkor biztos, hogy már egyszer szerepelt, tehát az összegből való levonással helyes műveletet végzünk el.)

8.2 NOMAD

Áttérés lineáris kényszerekre

Először hasonló, nem lineáris modelleket alkalmaztam, mint a fenti részben. A probléma ott volt, hogy a NOMAD elvár egy tippelt kezdőértéket, de a futások során nem lépett ki soha ebből az értékből. Ennek oka a következő lehetett: a költségszámításoknál kihasználtam a változók egész korlátozását, ezzel indexeltem meg a tömböt, ahol az erőművekre vonatkozó információkat tároltam. A NOMAD a MATLAB-bal ellentétben nem tudja betartani ezt a korlátozást, ugyan az egész értéktől való eltérés toleranciáját be lehet állítani akár 1e-10-re, de attól ez még nem lesz alkalmas vektorindexelésre. Így egésszé kerekítést kellett alkalmaznom, így a kis módosításoknak semmilyen hatását nem érezte, mivel mindent a legközelebbi egészre kerekített, a lépegetések eredménye az lett, hogy nem érdemes kimozdulnia. Másrészről a korlátozások túl szigorúak voltak, nem tudott úgy egy elemet megváltoztatni a vektorban, hogy a többit ne kelljen megváltoztatnia, mivel egy sorrendről beszélünk, ahol az értékek nem függetlenek egymásától. Így kis módosításokkal nem ment semmire.

Ennek megoldására az egész változórendszert át kellett alakítani. X már nem egy vektor, ami a változók sorrendjét tartalmazó $1 \dots n$ közötti értékeket vehet fel. Helyette egy $n \cdot n$-es mátrix, melynek sorai jelentik az erőművet, oszlopai az időpillanatokat, ha egy metszetben 1-es van, akkor abban az időpillanatban az adott erőműnél van.

	$id\H{o}_1$	$id\H{o}_2$	$id\H{o}_3$	…	$id\H{o}_n$
$er\H{o}m\H{u}_1$	1	0	0	…	0
$er\H{o}m\H{u}_2$	0	0	1	…	0
$er\H{o}m\H{u}_3$	0	0	0	…	0
…	…	…	…	…	…
$er\H{o}m\H{u}_n$	0	1	0	…	0

A fenti példában az elsőnek az első erőművet látogatjuk meg, másodszorra az n. erőművet, harmadszorra a második erőművet.

Új korlátozások: minden időpillanatban legyen valahol, és minden helyen legyen legalább egyszer. Ezek a kényszerek már lineárisan is megfogalmazhatóak, mivel annyit várunk el a változó mátrixtól, hogy minden sorában és oszlopában legyen legalább egy egyes. Mivel a NOMAD egy vektort ad vissza változóként, a mátrixot oszloponként elhelyezzük egy vektorban. Lineáris kényszerrel, azaz a minden egyes változó értékét egy konstanssal beszorozva akarjuk megkapni a felső kényszereket. Ehhez egy $2 \cdot n$ db $n \cdot n$-es kényszerre lesz szükségünk.

A kényszer mátrix:

	t_1					t_2					…	t_n				
	e_1	e_2	e_3	…	e_n	e_1	e_2	e_3	…	e_n	…	e_1	e_2	e_3	…	e_n
1	**-1**	**0**	**0**	…	**0**	**-1**	**0**	**0**	…	**0**	…	**-1**	**0**	**0**	…	**0**
2	**0**	**-1**	**0**	…	**0**	**0**	**-1**	**0**	…	**0**	…	**0**	**-1**	**0**	…	**0**
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
n	**0**	**0**	**0**	…	**-1**	**0**	**0**	**0**	…	**-1**	…	**0**	**0**	**0**	…	**-1**
1	*-1*	*-1*	*-1*	…	*-1*	*0*	*0*	*0*	…	*0*	…	*0*	*0*	*0*	…	*0*
2	*0*	*0*	*0*	…	*0*	*-1*	*-1*	*-1*	…	*-1*	…	*0*	*0*	*0*	…	*0*
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
n	*0*	*0*	*0*	…	*0*	*0*	*0*	*0*	…	*0*	…	*-1*	*-1*	*-1*	…	*-1*

A vastagon szedett kényszerek vonatkoznak a minden erőmű legalább egyszerre, a dőlttel a minden időpillanatban legalább egy helyenre. A kényszer teljesül, ha $\bar{\bar{A}} \cdot \bar{x} \leq -1$

Tehát több helyen lehet egy időben, és többször is láthat egy erőművet, így az egyes változókat egymástól függetlenül is lehet változtatni egy bizonyos határig. Nekünk viszont szükségünk van egy olyan végső megoldásra, ahol mindenkit pontosan egyszer látogatunk meg, hisz nincs érteleme már megjavított erőművet ismételten megjavítani. Ezt nem kényszerbe kódolva tartatjuk be, hanem felhasználva azt, hogy az optimális útiköltség úgy kapható meg, ha feleslegesen nem látogat helyeket, amiknek plusz költségük lenne. Tehát amennyiben optimális megoldást kapunk, az helyes is. Annyi hiba lehet, hogy az utolsó pillanatban elmegy egy erőműhöz, de közben ott is marad az n-1-dik erőműnél, mivel a költséget az előző erőműtől vett távolság jelenti, tehát a helyben maradás ingyen van.

Több hajó

Itt már minden hajóra fel kell venni minden időpillanatra minden erőművet, hogy megtartsuk a fenti szabadságot a változók változtatásában.

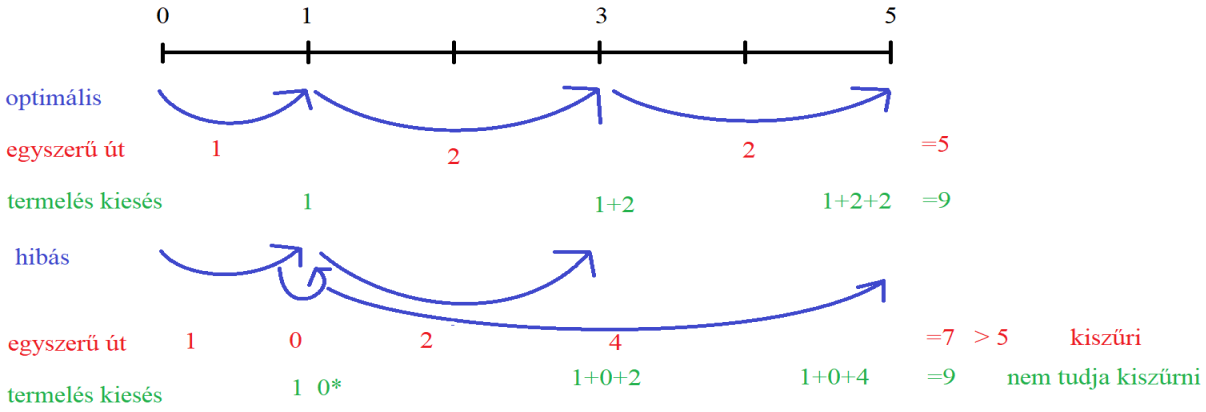
	$idő_1$				$idő_2$				…	$idő_n$		
	$hajó_1$	$hajó_2$	…	$hajó_n$	$hajó_1$	…	$hajó_n$		…	$hajó_1$	…	$hajó_n$
$erőmű_1$			…			…			…		…	
$erőmű_2$			…			…			…		…	
$erőmű_3$			…			…			…		…	
…	…	…	…	…	…	…	…	…	…	…	…	…
$erőmű_n$			…			…			…		…	

Probléma a kényszerek optimalizációval való betartatásával

A veszteség egy erőműre nem annyi idő, amíg az előtte lévőtől eljutunk odáig, hanem az addig megtett összes út, hiszen végig nem üzemel. Tehát az első út n*szeresen bünteti a többit, a 2. n-1szeresen, stb.

De ebben az estben nem igaz a fenti sejtés, jobb eredményeket kap az algoritmus, ha n*szeres szorzóra akár többször berakja a legkisebb utat, mintha oda tenne egy hosszabbat.

Példa:



Itt valójában eltelik egy óraütés, tehát a többi erőmű kiesésének növekednie kellene, de ezek az időpillanatok szimbólikusak, az útból számolódik a valós.

6. ábra Azonos termelési kiesés értelmes és értelmetlen úton

Megoldás:

Ha az az eset áll fenn, hogy a lépés költsége 0 (tehát nem ment át másik erőműhöz), büntetést számolok fel az algoritmusnak a 0 költségű út helyett. Ha túl nagy büntetés: nem mer ellépni, nem talál jó megoldást. Ha túl kicsi büntetés: nem tartja vissza.

Választásom: megnézem a még meg nem látogatott erőművek közül azt, amelyik a legközelebb van hozzá, és a távolságánál eggyel nagyobb étékkel büntetem, hogy jobbnak lássa inkább a legközelebbivel próbálkozni. (A volt-e már azért kell, mert különben kiszemelhetne magának egy kellemesen közeli bázist, ahová „olcsón visszajárhatna”, ha pozíciót kell változtatnia.)

Hogy megnézzem ezen módszer eredményre generált hatását 30 futásra, 3 erőművel teszteket futtattam.

12. táblázat:Penalty-val és nélkül eredmények az Alapesetben
**Az optimális költségűek közül mind értelmes válasz.*

Alapeset	Penalty nélkül		Penaltyval	
	Optimális költség	Értelemes válasz	Optimális költség	Értelemes válasz
Ga()	30	7	22	27*
Nomad	30	30	30	30

13. táblázat: Penalty-val és nélkül eredmények az Üzemanyagköltség figyelembevétele esetben
**Az optimális költségűek közül mind értelmes válasz.*
***A nem optimális értékek csak 0.003-mal haladták meg az optimálisat.*

Üzemanyagköltség figyelembevétele	Penalty nélkül		Penaltyval	
	Optimális költség	Értelemes válasz	Optimális költség	Értelemes válasz
Ga()	26**	24	27	28*
Nomad	30	30	30	30

14. táblázat: Penalty-val és nélkül eredmények a Változó termelés és javítási idő figyelembevétele esetben

Változó termelés és javítási idő figyelembevétele	Penalty nélkül		Penaltyval	
	Optimális költség	Értelemes válasz	Optimális költség	Értelemes válasz
Ga()	21	27	21	30
Nomad	30	30	30	30

A 14. táblázat eredményein az látszik, hogy annyira jelentős egy plusz erőmű többletköltsége, hogy nem szükséges a penalty kis esetszám mellett, de a nagy esetszámokra való tekintettel és a többi teszt alapján úgy döntöttem, hogy alkalmazom ezt a fajta rosszlépésbüntetést. A NOMAD kis eset szám mellett biztató eredményeket mutatott, de ahogy a 9.1 alfejezetből látszani fog, nagy eset-/ erőműszám mellett túl nagy a futási ideje.

8.3 Ga Linear

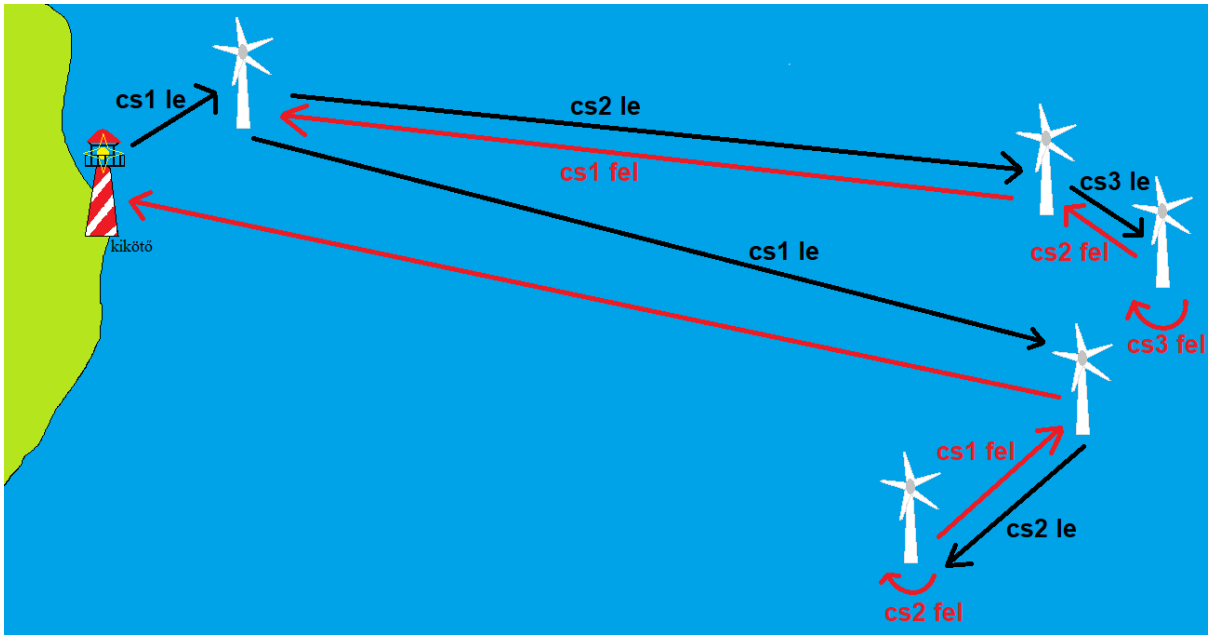
Majdnem identikus a NOMAD lineáris leírásával, csak a függvényhívás szintaxisából következően vannak apróbb eltérések.

8.4 Heurisztikus

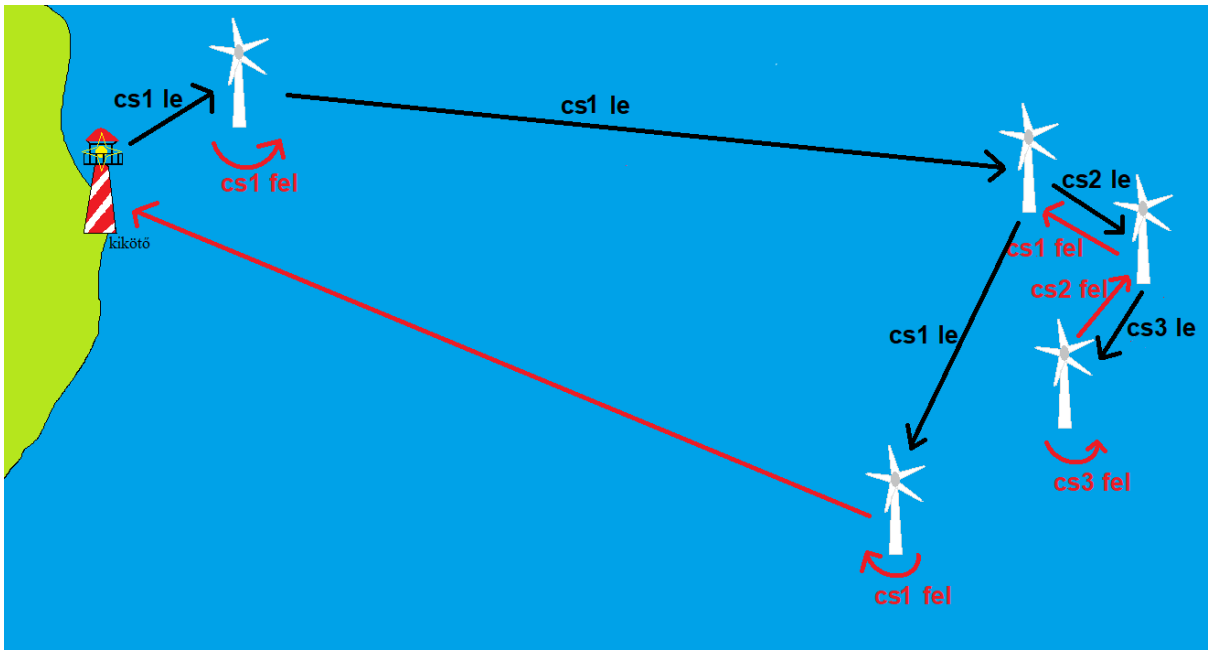
A heurisztika egyszerű: mindig a legközelebbi erőművet vesszük a következőnek.

Több hajó esetén egyszerűen mindig az adott hajóhoz legközelebb eső erőművet adom meg hajóknak, természetesen okosabb heurisztikát is lehetne alkalmazni, de a cél csak egy egyszerű megvalósítás volt, hogy a nulla erőfeszítés eredményét össze lehessen mérni a fenti modellekkel.

Több csapat esetén, amíg van csapat, megy a legközelebb eső erőművekhez, majd fordított sorrendben elindul begyűjteni őket, mindet felveszi, és egyszerre indul el újra helyre lerakni őket. Ez szemmel láthatóan nem a legjobb megoldás, míg az 6.6 fejezetben bemutatott modell nem él sorrendiségi megkötésekkel, szabadon cserélgetheti a változókat (természetesen az 6.6 fejezet kényszereinek betartásával). Tehát megteheti, hogy nem rak le minden csapatot, hanem megvárja egy közeli erőműnek a gyors megjavítását, majd minden csapattal együtt indul el egy távolabbi csoport megjavítására, ezzel megspórolva a messzi csoport és a közeli közti ingázás költségét. Az így kapott eltérést hivatott szemléltetni a 7. ábra és a 8. ábra, természetesen heurisztika által készített terv nem minden esetben rosszabb, ehhez az is kell, hogy az ábrákon látható legközelebbi erőművet gyorsan lehessen javítani, a többiek kiesése ne legyen nagyon jelentős az üzemanyagköltséghez képest stb., de lehetséges olyan paraméterfelállás, amikor ez megtörténik.



7. ábra: A heurisztika által létrehozott terv egy közeli és egy messzi farm esetén



8. ábra: Egy lehetséges terv a kötetlenebb optimalizáló által egy közeli és egy messzi farm esetén

Egy másik problémája lehet a heurisztikának, ha egy csapat hosszabb ideig dolgozik egy erőművön, mivel meg fogja várni ezen csapat elkészülését, ezzel megakadályozva, hogy a gyorsabb csapatok új munkálatokba kezdhessenek. Ez a probléma szintén nem áll fenn egy szabadon cserélgethető optimalizálónál.

A költségszámításhoz a nem lineáris esetben alkalmazott függvényeket használtam, mivel a változó mátrix szerkezetei tejesen megegyeznek.

9. Tesztek

A kapott megoldások nem mind voltak megfelelőek a várakozásoknak. Időnként az sem teljesült, hogy minden erőművet meglátogassunk, ennek oka, hogy egyik használt eszköz sem tartatja be szigorúan a kényszereket. És volt, amikor az sem teljesült, hogy egy erőművet egyszer látogassunk meg. Ga() nonlinear esetben ez szintén a kényszerek be nem tartásával magyarázható. Míg a lineáris esetben a modell engedte meg a nem betartást, és mivel nem találta meg az optimális megoldást, nem is tartódott e az implicit kényszert.

Összesen 100 különböző tesztet generáltam minden modellre a MATLAB beépített véletlenszám-generátorával. A paramétereket megadott határok között kellett kigenerálni, ezen határokat tartalmazza a 15. táblázat.

15. táblázat A tesztesetekben szereplő paraméterek lehetséges értékei

	Minimális érték	Maximális érték
Erőműszám	5	20
Hajó sebessége	40 km/h	70 km/h
Erőmű profitja	250 EUR	1000 EUR
Koordináták	0 m	50000 m
Benzinköltség	1 EUR/km	5 EUR/km
Javítási idő	1 h	10 h
Hajószám (ha nem egy)	2	5
Csapatszám (ha nem egy)	2	5

A teszteseteket erőműszám szerint növekvő sorrendben szerepelenek a diagrammok mindenkori vízszintes tengelyén.

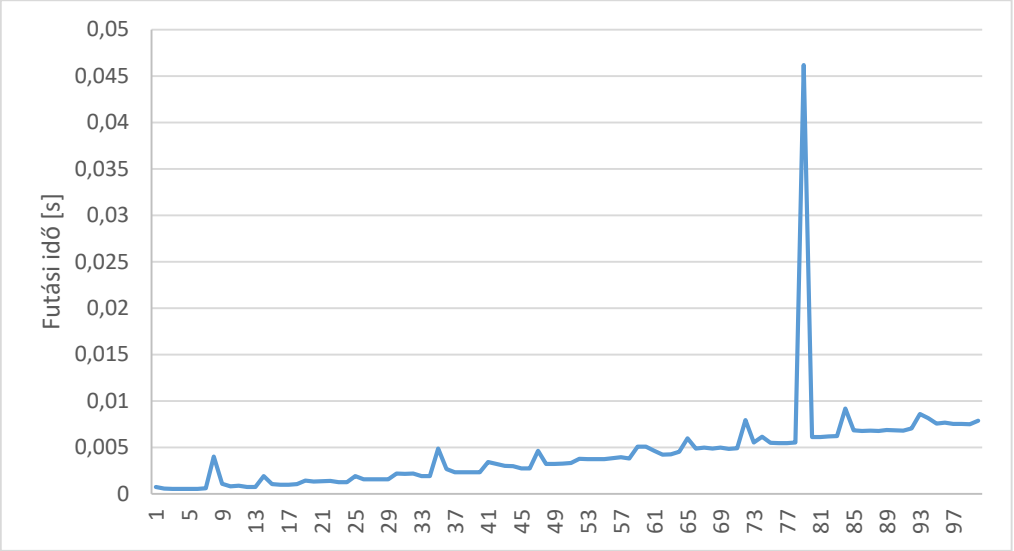
9.1 Futási idő

Bár a NOMAD dokumentációban az van, hogy max. 16 percig fut, voltak többórás futások. A max. futási időt lekorlátoztam 3,6 percre, de így is volt, ami órákig futott. Ezért NOMAD-os tesztesetekből csak a legkevesebb erőműszámmal rendelkező eseteket futtattam le.

A nem lineáris megoldás kellően gyorsan futott ahhoz, hogy egy tesztesetre 10-szer is lefuttassam, így ez esetben átlagos futási idők és átlagos veszteségeredmények vannak.

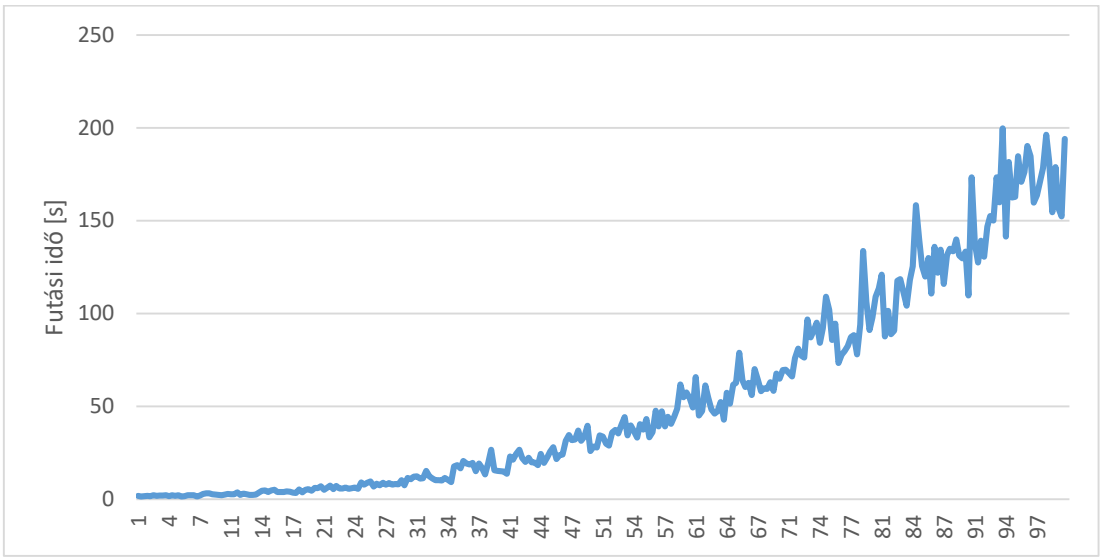
A mérésbe zaj került, ennek fő oka, hogy a várakozásaimat messze felülmúlták a futási idők. Így egész éjszakákra bekapcsolva kellett hagynom a gépet. Az első alkalom után jöttem rá, hogy nem kapcsoltam ki a Windows automatikus altatását időkorlát után, így a futási időkből kirívó, órás várási eredményeket kaptam. A másik zajt a Chrome megnyitása okozta, amit amennyiben lehetett, igyekeztem elkerülni, de amikor azt mégsem lehetett, akkor látványosan megdobta a futási időket és a ventilátor fordulatszámát.

Természetesen a leggyorsabban a heurisztika futott le, lévén, hogy neki nem kellett megoldások között keresgélnie, egyértelműen tudta minden lépésben, kit válasszon következőnek. Az idő egyenesen arányosan nő az erőműszám növekedésével ezen oknál fogva, ezt mutatja a 1. diagram.



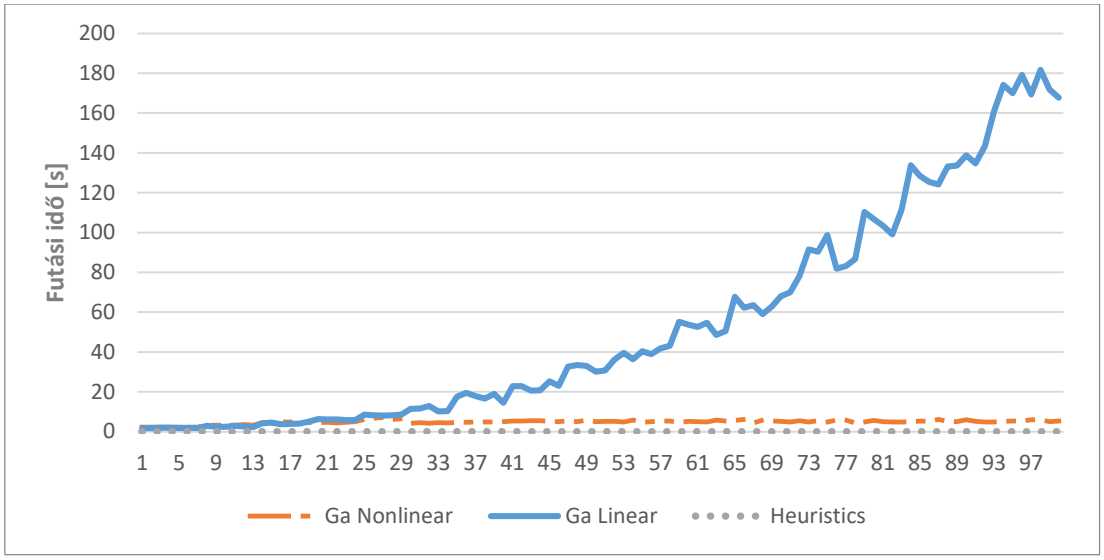
1. diagram Heurisztikus algoritmus futási ideje másodpercben az Alapesetben

A nem lineáris ga() futási ideje nagyjából egyenesen arányos változó számmal, még a lineáris kényszerekkel megfogalmazott modell futási ideje – a 2. diagram alapján – logaritmikusan aránylik az erőműszámokhoz. Ennek oka az lehet, hogy a nem lineáris megoldásban a megoldások között csak egy permutációját keressük az erőművek sorrendjének, míg a lineárisnál a változók négyzetével arányos mátrixban kell keresni optimális választást.



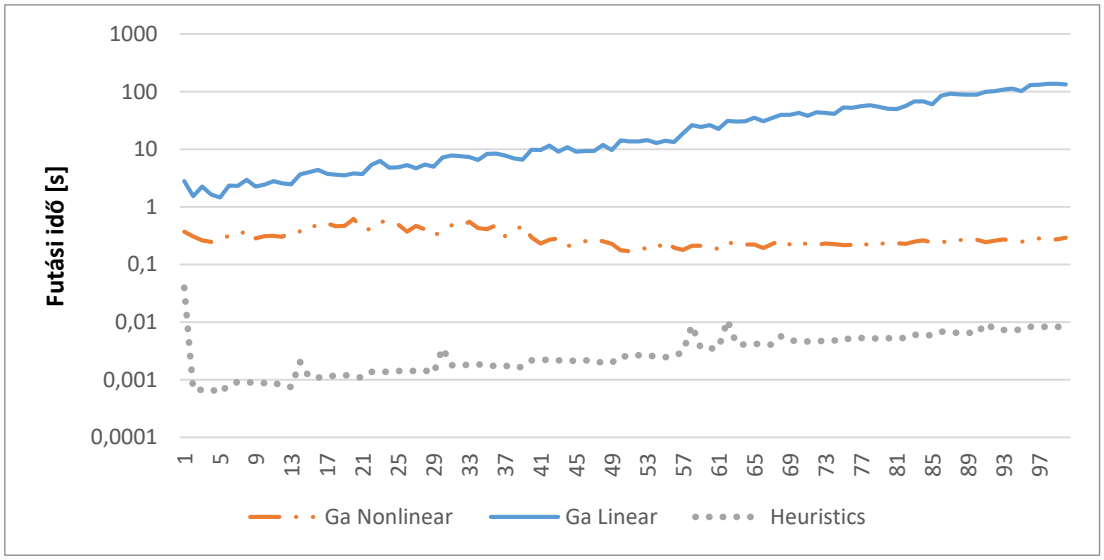
2. diagram Ga() lineáris algoritmus futási ideje másodpercben az Alapesetben

A NOMAD még az időben logaritmikusan növő lineáris megoldásnál is lassabban futott, így a csak kevesebb számú tesztesetet tudtam futtatni vele. A futási idő mértéke felveti a kérdést, hogy érdemes-e órákat várni egy ütemezésre, vagy ezalatt annyira jelentős termeléskiesés történik, hogy egy kevésbé megfontolt tervvel elkezdeni a javítási munkálatokat jobban megérné-e.



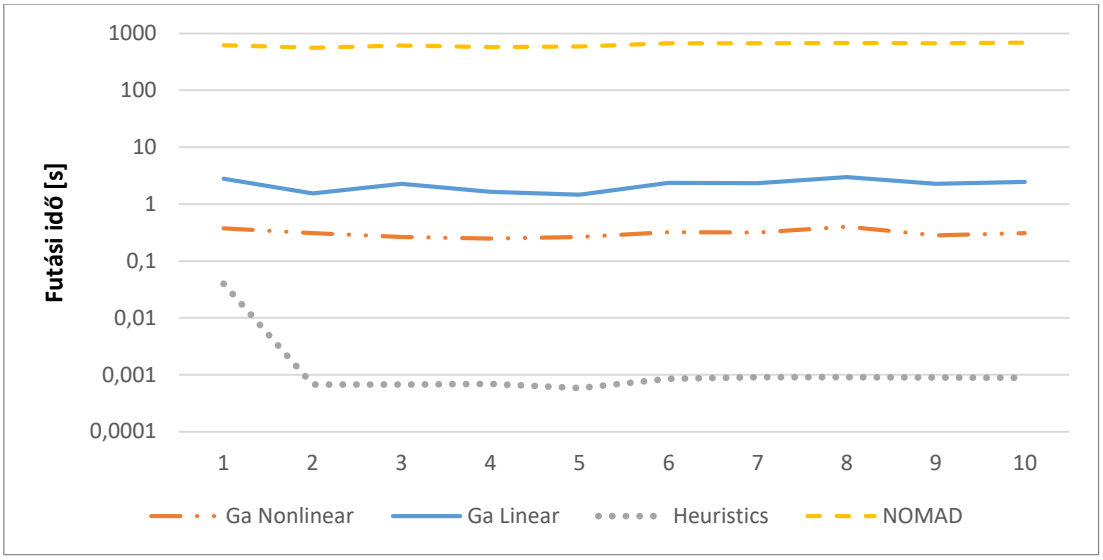
3. diagram Algoritmusok futási idejének összehasonlítása másodpercben az Alapesetben

Már egyszerű eseteknél is jelentősen lassan fut a lineáris megoldás, ahogy a 3. diagram mutatja. Bonyolultabb modellek estén a lineáris annyira lassan fut, hogy már csak logaritmikus skálán lehet összehasonlítani a másik kettő futási idejével, ezt szemlélteti a 4. diagram.



4. diagram Algoritmus futási idejének összehasonlítása másodpercben logaritmikus skálán Változó termelés és javítási idő figyelembevétele esetben

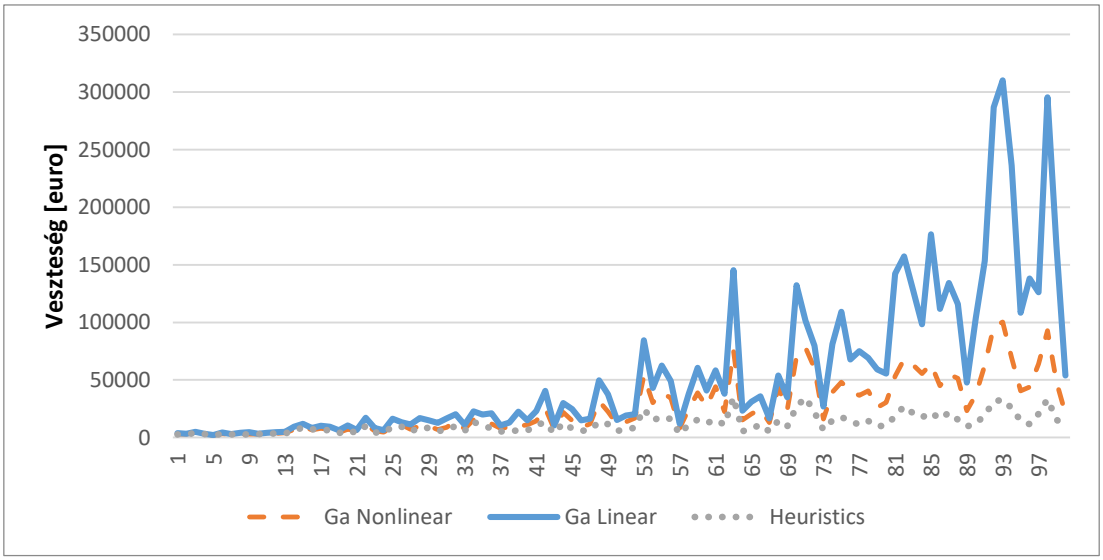
De mindezen eltérések nem is mérhetők a NOMAD-tól való időeltéréshez, ami a lassúságnak hihetetlen határait feszegeti, jól látszik az a 5. diagram, ami nagyságrendi különbségek miatt logaritmikus tengelybeosztást használ a futási idő mérésére.



5. diagram Algoritmusok összehasonlítása futási idő szerint másodpercben, logaritmikus skálán Változó termelés és javítási idő figyelembevétele esetben

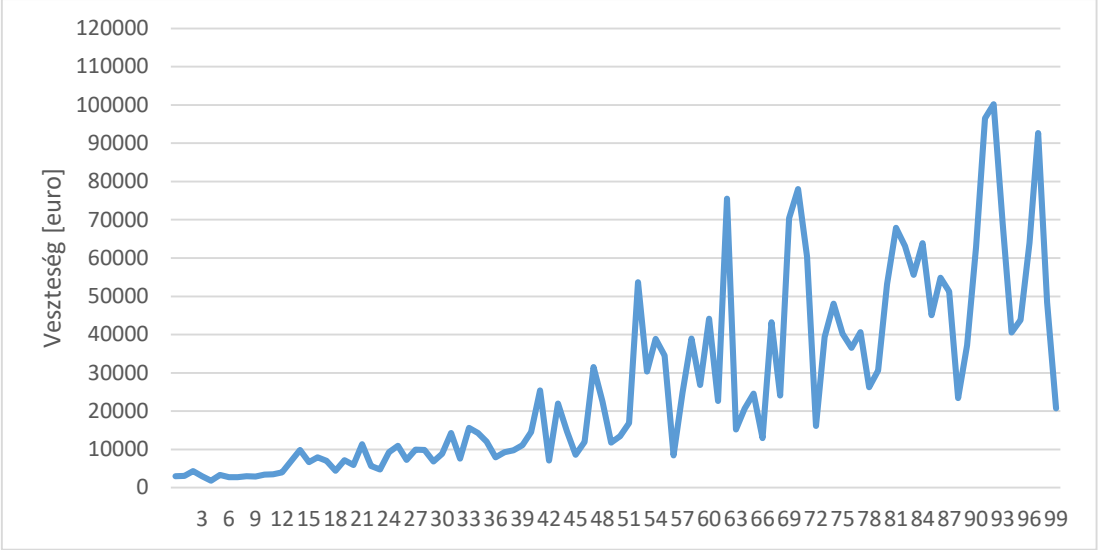
9.2 Veszteség

A 6. diagramról leolvasható, hogy a heurisztikus algoritmus jóval kisebb költségeket produkál, mind bármelyik másik algoritmus. De a feladatok karakterisztikáját jól megtalálja mindegyik algoritmus, nagyjából párhuzamosan adnak meg értékeket vélt optimumnak.

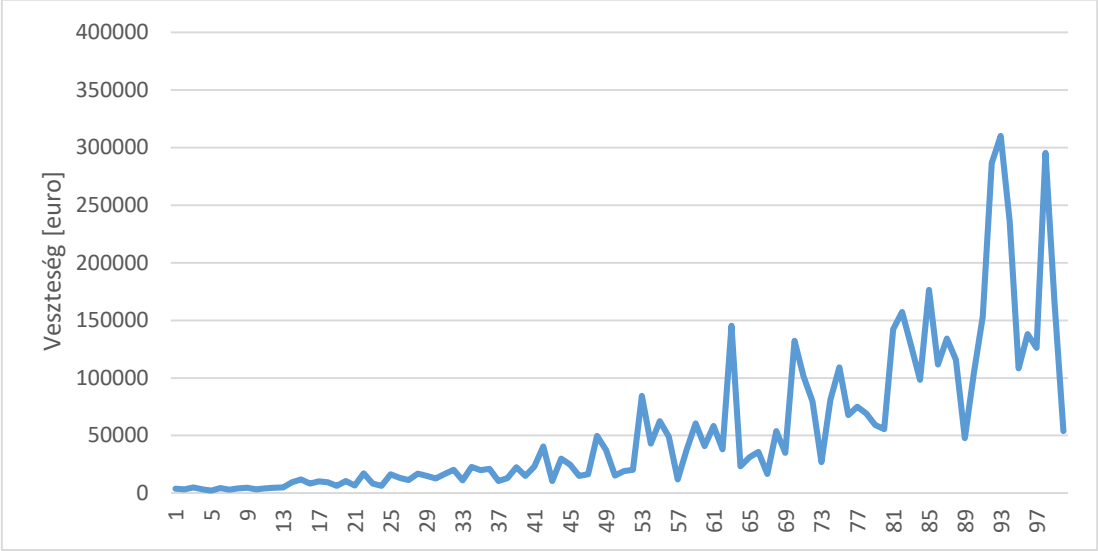


6. diagram Algoritmusok összehasonlítása veszteség szerint euróban, az Alapesetben

Az első tesztesetet még a lineáris ga()-val is többször lefuttattam, ez látszik a 8. diagramon. Ezene, és nem lineáris többszörös lefutásait ábrázoló 7. diagramon is az látszik, hogy ahogy nő az erőműszám, nő a szórásuk a vélt optimumoknak.

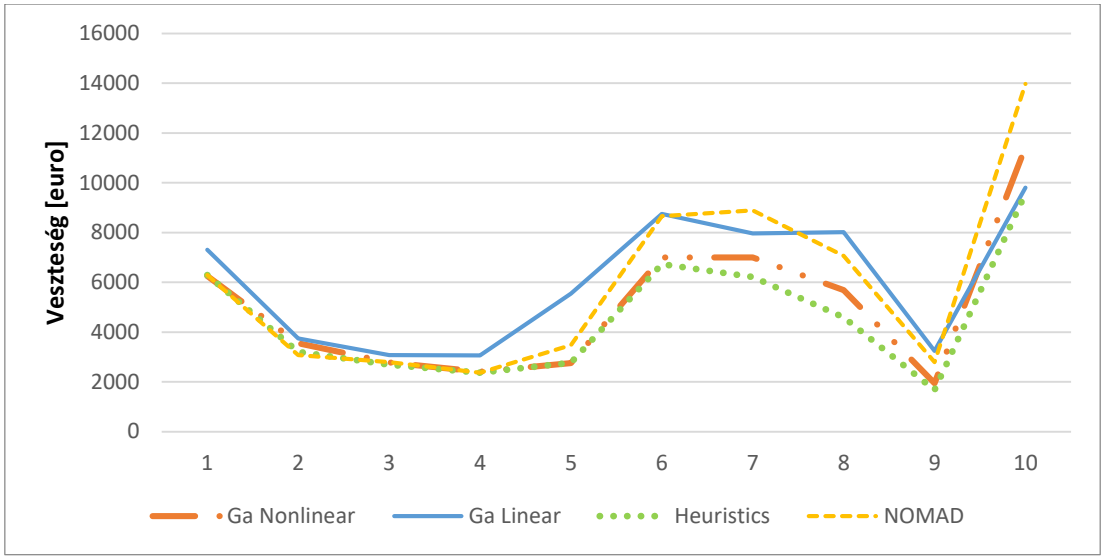


7. diagram Ga() nem lineáris átlagos vesztesége euróban az Alapesetben



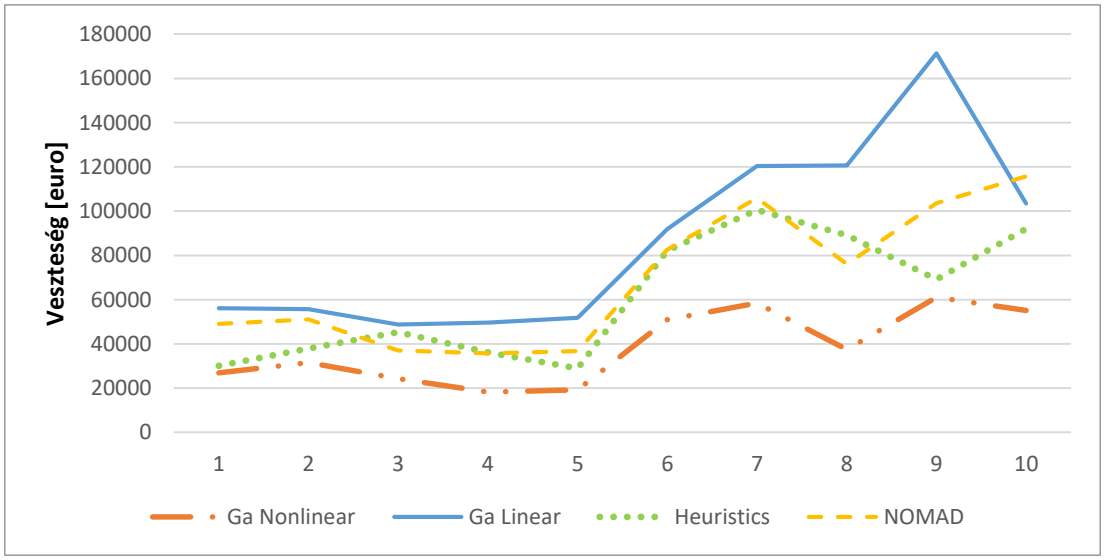
8. diagram Ga() lineáris átlagos vesztesége euróban az Alapesetben

Kis erőműszám esetén nem szignifikáns a különbség az eredmények között, és mivel a NOMAD-ot csak ilyen esetekre tudtam lefuttatni, nem derült ki, hogy jobban tudna-e teljesíteni, egy 10 esetes összehasonlítást mutat a 9. diagram.



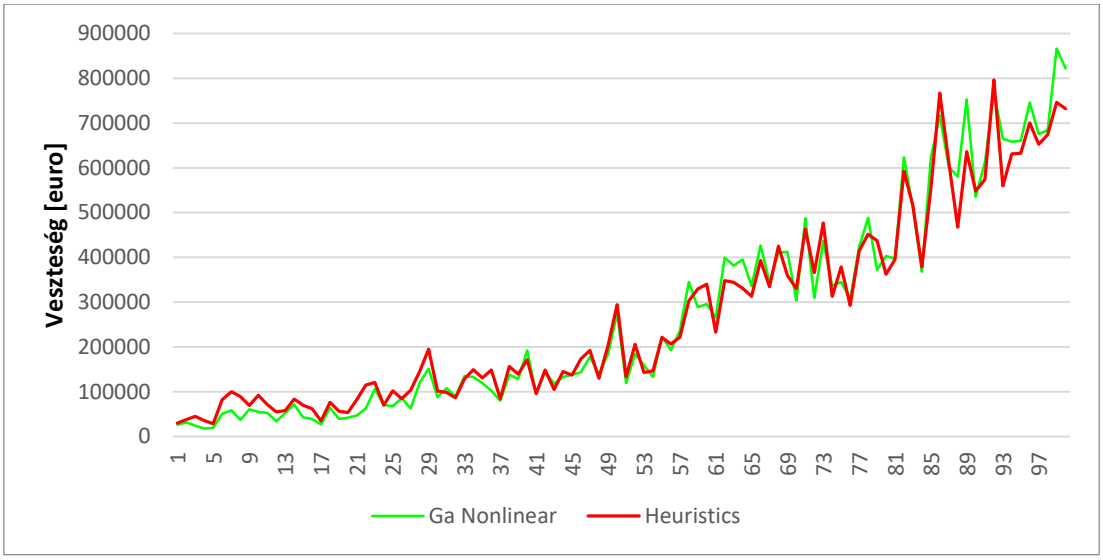
9. diagram Algoritmusok összehasonlítása veszteség szerint euróban, Üzemanyagköltség figyelembevétele esetben

Mivel a NOMAD-ot csak kis esetszámra tudtam futtatni, így „nagyobb felbontású" diagramot kaptam az első, alacsony erőmű számú esetről. A 10. diagramon jól látszik, hogy kis erőműszám mellett az egyszerű heurisztikát lehagyja a nem lineáris ga().



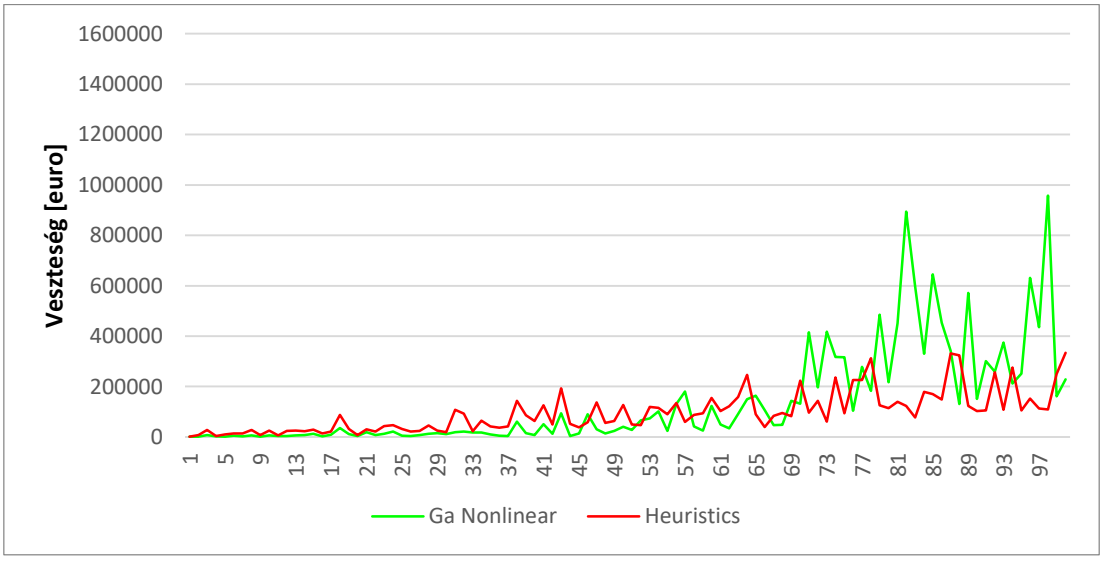
10. diagram: Algoritmusok összehasonlítása veszteség szerint euróban, Változó termelés és javítási idő figyelembevétele

Azonban sok erőműszámú, egyszerű esetekre elég lelombozó az eredmény, a heurisztika lehagyja az összes többi algoritmust. De változó erőműprofitot és javítási időt bevezetve a 11. diagramon láthatóan a nem lineáris kezdi megközelíteni a heurisztikát, mivel az csak a távolságokat nézi.



11. diagram A ga() nem lineáris és a heurisztikus összehasonlítása veszteség szerint euróban, Változó termelés és javítási idő figyelembevétele esetben

Több hajó bevezetésével a heurisztikát le is hagyja a nem lineáris, de az erőműszám növelésével ismételten a heurisztikus megoldással kapunk jobb eredményeket. Ezt szemlélteti a 12. diagram. A romlást az magyarázhatja, hogy egyre több permutációval kell próbálkoznia a generikus algoritmusnak, míg a heurisztika minden lépésben egyértelműen választ egy elemet. Fontos emlékezni, hogy a nem lineáris 10-szer próbálkozik minden esetnél, és ennek az átlaga van itt feltüntetve, ebből is előnye származik. Ezen kívül a heurisztika több hajóra nem a legokosabb, ennek bővebb kifejtése szerepel a 8.4 fejezetben.



12. diagram Ga() nem lienáris és heurisztikus összehasonlítása veszteségre euróban, Több hajó ütemezése esetben

10. Összegzés

Mint láthattuk, a szélerőmű-farmok napi működését rengeteg paraméter befolyásolja a valós világban, ami megnehezíti az őket leíró modell határainak meghúzását, és a már kiválasztott paraméterek valóságból történő leképezése sem egyértelmű. Bár modelljeimben a lehetséges paraméterek közül keveset vettem figyelembe, az azok közti függőségek és ráhatások is jelentősen növelték a költségszámítások komplexitását.

Az általános matematikával megadott modelleket át kellett fordítani az optimalizáló eszközök preferált nyelvére. Ez nem csak a képletek függvénybe írását foglalta magában, hanem szerkezetei változtatásokat is a modellekben. Ezen felül ugyan arra az estre ugyan azon kényszerekkel több féle változó szerkezetet is meg lehet adni, így kaptam a lineáris és nem lineáris modelljeimet.

Az intuitív feltételezésünkkel ellentétben már kis erőműszám esetén is óriási különbségek vannak a különböző tervek között, ez alátámasztja optimalizálás igényét. A teszt eredmények jól mutatják, hogy még az optimalizáló algoritmusok eredményei között is jelentős különbségek vannak, annak függvényében is, hogy hogyan alkotjuk meg a változó és a kényszerek szerkezetét.

Bár a heurisztikát egyik optimalizáló sem tudta hosszútávon legyőzni sem időben, sem költségben, a paraméter számok növelésével valószínűsíthetően egyre jobban teljesítenének a heurisztikához képest, így érdemes finomítani a felépítésüket. Ez abból látszik, hogy több hajó esetén a nem lineáris kényszerekkel dolgozó ga()-nak sikerült a heurisztikánál jobb tervet adnia. Általában is a nem lineáris szerepelt legjobban az optimalizálók közül, mivel a lineáris ga() a magas permutáció lehetőség miatt sok időt igényelt egy-egy lefutásához, és közel sem tudott olyan jó tervet adni, mint a nem lineáris változat.

A NOMAD értékelése lezáratlan maradt, mivel nagy futási ideje miatt csak kis erőműszámra tudtam futtatni, viszont ott heurisztikával összemérhető eredményeket hozott veszteségben. Felmerül a kérdés, hogy ha sikerülni a futási idejét leredukálni, akkor nagy erőműszám esetén milyen eredményeket adna.

10.1 Szimulátor

Mivel a probléma végső megoldása a szimulátorba beintegrálása lenne az optimalizálónak, a Több csapat egy hajóval esetet összekapcsoltam a szimulátorral. Beintegrálni még felesleges lenne a jelenlegi állapotában bármelyik algoritmust, mivel még nem kezelnek annyi paramétert, amennyit a szimulátor. Mivel a hibás eredmények kiszűrése még nem történik meg, ezért a heurisztikus ütemtervet adtam meg szimulálásra, mivel az biztosan helyes, ha nem is optimális.

A szimulátornak a grafikus felhasználó felületén kívül létezik egy C# nyelvű API-ja, amin keresztül lehet eseteket felállítani és ütemtervet megadni annak lefuttatásra, ezt használtam a szimulációk futtatására. Mivel az optimalizáció során MATLAB-ot használtam, ezért annak saját fájlformátumában, *.mat*-ban tároltam az adatokat. Ezen adatok kódba való átemeléshez a csmatio[16] NET libraryt használtam, mely segít matlabmátrixokat többdimenziós double tömbökké konvertálni.

Szimulátorra való konvertálás

A szimulátor által használt paraméterek egy része eltér az általam használt paraméterektől, ezért köztük megfeleltetéseket kellett alkalmaznom. Ezen felül a szimulátor jóval több változót alkalmaz, mint az én modelljeim, ezért azokat konstans semleges adatokkal helyettesítettem (például meghibásodás típusa, technikus képességei), ami nem okozott problémát, mivel az ütemterv sem épített ezen paraméterek értelmességére.

Mértékegység

Az én világomban egy képzelt 2D Descartes-koordináta rendszerben helyezkednek el az erőművek, míg a szimulátornak többek között szélességi és hosszúsági fokokban lehet megadni a helyeüket. Egyszerűen megnéztem, hogy közelítőleg hány km egy fok, és átváltottam az értékeket. Ez természetesen nem pontos, mivel a földrajzi elhelyezkedés függvényében az átváltási szám változik, de teszteléshez megfelelő ez a pontosság. Az én rendszerem bal alsó sarka a [0; 0] pont, melyet a szimulátor számára az (53°, 358°)-kal feleltettem meg, mivel az API mintakódjában ez volt megadva mintaként, mint értelmes számérték.

Az x érték felel meg a hosszúsági foknak, az y a szélességinek, melyre a következő átváltást alkalmaztam: 1° long = 111320 m, 1° lat =110570. [17]

A szimulátorban a hajó sebességének mérése csomóban történik, míg én km/h-t használtam, ezt a konvertálást 1 km/h = 0.54 csomó értékkel csináltam.

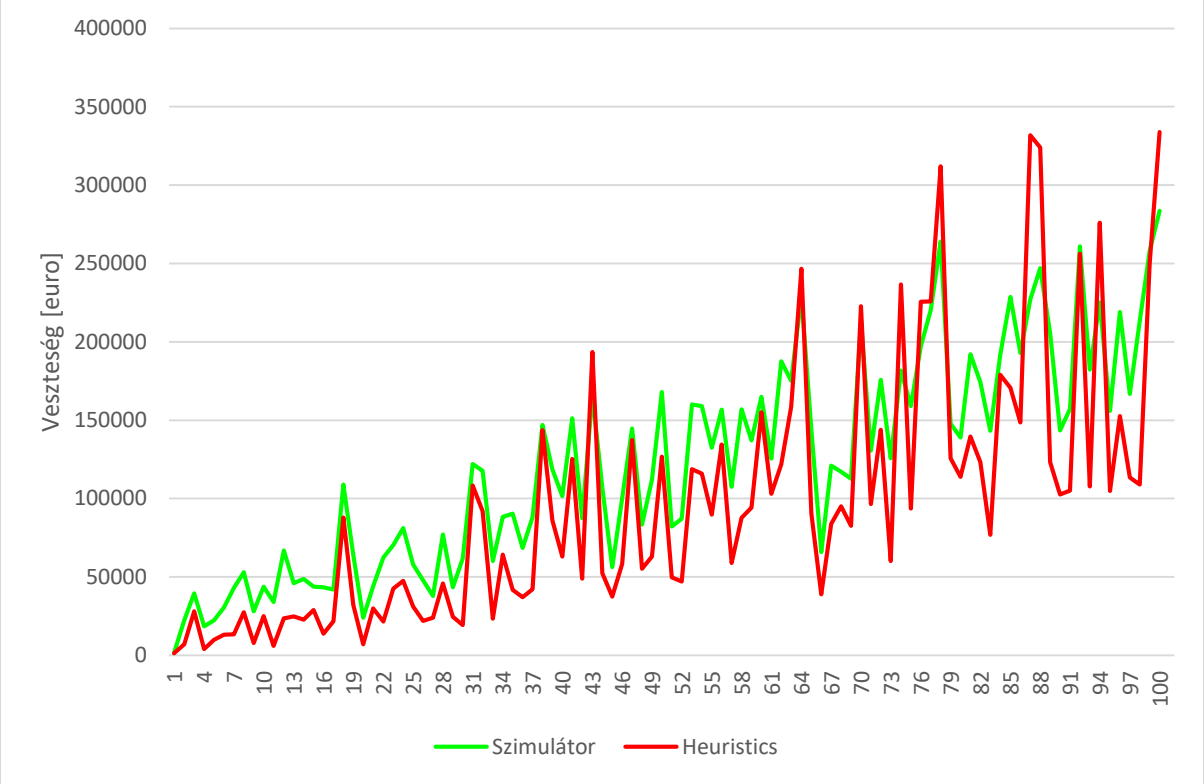
Költség

A szimulátorban a turbinák termelése megawattban van mérve, míg nálam euróban. Ennek feloldására az euróban megadott profitomat beadtam az erőműveknek, mint megawatttermelést, és konstans időjárást adtam meg, melyen pont ennyit termelnek, hogy megfeleljen a modelljeimnek, ahol nincs időjárás.

Hajóköltséggel nem számol a szimulátor, de visszaadja erőműtől erőműig a hajók megtett távolságát, így azok összegzéséből és a megfelelő költséggel való felskálázásból megkapható a modellben értelmezett üzemanyagköltség.

Eredmény

A szimulátor eredményének összehasonlítását a heurisztikus algoritmus eredményével a 13. diagram mutatja. Bár karakterisztikájukban hasonló a két függvény, mégis megfigyelhetünk nem elhanyagolható eltérésket. A koordináta transzformációra vezethetőek vissza az eltérések, mely súlyosan pontlatan, ezért a szimulátor hajóinak megtett távolsága jelentősen eltér a heurisztika hajóinak megtett távolságától. Mivel a sebességek átváltás után is majdnem azonosak, a más távolságok miatt az erőművek kieséssel töltött ideje is el fog térni, így a veszteségek üzemanyag és termelés kiesés tényezőjében is megjelenik a hiba, jelentőssé téve az összhibát.



13. diagram: A szimulátorral és a heurisztikus algoritmus költségfüggvényével kapott veszteségek

10.2 Futurework

Mindenekelőtt kell egy ellenőrző, amely kiszűri az olyan terveket, amelyek nem pontosan egyszer látogatnak meg minden erőművet. Jó, ha ez egy független modul az összes tervezőtől, mint látható volt, attól nem biztos, hogy az algoritmus betartja a kényszereket, hogy a dokumentációban az áll, hogy bármilyen tolerenciahatárt megadhatunk neki.

A kipróbált algoritmusok közül a nem lineáris ga()-n lehetne fejleszteni, tanulmányozva annak belső működését, és olyan alakban megadni neki a modelleket, amelyeket a legkönnyebben kezel.

Másik járható út feladatspecifikus megoldók keresése.

A kezdeti szkepticizmussal ellentétben láthattuk, hogy még egy egyszerű heurisztika és jó eredményt adhat, érdemes lehet jobb heurisztikákat megalkotni, akár nem is törekedve az optimumra, megelégedve egy szuboptimummal és a gyors lefutási idővel.

Hogy a valóságban is használhatóak legyenek a modellek, minél több paraméterrel ki kell őket bővíteni a felsoroltakból.

Ha elkészül egy sokparaméteres modell kellően gyors és hatékony algoritmussal, akkor össze kell kötni a szimulátorprogrammal, hogy a kezelői felületről is lehessen indítani ütemtervjavaslat készítését, majd az eredményt olyan formában kell visszajuttatni, hogy az már indítható szimuláció legyen.

11. Irodalomjegyzék

[1] https://ourworldindata.org/renewables. [Online]

[2] http://gwec.net/global-figures/graphs [Online]

[3] Daniel Fraile, Ariola Mbistrova:
Wind in power 2017 (Annual combined onshore and offshore wind energy statistics)
WindEurope
2018. február.

[4] http://www.wind-power-program.com/turbine_characteristics.htm. [Online]

[5] A. J. Chipperfield, P. J. Fleming and C. M. Fonseca*:*
GENETIC ALGORITHM TOOLS FOR CONTROL SYSTEMS ENGINEERING
Department of Automatic Control and Systems Engineering, University of Sheffield

[6] Digabel, S. Le:
Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm.
2011.

[7] https://www.inverseproblem.co.nz/OPTI/index.php/GetStart/Basics. [Online]

[8] S. Lucidi, M. Sciandrone*:*
A Derivative-Free Algorithm for Bound Constrained Optimization, Computational Optimization and Applications

[9] Hedengren, J. D. and Asgharzadeh Shishavan, R., Powell, K.M., and Edgar, T.F.:
Nonlinear Modeling, Estimation and Predictive Control in APMonitor, Computers and Chemical Engineering
2014.

[10] Michael Patriksson, Adam Wojciechowski, Ann-Brith Strömberg, Lina Bertling Tjernberg:
An optimization framework for opportunistic maintenance of offshore wind power system
2009.

[11] Jannie Jessen Nielsena, John Dalsgaard Sørensenab:
On risk-based operation and maintenance of offshore wind turbine components
Department of Civil Engineering, Aalborg University, Denmark
Wind Energy Division, Risø-DTU, Denmark
2011.

[12] Bhaba R. Sarker, Tasnim Ibn Faiz:
Minimizing maintenance cost for offshore wind turbines following multi-level opportunistic preventive strategy
Department of Mechanical and Industrial Engineering, Louisiana State University, Baton Rouge, United States
2016.

[13] https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm [Online]

[14] https://en.wikipedia.org/wiki/Knapsack_problem [Online]

[15] https://www.tno.nl/en/focus-areas/ecn-part-of-tno/ [Online]

[16] https://www.mathworks.com/matlabcentral/fileexchange/16319-csmatio-mat-file-i-o-api-for-net-2-0 [Online]

[17] http://www.longitudestore.com/how-big-is-one-gps-degree.html [Online]