

TDK Dolgozat

Hajdu László

2014

Szegedi Tudományegyetem
Juhász Gyula Pedagógusképző Kar
Informatika Alkalmazásai Tankszék

Kétmenetes gráfszínezési heurisztika műszakkiosztási feladat megoldására

Tudományos Diákköri Konferencia dolgozat

Készítette:

Hajdu László

III. Programtervező informatikus BSc

Témavezetők:

Dr. Krész Miklós
főiskolai tanár

Tóth Attila
tanársegéd

Szeged

2014

Tartalomjegyzék

ABSZTRAKT	2
BEVEZETÉS.....	3
1 A MŰSZAKKIOSZTÁSI PROBLÉMA, MINT OPTIMALIZÁLÁSI FELADAT	4
1.1 A MŰSZAKKIOSZTÁS EGYES ALKALMAZÁSI TERÜLETEI	4
1.2 TÖMEGKÖZLEKEDÉS OPTIMALIZÁLÁSA	7
1.3 MŰSZAKKIOSZTÁSI FELADAT	8
1.4 A MATEMATIKAI MODELL	10
2 KÉTLÉPÉSES HEURISZTIKA.....	13
2.1 KEZDETI EMBERSZÁM BECSLÉSE	13
2.2 SZABADNAPMINTÁK GENERÁLÁSA.....	14
2.3 GRÁF FELÉPÍTÉSE A MŰSZAKOK ALAPJÁN	16
2.4 GRÁF SZÍNEZÉSE	18
2.5 A GRÁF ÁTSZÍNEZÉSE TABU SEARCH SEGÍTSÉGÉVEL	20
3 EREDMÉNYEK.....	24
ÖSSZEFOGLALÁS.....	28
IRODALOMJEGYZÉK.....	29

Absztrakt

Napjainkban fontos szerepet tölt be a nagy foglalkoztatási létszámmal rendelkező vállalatoknál a munkavállalók feladatainak az adott tervezési időszakra vonatkozó költséghatékony ütemezése, műszakokhoz való hozzárendelése. A legtöbb esetben ezek a műszakok nem azonos hosszúságúak, így egyes emberek kevesebbet, míg mások többet dolgoznak adott időszak alatt. Általában a vállalat az alaphér mellett a túlóráért egy bizonyos túlóraszorzóval megnövelt túlórabért fizet a munkavállaló részére. A költség, melyet minimalizálni szeretnénk, az alul- és túlfoglalkoztatottság mértékéből adódik. A teljesítendő műszakok összmunkaideje segítségével az elvárt munkaidőket ismerve előre becsülhetjük, hogy milyen ember-szám mellett lesz az alul- és túlfoglalkoztatottság minimális. Az emberszám és az összmunkaidő segítségével számolhatunk egy elméleti alsó korlátot. Mindezek mellett a megoldásnak meg kell felelnie a törvényileg előírt szabályoknak is, amelyek többek között a szabadnapok kiosztásának gyakoriságára és a heti maximális munkaidő értékére vonatkozó kritériumokat tartalmazznak.

A fenti korlátozó tényezőkkel definiált feladat NP teljes, így egzakt megoldások csak kisebb méretű problémák esetében lehetségesek. A gyakorlati életben jellemző problémák azonban jelentős méretű feladatokat definiálnak: a műszakkiosztást gyakran több száz munkavállaló részére és jellemzően 1-3 hónapos időszakra vonatkozólag kell megvalósítani. Mindezeket túl, a valós környezetben való felhasználás során az elvárt futásidő néhány perctől néhány óráig terjed, ezért alkalmazás-orientált megoldás általában csak heurisztikák használatával érhető el.

Megoldásként egy kétlépéses gráfszínezési algoritmust valósítottunk meg. Az algoritmus először kiosztja a szabadnapokat, majd felépít egy gráfot, melynek csúcsait az elvégzendő műszakok alkotják, két csúcs között pedig akkor van él, ha a két műszak nem osztható ki ugyanazon ember számára. Az eljárás először egy becslést hajt végre az emberszámról vonatkozólag a munkaidők alapján, majd ezzel az „emberhalmazzal” kiszínezi a gráfot mohó módon. Végül egy tabukeresés segítségével átszínezi a gráfot úgy, hogy csökkenjen a költség.

Az algoritmust mesterségesen generált és valós inputokon teszteltük, a módszert összehasonlítottuk egészértékű programozási feladat által adott eredményekkel kisméretű feladatok esetén (napi műszakok maximális száma 50). Végeredményként megállapíthatjuk, hogy egy viszonylag nagyméretű bemenet esetén is (napi műszakok maximális száma 200) az algoritmusunk legtöbb esetben elérte az elméleti alsó korlátot, és futásidőben is elfogadható értékeket produkált.

Bevezetés

A műszakkiosztási feladat, mint probléma, a való életben sok helyen jelen van. A probléma lényege, hogy bizonyos előre meghatározott elvégzendő feladatokat fedjünk le lehetőleg kevés számú emberrel úgy, hogy az emberek foglalkoztatottsága hatékony legyen, és a megoldásunk megfeleljen bizonyos terület-specifikus megkötéseknek.

Maga az alapprobléma sok felhasználási területre általánosítható. Elmondhatjuk, hogy számos olyan helyen alkalmazzák, ahol embereket foglalkoztatnak az egyes vállalatok, illetve állami intézmények. Jellemző területek a közlekedési társaságok emberütemezése, a call centerek műszakbeosztása, és a kórházi nővérek műszakokba történő beosztása. Ezeken a területeken a dolgozók ütemezése NP teljes feladat [1]. Ebből, és a feladat sajátosságaiból adódóan a probléma egészértékű programozási feladatként történő megoldása a valós felhasználás során elfogadható futásidőket csak ritkán produkál. Céлом a dolgozat során az volt, hogy a futásidők szempontjából hozzam létre az egészértékű programozási feladat egy valós alternatíváját a problémára. Az algoritmus létrehozása során a közlekedési társaságok buszvezetőinek műszakkiosztásával foglalkoztam, és erre a problémára adtam egy heurisztikus megoldó módszert. Az algoritmus kezdetben három előkészítő fázist hajt végre, azonban a megoldást az ezt követő két lényegi fázis során alakítja ki. Az előkészítő fázisok alatt az algoritmus becsül egy emberszámot a munkaidők alapján, felépít egy gráfot, valamint kezdeti szabadnapmintákat generál minden egyes dolgozó számára. Az ezt követő első lényegi fázisban egy gráfszínezést valósít meg, mellyel egy kezdeti megoldás jön létre. A második lényegi fázis során a kezdeti megoldás segítségével egy lokális kereső algoritmus átszínezéseket végez a gráfon, melynek eredményeképpen javít a kezdeti megoldásunkon.

A dolgozatomban ezt a problémára adott heurisztikus megközelítést fogom ismertetni és összehasonlítani a probléma egészértékű programozási feladatként történő megoldásával, kisméretű bemenetek esetén. A továbbiaknak bemutatom a műszakkiosztási probléma alkalmazási területeit, valamint a problémát és annak hátterét a valós felhasználás során. Ezután az egészértékű programozási feladat modellje, valamint a heurisztika lépései kerülnek ismertetésre. A tesztelés során az egészértékű programozási feladat és az algoritmus által elért eredményekről számolok be generált, valamint valós környezetből származó bemeneteken.

1 A műszakkiosztási probléma, mint optimalizálási feladat

1.1 A műszakkiosztás egyes alkalmazási területei

Napjainkban a műszakkiosztási feladat az ipar egyre több szegmensében jelen van. Leggyakrabban közlekedési vállalatoknál, ahol az egyes teljesítendő járatokhoz kell hozzárendelni adott mennyiségű és képzettségű embert. Az alapfeladat különböző fajtái megtalálhatóak olyan területeken is, mint kórházi alkalmazottak ütemezése, vagy call centerek műszakbeosztása. A műszakkiosztási feladatot általában egymást követő napok sorozatára végezzük, amit *tervezési időszaknak* nevezünk. Ez általában 1-3 hónap hosszúságú, mely időszak alatt minden napon adott egy munkavállaló által elvégzendő feladat, amelyet *műszaknak* nevezünk. A feladat során a rendelkezésre álló munkavállalókat kapjuk meg, amely azt határozza meg, hogy a megoldás során maximálisan mennyi embert foglalkoztathatunk. A feladatban a felhasználható munkavállalókat és azok halmazát *emberhalmaznak* nevezzük. A megoldásunkkal szemben elvárt, hogy megfeleljen bizonyos törvényileg előírt előre megadott szabályoknak, melyek meghatározzák, hogy az emberek milyen módon lehetnek a műszakokra beosztva. Ezeket a feltételeket illetve szabályokat *korlátozó tényezőknek* nevezzük. Mivel egy ember nem dolgozhat a tervezési időszak minden napján, ezért az erre vonatkozó korlátozó tényezők szerint *szabadnapokat* kell számára kiosztanunk. A tervezési időszakunk általában hónapokra van osztva. Ezen hónapokra egyesével kiszámolható a hétköznapi napok száma, amely meghatározza, hogy egy munkavállalótól mennyi ledolgozott napot várunk el. Ezeket a napokat összefoglalva *elvárt munkanapoknak* nevezzük. Az alapfeladat lényegében az, hogy az előre megadott emberhalmazt felhasználva adjunk egy hatékony megoldást, és teljesítsük az összes műszakot úgy, hogy minden műszakra osszunk ki pontosan egy embert, és a kapott megoldásunk teljesítse az előre definiált korlátozó tényezőket minimális költséggel. Ez kevés dolgozó esetén viszonylag könnyen, akár kézzel is elvégezhető feladat. Azonban, mivel általában a vállalatok illetve állami intézmények sokszor több száz személyt foglalkoztatnak, rendkívüli módon komplexé válhat a probléma.

A korlátozó tényezők alkalmazási területenként eltérőek lehetnek. Minden egyes területen jelen vannak olyan szabályozások, melyek előírják, hogy két elvégzett műszak között el kell telnie egy meghatározott pihenőidőnek. Tehát, ha esti műszakot végez egy dolgozó, akkor másnap reggelre ne osszák be dolgozni. Megjelennek olyan törvényileg előírt szabályozások is a feladatban, amelyek megkövetelik, hogy adott munkavállalónak egy tervezési időszak alatt (1-3 hónap) minimálisan mennyi szabadnapot köteles kiosztani a munkáltató, illetve ezek a szabadnapok hol és hogyan kerülhetnek kiosztásra. Általánosságban elmondható a korlátozó

tényezőkről, hogy az előbb említettek majdnem minden alkalmazási területen megjelennek. Azonban, attól függően, hogy mely munkaterületre kívánunk műszakokat kiosztani, jelen lehetnek terület-specifikus szabályok is. Ezek általában két csoportba sorolhatók. Azokat, amelyeket semmilyen esetben nem szeghetünk meg a megoldásunk során *Hard Constraint*-ként definiálja a szakirodalom. Amelyeket bizonyos esetekben megszeghetünk, *Soft Constraint*-nek nevezik. A következőkben ismertetni fogom a probléma egyes területeken fellelhető változatait.

A repülőgép társaságok pilótaütemezése során felmerülhetnek olyan korlátozó tényezők, melyek egyes repülőgéptípusok vezetését bizonyos képzettséghez kötik. Tehát a pilóták közül legalább egynek rendelkeznie kell a megfelelő képesítéssel a járat elvégzéséhez. Itt minden egyes járathoz általában két pilótát osztanak be úgy, hogy egy kapitány és egy másodpilóta teljesít szolgálatot. Továbbá a járatok sokszínűségéből adódóan előtérbe kerül az a probléma, hogy az egyes műszakok hossza jelentősen eltérhet egymástól. [2,3]

Szintén közlekedési társaságok által használt területhez tartozik a buszvezetők ütemezése, mely nagyon hasonló a pilótaütemezés problémájához. Annyi a különbség, hogy itt legtöbb esetben nincs képzettséghez kötve egy adott műszak teljesítése. Ezzel szemben viszont sokkal összetettebb szabályrendszernek kell megfelelnie a megoldásnak. Például, más szabály vonatkozik valakire, ha 50 km alatti, vagy 50 km feletti járatokat teljesít a műszak során. Ezen a területen általában a buszvezetők szolgálatot teljesítenek a nap során. Azt az időt, amely alatt szolgálatban vannak, *szolgálati időnek* nevezzük. A szolgálati időn belül különböző tevékenységeket kell folytatniuk, melyeknek csak egy része számít bele az elszámolt idejükbe, ami alapján a bérüket kapják. Ezt a szolgálati időn belüli időt, mely általában kevesebb, mint a teljes szolgálatban töltött idő, *munkaidőnek* nevezzük. Minden buszvezető rendelkezik egy szerződéssel, ami kifejezi, hogy napi átlagban mennyi munkaidőt várnak el tőle. Ehhez kapcsolódóan meg kell említenünk még az *elvárt munkaidőt*, amit úgy kapunk meg, hogy a tervezési időszak alatti elvárt munkanapok számát megszorozzuk az adott dolgozó szerződéstípusával. Amely munkavállaló az elvárt munkaidején felül dolgozik *túlfoglalkoztatottnak*, míg ezt a munkaidőt el nem érő dolgozót *alulfoglalkoztatottnak* nevezzük. Meg kell említeni, hogy ha egy buszvezető munkaideje meghaladja a szerződése szerinti elvárt munkaidejét a tervezési időszak alatt, akkor az ezen felüli időt *túlórának* nevezzük, mely után egy *túlóraszorzóval* megnövelt túlóráért kell fizetni számára. Itt jelen lehetnek olyan, úgynevezett osztott műszakok, melyek a szolgálati időn belül több munkaidő-részre vannak osztva, vagyis egy buszvezetőnek a napi munkaidejét két vagy több részletben kell teljesítenie. Ezen terület sajátosságaiból adódik, hogy léteznek ilyen osztott műszakok, mivel a nap folyamán az elvégzendő ese-

mények eloszlása nem szabályos, hanem általában a délelőtt és a délután folyamán van több teljesítendő járat. Tovább bonyolítja a helyzetet, hogy jellemzően a közlekedési társaságoknál a műszakok nem egyforma hosszúak, így egy adott emberszám mellett előfordulhat, hogy míg egy dolgozó jelentősen túlfoglalkoztatott, addig egy másik ugyanannyi munkanappal az elvárt munkaidejét sem teljesíti. [4–6]

A kórházakban általában a nővérek beosztásánál fordulhat elő ez a probléma. Itt a repülő társaságokhoz hasonlóan szintén korlátozó tényezőként jelentkeznek az egyes műszakok által megkövetelt képzettségi szintek. A dolgozók ezen a területen úgynevezett váltásokban dolgoznak, és ezek mentén kell elosztani a nővérek között a műszakokat. Jellemzően ezek a váltások reggeli, délutáni és éjszakai időszakokban vannak. Ezen a területen egy dolgozó műszakja teljes egészében munkaidőként van elszámolva, így itt a lényeg, hogy milyen felhasznált nővérszám mellett oldjuk meg a feladatot. [7,8]

A call centerek ütemezésekor figyelembe kell venni, hogy a szolgáltatás minősége érdekében megfelelő számú operátornak mindig rendelkezésre kell állnia, hogy a bejövő hívások minden esetben kezelhetőek legyenek. Itt a legnagyobb problémát az jelenti, hogy optimális esetben az operátorok foglaltsága közel teljes legyen, azonban a legtöbb esetben a terheltség előre nem jósolható meg. [9,10]

Ezekkel a területekkel foglalkozó vállalatok és intézmények életében fontos szerepet tölt be, hogy a dolgozóikat költséghatékonyan foglalkoztassák, illetve lehetőség szerint minimális számú emberrel végeztessék el a szükséges időszakra tervezett műszakokat. A munkaadónak egy alapbért mindenféleképpen ki kell fizetnie, akkor is, ha a munkavállaló nem teljesítette az alapbérben foglalt elvárt munkaidőt, illetve az ezen felül ledolgozott időért túlórabért is köteles kifizetni. Ez felesleges költségként jelentkezik a vállalat életében. Ezekben az esetekben jelenthet jelentős költségmegtakarítást, ha hatékonyan osztjuk szét az embereket a műszakok között. Általában az alulfoglalkoztatottság és a túlfoglalkoztatottság mértékét kell minimalizálnunk úgy, hogy mindemellett a lehető legkevesebb dolgozóval oldjuk meg a feladatot, továbbá jelen lehetnek a megszeghető korlátozások az úgynevezett soft constraintek, amelyeknek a betartottsága szintén megjelenhet a költségben.

A dolgozatban kizárólag a buszvezetők ütemezésének problémájával foglalkozom, és erre adok egy heurisztikus megoldó módszert. Ismertetem a busztársaságok közlekedésütemezésének a felépítését, és a problémát elhelyezem ebben a környezetben. Rendelkezni fogok erre a területre jellemző korlátozó tényezőkkel és definiálom, hogy e probléma esetén mely tényezőkre kell odafigyelni ahhoz, hogy hatékony legyen a buszvezetők foglalkoztatása.

1.2 Tömegközlekedés optimalizálása

A tömegközlekedés optimalizálása területén az egész probléma legtöbb esetben több fázisból épül fel. Tekintsük a következő négy fázist, melyek részletesebb rálátást biztosítanak az általam tárgyalt problémára, és elhelyezik azt a busztársaságok ütemezési problémái között, melynek a fázisai a következők:

- Járművek ütemezése
- Jármű-hozzárendelési feladat
- Járművezetők ütemezése
- Műszakkiosztás.

Járművek ütemezése: Ezen fázis keretein belül a tervezési időszakra adunk egy jármű-ütemezést. A busztársaság kezdetekben menetrendekben meghatározza az elvégzendő járatokat. Minden társaság rendelkezik egy gépjárműparkkal, melyek depókba vannak sorolva a tulajdonságaik alapján. Meghatározott, hogy bizonyos járatokat csak bizonyos tulajdonságokkal rendelkező gépjárművek végezhetnek el. Ezeket felhasználva a fázis során az a cél, hogy a járművek egy csoportjával lefedjük az elvégzendő járatok halmazát minimális költség mellett. Konkrét gépjárműveket e fázis keretén belül nem rendelünk az egyes járatütemezésekhez.

Jármű-hozzárendelési feladat: A fázis feladata az, hogy az előzőekben meghatározott járműütemezésekhez konkrét járműveket rendeljünk úgy, hogy teljesíthetők legyenek az adott járatok. Ehhez szükségünk van arra, hogy minden egyes jármű számára tankolásokat, karbantartásokat és parkolásokat osszunk ki az elvégzendő járatok között.

Járművezetők ütemezése: A probléma keretén belül az előző fázisokban meghatározott járműütemezésekhez hozunk létre azokat teljesítő vezetőütemezéseket. Ez lényegében azt jelenti, hogy az előző fázis eszközeseményeit (járatok, tankolások stb.) rendezzük ember által elvégezhető műszakokba. Ebben a fázisban hozzuk létre a feladatunk bemeneteként szolgáló műszakokat. Minden műszak különböző eseményekből áll, melyeket el kell végezni úgy, hogy bizonyos előírt egy napra vonatkozó szabályokat be kell tartani.

Műszakkiosztás: A fázis során az előzőekben létrehozott műszakokhoz fogunk kiosztani konkrét buszvezetőket. Ezen rész folyamán általában nem foglalkozunk a műszakokon belüli egyes eseményekkel, mivel egy műszakot teljes egészében egy dolgozónak osztunk ki a megoldás során. A dolgozat során ezt a fázist fogom részletesebben ismertetni.

Meg kell említenünk, hogy a szakirodalomban léteznek ezeket a fázisokat kombináló optimalizálási módszerek is. Többek között kezelhetők egyben a járművezetők vezénylése és ütemezése fázisok, valamint kombinálható a járművek és emberek ütemezése is. Általános-

ságban a kombinált módszerekről elmondható, hogy az ütemezések ily módon történő megközelítése hatékony megoldást ad ugyan, de ezen a módon történő optimalizálás számítási igénye magas. Ilyen, a probléma kombinált módon történő optimalizálására ad példát a következő szakirodalom.[11] A gyakorlatban, ahol általában nem állnak rendelkezésre hatékony optimalizáló eszközök a fázisokat külön kezelik.

1.3 Műszakkiosztási feladat

A feladat az, hogy adott egy emberhalmazunk, amely legyen C . Ennek segítségével el kell végeznünk előre meghatározott műszakokat, melyek halmazát jelölje S . A műszakjaink adottak, ugyanúgy, mint a felhasználható emberek halmaza. A feladat megoldása során a cél, hogy a műszakokat le kell fednünk az emberekkel, tehát legyen $f = S \rightarrow C$ egy hozzárendelés, melynek során a műszakokat lefedjük a járművezetőkkel. Vagyis az előre megadott emberhalmazt felhasználva adjunk egy hatékony megoldást úgy, hogy minden műszakra pontosan egy embert osszunk ki. A megoldásnak meg kell felelnie bizonyos korlátozó tényezőknek, melyek közlekedési társaságoként eltérőek lehetnek. Léteznek specifikusan erre a területre vonatkozó általános szabályok, melyeket minden társaságnak be kell tartani. Ezek a szabályok [12] a következők:

Szabadnapokra vonatkozó szabályok:

- Az egy hónapban kötelezően kiadandó szabadnapok száma szabályozva van. A törvény előírja, hogy egy buszvezető számára legalább 4 napot kötelesek vagyunk kiadni egy hónap alatt. Ezt jelölje RD .
- Legfeljebb 6 nap folyamatos munkavégzés után szabadnapot kell kiadnunk az adott buszvezető számára. Ezt jelölje WD .

Pihenőidőre vonatkozó szabály:

- Egy dolgozó nem kaphat egy nap két műszakot és az egymást követő műszakjai között legalább 12 óra pihenőidőt kell hagynunk. Ez azt jelenti, hogy ha egy buszvezető elvégzett egy adott műszakot, utána 12 órán keresztül nem kezdheti el a következő műszakját.

Munkaidőre vonatkozó szabály:

- A heti maximális munkaidő nem lehet 48 óránál több. Jelölje *WT*. Ez azt jelenti, hogy egy héten egy buszvezető által elvégzett műszakok munkaidejének az összege nem lépheti túl a törvényileg előírt értéket, mely esetünkben 48 óra.

Ezen szabályokon felül minden egyes busztársaság rendelkezhet specifikusan saját magára szabott szabályokkal, azonban mivel ezek általában minden társaságnál különbözőek, a dolgozat keretén belül ezeket nem tárgyalom. Próbálok viszont egy olyan heurisztikus megoldást adni a problémára, amely keretében a korlátozó tényezők rugalmasan bővíthetők.

Tekintsük az elvégzendő műszakokat. Egy ilyen műszak ténylegesen több különböző eseményből áll, melyek előírják az adott járművezetőnek, hogy a szolgálati ideje alatt mely feladatokat kell teljesítenie, és mekkora munkaidő kerül elszámolásra. Mivel egy adott hosszúságú tervezési időszakra szeretnénk a kiosztást elvégezni, ezen időszakon belül minden egyes műszakot fel kell venni egy adatszerkezetbe a megfelelő tulajdonságokkal. Ezek a tulajdonságok a következők:

- dátum
- kezdési idő
- befejezési idő
- munkaidő

A dátum azt jelenti, hogy a tervezési időszak melyik napján van az elvégzendő műszak. A tervezési időszak hosszát jelölje esetünkben *L*. A kezdési és a befejezési idő egy percben adott érték, mely meghatározza, hogy a nap mely percében kezdődik és végződik a buszvezető műszakja. Végül a munkaidő kifejezi, hogy a dolgozó szolgálati idejéből mennyi idő telik ténylegesen munkával. Meg kell jegyeznünk, hogy a munkaidő a legtöbb esetben nem egyenlő a kezdési és befejezési idő között eltelt idővel, azaz a szolgálati idővel, mivel jelen lehetnek a fentebb tárgyalt úgynevezett osztott műszakok, továbbá a műszakban lehetnek, úgynevezett holtidők, melyek nem számítanak bele egy dolgozó munkaidejébe.

A cél megoldást adni a problémára, mellyel szemben elvárt, hogy megfeleljen a korlátozó tényezőinknek, továbbá minimális legyen a költsége. A probléma és a költség megértéséhez tekintsük először a buszvezetők halmazát, akik rendelkeznek egy *szerződéssel*, mely alapján ki lehet számolni a számukra adott tervezési időszak alatt elvárt munkaidőt. Esetünkben minden dolgozónak 8 órás szerződése van, számunkra ez az elvárt munkaidő szempontjá-

ból fontos, melyet megkaphatunk úgy, hogy egy adott időszak alatti elvárt munkanapok számát megszorozzuk a szerződés típusával, vagyis 8 órával.

$$\text{elvárt munkaidő} = \text{munkanapok száma} * \text{szerződés típusa}$$

Hónaptól függően általában 20-21 nap az elvárt munkanapszám, ami azt jelenti, hogy ha például egy 20 munkanapos hónapot tekintünk, akkor egy dolgozó ledolgozott munkaidejének a lehető legjobban meg kell közelítenie a 160 órát. A példánál maradva, mivel a szerződés szerint az adott buszvezetőnek mindenféleképpen ki kell fizetni a 160 órát, nem jó megoldás, ha ténylegesen kevesebbet dolgozik ennél. Azonban ha ezt túlteljesíti, akkor túlórárt is kötelesek neki kifizetni, amit egy megadott túlóraszorzóval kell megszorozni. Tehát költséghatékonyabb, ha egy műszakját egy másik, például alulfoglalkoztatott embernek adjuk oda, vagy valamely műszakját elcseréljük egy másik buszvezető műszakjával, melynek hatására csökken a túlórája. A megoldás során a fentebb leírt szabályok figyelembevételével dolgoztam. Így a heurisztikus megoldás esetén alkalmazott költség a következő:

$$\text{költség} = \alpha * \text{alulfoglalkoztatottság} + \beta * \text{felülfoglalkoztatottság} + \gamma * \text{emberszám}$$

Ahol esetünkben α az alulfoglalkoztatottság költségszorzója, β a felülfoglalkoztatottság költségszorzója és γ az emberszám költségszorzója. Esetünkben ezek a költségszorzók egyenlően egyek. Tehát a cél az, hogy lehetőség szerint minimális emberszám mellett legyen az dolgozók alul- és felülfoglalkoztatottsága a lehető legkevesebb. A probléma részletesebb megértéséhez írjuk fel a matematikai modelljét.

1.4 A matematikai modell

Legyen D a tervezési időszak napjainak sorozata, D^{Week} egy hét napjai, D^{Mon} pedig egy hónap napjai. Emellett definiálnunk kell a tervezési időszak hosszát, vagyis az ütemezett napok számát, ami legyen L , továbbá legyen L_m az m hónap napjainak száma. A továbbiakban jelölje j és p ($j \in D, p \in D$) a napokat és w a heteket. Szükségünk van az emberek halmazára, ez legyen C , valamint a műszakok S halmazára, ahol jelentse S_j a j napon elvégzendő műszakokat. Legyen i ($i \in C$) az i . sofőr, valamint k és q ($k \in S_j, q \in S_j$) jelöljenek egy-egy műszakot. Legyen SS_{pq}^{jk} egy kompatibilitási reláció, melynek értéke 1, ha a j . nap k . műszakja és a p . nap q . műszakja lehet ugyanahhoz az emberhez rendelve, 0 különben. Erre azért van szükség, mert léteznek egymást követő napokon olyan elvégzendő műszakok, melyeket nem

végezhet ugyanazon ember, mivel nincs közöttük annyi idő, amennyit a törvény pihenőidőként előír. Legyenek WT maximális munkaidő egy héten, WD az egymást követő maximális munkanapok száma, D_p^{WD} a p . naptól kezdődő egymást követő $WD + 1$ napok, RD a havi minimális pihenőnapok száma. Minden egyes járművezetőhöz szükségünk van a szerződése szerint elvárt munkaidőre, ami esetünkben 8 óra. Ez legyen $ct(i, m)$, valamint ahhoz hogy az emberszámot is minimalizálni tudjuk, legyen $cc(i)$ egy dolgozó alkalmazásának a költsége. Jelentse $wt(j, k)$ és $st(j, k)$ minden j . nap k . műszakjának a munkaidejét és a szolgálati idejét. Végül legyen ET a túlóra költségszorzója.

A modell felírásához szükség van a következő változókra:

1. $x_{ijk} \in \{0,1\}$ értéke 1, ha az i . járművezetőnek osztjuk ki a j . nap k . műszakját, különben 0.
2. $z_{ij} \in \{0,1\}$ értéke 1, ha az i . járművezető dolgozik a j . napon, különben 0.
3. $y_i \in \{0,1\}$ értéke 1, ha az i . járművezető dolgozik az egész tervezési időszak alatt, különben 0.
4. $\pi_i \geq 0$ legyen az i . járművezető túlórája

Az egész értékű modell feltételei a következők:

$$\forall j \forall k \sum_i x_{ijk} = 1 \quad (1)$$

$$\forall i \forall j \sum_k x_{ijk} = z_{ij} \quad (2)$$

$$\forall i \sum_j z_{ij} \leq Ly_i \quad (3)$$

$$\forall i \forall j \forall p \forall k \forall q x_{ijk} + x_{ipq} \leq SS_{pq}^{jk} + 1 \quad (4)$$

$$\forall i \forall w \sum_{j \in D_W^{Week}} \sum_k x_{ijk} wt(j, k) \leq WT \quad (5)$$

$$\forall i \forall p \sum_{j \in D_p^{WD}} z_{ij} \leq WD \quad (6)$$

$$\forall i \forall m \sum_{j \in D_m^{Mon}} z_{ij} \leq L_m - RD \quad (7)$$

$$\forall i \sum_m (\sum_{j \in D_m^{Mon}} \sum_k x_{ijk} wt(j, k) - ct(i, m)) \leq \pi_i \quad (8)$$

A célfüggvényünk pedig a következő:

$$\min \sum_i (ET \pi_i + y_i cc(i)) \quad (9)$$

Az (1) feltétel azt írja le, hogy minden embert pontosan egy műszakhoz kell hozzárendelnünk. A (2) biztosítja, hogy egy ember akkor és csak akkor dolgozik egy nap, ha van hozzárendelve műszak, továbbá hogy legfeljebb egy embert lehet egy műszakhoz rendelni. A (3) feltétel biztosítja, hogy egy embert akkor és csak akkor alkalmazunk, ha dolgozik valamelyik napon. A (4) feltétel zárja ki azt, hogy egy emberhez két olyan műszakot rendeljünk, melyek közül a szabályok miatt csak az egyiket tudná elvégezni. Az (5) definiálja, hogy egy héten a munkaidő nem haladhatja meg a törvényileg előírt maximális munkaidőt, vagyis 48 órát. A (6) feltétel biztosítja, hogy a vezetőknek legfeljebb 6 egymást követő munkanapjuk lehet. A (7) biztosítja, hogy legyen meg a tervezési időszakra a minimális pihenőnapszám, vagyis hónaponként legalább 4 szabadnapot ki kell adnunk. A (8) feltételben definiáljuk a túlóra mértékét. Végül a célfüggvényünk (9) minimalizálja az emberek túlóraköltségének és az alkalmazási költségének az összegét. A heurisztikus módszer során fontos megemlítenünk azt, hogy az algoritmus két fázisa miatt a költségünket két részre bontottuk. A gráf színezése során az algoritmus az emberszámot minimalizálja, míg a második fázis során, az alul- és túlfoglalkoztatottságot csökkenti. Az algoritmus második fázisa a költség szempontjából tekinti az alulfoglalkoztatottságot is, mivel ha a színezés közben, az algoritmus által végzett kezdeti becslés által kialakított emberszámmal nem vagyunk képesek megoldani a feladatot, akkor az utoljára bevezetett ember kevés műszakot kap, így túl nagy lesz az alulfoglalkoztatottsága. Ezért ahhoz, hogy a megoldásunk elfogadható legyen, az alulfoglalkoztatottságot is minimalizálnunk kell. A költség ilyen módosítása a heurisztikus megoldásunk során nem fog lényegében különbözni az egészértékű programozási feladat költségétől, mivel a túlfoglalkoztatottságot szintén minimalizáljuk, és az alulfoglalkoztatottság minimalizálása csupán kiegyenlítettebbé teszi a foglalkoztatottságot.

A megoldás során az általam megadott heurisztikus módszert összehasonlítottam ezen egészértékű programozási feladat által adott eredményekkel. A következő fejezetben ismertettem a feladat megoldására adott algoritmust.

2 Kétlépéses heurisztika

Az algoritmus egy gráfszínezésen alapuló kétlépéses heurisztikus módszer. Az elméleti alapját M. Gamache által fejlesztett módszer [13] képezte, ahol tabu keresésen alapuló gráfszínezési algoritmust mutattak be a repülőgép pilóták ütemezésére. Ebben a problémában az elvégzendő járatok száma szempontjából bizonyos pilótákat előnyben, míg másokat hátrányban kell részesíteni a képzettségük alapján. Az általam bemutatott algoritmus alapját ez a módszer képezte úgy, hogy esetünkben figyelembe kell venni az első fejezetben leírt szabályokat, és a megoldásunkat úgy kell megkonstruálnunk, hogy a buszvezetők ütemezése területén a fontos szempontoknak eleget tegyen.

Az algoritmus különböző fázisokból áll melynek lépései a következők:

1. Kezdeti emberszám becslése
2. A kezdeti emberszámhoz szabadnapminták generálása
3. Gráf felépítése a műszakok alapján
4. Gráf színezése
5. Tabu Search segítségével a gráf átszínezése

Meg kell említenünk, hogy míg az algoritmus 1-3. része az előkészítés fázisába tartozik, addig a 4. és 5. rész a címben is említett két lényegi lépés, amelyek során a megoldást kialakítjuk.

2.1 Kezdeti emberszám becslése

Mielőtt elkezdenénk építeni és színezni a gráfunkat, szükségünk van egy emberszám becslésre, mely segítségével meghatározhatjuk, hogy az adott mennyiségű ledolgozandó műszakot milyen elméleti emberszám mellett lehetne a leghatékonyabban szétosztani a buszvezetők között. Mivel ismerjük az összes elvégzendő műszak munkaidejét, és dolgozóink szerződését, könnyen kiszámolható egy olyan emberszám, mely esetén az összes buszvezetőnek átlagban a szerződése szerint elvárt munkaidejétől a lehető legkisebb mértékben fog eltérni a tényleges munkaideje. Természetesen ez nem jelenti azt, hogy ez a legkisebb eltérés el is érhető, azonban emberszám tekintetében egy viszonylag jó becslést fogunk kapni. Fontos látnunk, hogy az emberszámra egy triviális alsó korlát a műszakok száma a legnagyobb terheltségű napokon, mivel az összes műszak kizárja az összes aznapi műszakot. Az emberszám becslését a következőképpen végezzük:

$$\text{becsült emberszám} = \text{round}\left(\frac{\text{összes munkaidő}}{\text{szerződés szerinti elvárt munkaidő egy sofőr esetén}}\right)$$

Az összes munkaidőt úgy kapjuk, hogy összeadjuk az összes elvégzendő műszak munkaidejét. Így megállapítható, hogy összesen mennyi munkát kell elvégezniük a buszvezetőknek. Ezt el kell osztanunk egy dolgozó tervezési időszak alatt elvárt munkaidejével. Ez azt jelenti, hogy kiszámolható, hogy ha minden buszvezető a szerződése szerinti mennyiségű munkát végez, akkor mennyi munkaidő fog kimaradni, amely alul- vagy felülfoglalkoztatottságként jelenik meg. Fontos látnunk, hogy esetünkben, mivel az elvégzendő műszakok összmunkaideje adott, a problémát az jelenti, hogy a becsült emberszám segítségével elvégezzük a műszakok kiosztását úgy, hogy az egyes buszvezetők munkaideje a lehető legkevésbé térjen el az elvárt munkaidejétől.

Az előre becsült emberszám segítségével meghatározhatunk egy elméleti alsó korlátot a foglalkoztatásra vonatkozóan, melynél jobb megoldást nem lehet elérni adott emberszám mellett, és melyet úgy kapunk, hogy az összes műszak összes munkaidejéből kivonjuk a felhasznált buszvezetők szerződése szerinti elvárt munkaidejét. Így megkapjuk, hogy ha mindenki az elvárt munkaideje szerint dolgozik, akkor mennyi túlfoglalkoztatott vagy alulfoglalkoztatott idő lesz globálisan. Az elméleti alsó korlátunk kiszámolható a következő módon.

$$\text{elméleti alsó korlát} = \text{összes munkaidő} - \text{becsült emberszám elvárt munkaideje}$$

A megértéséhez vegyük a következő egyszerű példát. Ha a tervezési időszak alatt ledolgozandó műszakok összes munkaideje 30 274 perc, és egy ember egy hónapra számolt elvárt munkaideje $21 \cdot 480 = 10\,080$ perc, akkor kiszámolható, hogy a feladat megoldásához becsült emberszám 3 lesz. Mivel $3 \cdot 10\,080 = 30\,240$ perc. Így a példánk esetében ez úgy adódik, hogy $30\,274 - 30\,240 = 34$ ami azt jelenti, hogy a becsült emberszámunk esetén az elméleti alsó korlátunk 34 lesz, ami túlfoglalkoztatottságként fog jelentkezni. Előfordulhat azonban, hogy a becsült emberszám segítségével nem tudjuk megoldani a feladatot, ezért emberszámot kell növelnünk az algoritmus későbbi fázisai során, melynek következtében az összes alulfoglalkoztatottság növekedni fog.

2.2 Szabadnapminták generálása

A szabadnapokra vonatkozó szabályokat a következőképpen fogjuk betartani. Először bizonyos előre definiált szabadnapmintákat fogunk létrehozni, melyek teljesítik a törvényileg előírt és egyéb feltételeket. Ez azt jelenti, hogy minden egyes buszvezető számára előre megkötött kezdeti mintákat generálunk, melyek meghatározzák, hogy legalább mely napokon kell szabadnapot kiadni számukra. Ezeknek a mintáknak egyik előnye, hogy általuk a heurisztikus

módszer rugalmassá válik, tehát lehetőségünk van arra, hogy az általunk használt szabályok mellé könnyen újakat definiáljunk. Sok alkalmazási területen bonyolult szabályok vannak, például meg lehet szabva, hogy egy hónapban legalább egy szabadnapnak hétvégére kell esnie, vagy a dolgozó előre leadhat egy szabadnap igényt, melyet a megoldás során soft constraint-ként veszünk figyelembe. Az egyre bonyolultabb szabályok a legtöbb esetben megnehezítik a szabadnapok helyes kiosztását. Egyes esetekben szükség lehet továbbá arra is, hogy a dolgozó által előre leadott szabadnapokat fixen rögzítsük a megoldás során, vagy előre meghatározott olyan mintákat hozzunk létre, melyekben megadott napokon kötelezően szabadnapot kell kiadnunk a buszvezetők részére.

Valószínűség szerinti szabadnapminta generálás

Egyik ilyen, a későbbiek során nem használt módszerünk a szabadnapok bizonyos valószínűségek szerinti véletlen generálása. Ez úgy működik, hogy minden egyes buszvezető minden egyes napjához egy, a napi műszakok száma szerinti valószínűséget rendelünk, mely azt fejezi ki, hogy adott napon mekkora valószínűséggel lesz szükség az adott buszvezető munkájára. Természetesen csak olyan mintát generálhat ez a módszer, amely teljesíti a szabályokban megadott feltételeket. Ezen módszer nem bizonyult hatékonynak, mivel tapasztalataim alapján az algoritmus egy későbbi fázisában a lokális keresésen alapuló átszínezések esetén nem garantál elég nagy keresési teret az átszínező algoritmus részére.

6-1 szabadnapminták

Az ötlet az, hogy a kezdeti mintáink meghatározásakor szabadnapmintát használunk ugyan, de nem kötjük ki véglegesen az összes kiadandó szabadnapot, hanem hagyjuk, hogy egy későbbi fázis tegye meg ezt, mely már figyelembe veszi, hogy a költség szempontjából mely minta a legjobb. A szabadnapminták generálása során a későbbi megoldás szempontjából a legcélravezetőbb az volt, ha egy olyan kezdeti mintát generálunk, ami minimálisan teljesíti az előre meghatározott követelményeket. Így a minta a gráf színezése során is kevés kizárást fog eredményezni, és a lokális keresésen alapuló átszínezés során a tabu keresésnek viszonylag nagyméretű keresési tere lesz. A 6-1 szabadnapminta azt jelenti, hogy minden esetben 6 nap munka után 1 szabadnapot osztunk ki. Egy ilyen minta teljesíti mindkét kikötésünket a szabadnapokra vonatkozólag. Az egyik ilyen, hogy legfeljebb 6 nap folyamatos munkavégzés után pihenőnapot kell kiadnunk az adott buszvezető számára. E feltétel teljesülése triviális a 6-1 minta esetén. A másik szabadnapokra vonatkozó feltételünk szintén teljesülni fog, mivel ha tekintjük a legrövidebb hónapot az év során, mely 28 napos, akkor könnyen belátha-

tó, hogy az a szabályunk, hogy egy hónap alatt legalább 4 szabadnapot kötelesek vagyunk kiosztani, szintén teljesülni fog a 6-1 mintánk esetén. Ezeket a szabadnapmintákat dolgozónként eltoljuk, mellyel egyenletesen elosztjuk a szabadnapterhelést az emberek között, így biztosítva azt, hogy hatékony legyen a szabályokat minimálisan teljesítő szabadnapok elhelyezkedése. Egy ilyen szabadnapmintára ad példát egy hetes tervezési időszak és 7 buszvezető esetén az 1. táblázat:

	1. nap	2. nap	3. nap	4. nap	5. nap	6. nap	7. nap
1. buszvezető	0	0	0	0	0	0	1
2. buszvezető	1	0	0	0	0	0	0
3. buszvezető	0	1	0	0	0	0	0
4. buszvezető	0	0	1	0	0	0	0
5. buszvezető	0	0	0	1	0	0	0
6. buszvezető	0	0	0	0	1	0	0
7. buszvezető	0	0	0	0	0	1	0

1. táblázat Szabadnapminták

Természetesen nem ez lesz a végleges mintánk a feladat végső megoldásában, mivel a buszvezetők számára a 6-1-es szabadnapkiosztás túlságosan megterhelő lenne, azonban a szabadnapmintában meghatározott szabadnapok az adott embereknek mindenféleképpen ki lesznek osztva. Ez azt jelenti, hogy a fázisban legenerált szabadnapmintáknak a fix részét határozzuk meg előre, míg a többi szabadnap elhelyezkedése az optimalizálás során fog kialakulni. Az algoritmus utolsó fázisában miután hozzárendeltük a dolgozókat a műszakokhoz, a lokális kereső algoritmusunk folyamán a rögzített szabadnapok mellé újabb szabadnapok fognak létrejönni. Így a végső megoldásunk során a ledolgozott napok száma az elvárt munkaidő alapján fog kialakulni. Miután a kezdeti szabadnapmintákat sikeresen legeneráltuk, nekikezdhetünk a gráf színezésének.

2.3 Gráf felépítése a műszakok alapján

Az alapötlet abból indul ki, hogy legyen $G = (V, E)$ egy olyan gráf, amely ha S az elvégzendő műszakok halmaza és S_i jelenti az i . műszakot, akkor a gráfunk minden $v_i \in V$ csúcsa reprezentál egy S_i műszakot. Tehát a tervezési időszakban előforduló valamennyi műszak legyen a gráfunk egy-egy csúcsa. Két csúcs között akkor és csak akkor van él, ha az általuk összekötött csúcsok által reprezentált műszakot a definiált szabályok szerint nem végezheti el egyazon ember. Az így kapott G gráf egy intervallum gráf [14] melynek színezésére ismert polinom idejű algoritmus, azonban az első fejezetben ismertetett korlátozások miatt a színezési feladatunk NP teljessé válik [15]. A színezés során legyen $c(v)$ a gráf v csúcsához

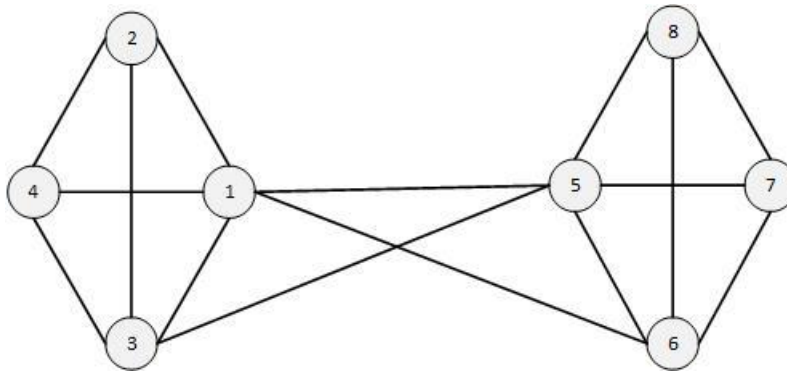
rendelt szín, és feleltessük meg ezeket a színeket a feladatunkban szereplő egyes buszvezetőknek. Legyen C a buszvezetők halmaza, és jelentse $f = S \rightarrow C$ azt a hozzárendelést, melynek során a műszakokat lefedjük a járművezetőkkel.

A gráf építésének megértéséhez vegyünk példának egy a feladat megoldása során használt bemenet részletét, és építsük fel a hozzá tartozó gráfot. Legyen az algoritmus egy kezdeti bemenete a következő táblázat ahol „id” a műszak azonosítója, a dátum, hogy mely napon kell elvégezni az adott műszakot, a kezdet és a vég, hogy az adott nap mely percében kezdődik, illetve végződik a műszak, és végül a munkaidő jelenti a szolgálati időből munkával töltött elszámolt időt percben.

id	dátum	kezdet	vég	munkaidő
1	2010.01.01	889	1 411	522
2	2010.01.01	540	984	444
3	2010.01.01	714	1135	421
4	2010.01.01	237	713	476
5	2010.01.02	396	850	454
6	2010.01.02	454	992	538
7	2010.01.02	702	1 138	436
8	2010.01.02	814	1 327	394

2. táblázat Példa bemenet a problémára

Építsük fel a 2. táblázatban adott bemenethez tartozó gráfot. Ehhez használnunk kell az első fejezetben tárgyalt szabályok közül a pihenőidőre vonatkozó szabályt, amely kimondja, hogy egy ember nem kaphat egy nap két műszakot illetve a műszakjai között legalább 12 óra pihenőidőt kell hagynunk. E feltétel betartásával a kapott gráf az 1. ábrán látható.



1. ábra Példa a felépített gráfra

Megfigyelhető, hogy mivel egy ember nem kaphat egy nap két műszakot, ezért minden nap a gráfunkban egy-egy teljes részgráf lesz. Ez alapján elmondható, hogy legalább akkora klikk biztosan van benne, amennyi a tervezési idő alatt a legtöbb műszakkal rendelkező nap műszakszáma. Természetesen előfordulhat, hogy például adott a nap során egy olyan legkésőbbi műszak, amely minden másnapi műszakot kizár, és így létezik egy ennél nagyobb klikk is a gráfban. Tehát az algoritmus során először felépítjük az előbb bemutatott gráfot.

2.4 Gráf színezése

A gráfszínezés során az egyes színek fogják jelenteni a járművezetőinket. A kezdeti emberszám becslésünk általában helyes, azonban előfordulhat bizonyos szélsőséges, túl rövid munkaidővel rendelkező bemenetek esetén, hogy nem színezhető ki adott számú színnel a gráfunk. Ezért képessé kell tennünk a színezési algoritmusunkat, hogy szükség esetén képes legyen a becsült emberszám mellé új színt is behozni. A megoldás során a színezésünk alapját a DSATUR algoritmus képezte, melynek részleteibe Brélaz 1979-es cikke nyújt kellő betekintést [16]. A hatékonysága és a gyorsasága miatt esett a választás erre az algoritmusra, és mert képesnek kell lennünk új emberek bevezetésére a becsült emberszámon felül. Az algoritmus egy mohó módszer, mely a csúcsokat szaturáció szerinti sorrendben színezi. Egy csúcs szaturációján értjük azt, hogy az adott csúcs szomszédjai között hány darab különböző szín fordul elő. Például ha egy csúcsnak 2 db sárga és 5 db piros szomszédja van, akkor a szaturációja 2.

Esetünkben annyival módosítottam az algoritmust, hogy a gráf színezése során a becsült emberhalmazt használja fel. Tehát tulajdonképpen egy k -színezést végzünk úgy, hogy szükség esetén új szín behozására is van lehetőségünk. Az algoritmus pseudokódja a következő:

Amíg van színezetlen csúcs

- i. *Kiválasztjuk a legnagyobb szaturációval rendelkező még kiszínezetlen csúcsot.*
 - ii. *Egyenlő szaturációk esetén a nagyobb foksámú csúcsot választjuk.*
 - iii. *A csúcsnak az aktuálisan legjobban alulfoglalkoztatott szint adjuk a lehetséges színek közül.*
 - iv. *Ha a lehetséges színek halmaza üres, bővítjük úgy, hogy egy új embert hozunk be a színek közé.*
 - v. *Frissítjük a szaturációját a szomszédos csúcsoknak.*
 - vi. *repeat.*
-

Tehát az algoritmus végigmegy a gráf csúcsain szaturáció és azon belül foksám szerint, majd minden lépésben ad egy szint az adott csúcsnak úgy, hogy a lehetséges színek közül mohó módon azt választja, amelyik aktuálisan a legkevesebb munkaidővel rendelkezik. Ezáltal az algoritmus próbálja egyenletesen elosztani a munkaidőt a járművezetők között. Ez az elosztás a színek között azért fontos, hogy az algoritmus későbbi részeinek átadott kezdeti megoldás a lehető legjobb legyen a költség szempontjából. A színezés során figyelembe kell vennünk, hogy adott szín adott napon az előre legenerált szabadnapminta alapján dolgozhat-e, továbbá, hogy teljesíti-e az egyéb feltételeket, mint például a heti munkaidőre vonatkozó szabályt, vagyis ha megkapja az adott műszakot, túl fogja-e lépni a heti maximális 48 óra munkaidőt. Ha a feltételek mellett nem lehetséges szint találni az adott lépésben a csúcsnak, akkor az algoritmus bővíti a kezdeti emberszám becslés által megadott kezdeti halmazt egy új emberrel úgy, hogy a legelső olyan buszvezetőt választjuk, aki az adott műszakot el tudja végezni. Fontos látnunk, hogy azzal, hogy egy kezdeti becsült halmazt alkalmazunk, a munkák eloszlása is kedvezőbb lesz a kezdeti megoldásunkban annál, mintha minden lépésben folyamatosan új színeket vezetnénk be, mivel így a munkaidők és a költség szempontjából egy viszonylag hatékony kezdeti megoldást kapunk.

Miután kiszíneztük az gráfunkat, rendelkezünk egy kezdeti megoldással, mely teljesíti a megadott feltételeket és minden műszak ki van osztva egy járművezető számára. Mivel a műszakok munkaidejét ismerjük, kiszámolhatjuk a kezdeti kiosztás esetén minden egyes emberre a munkaidejét. A költségünk az alul- és túlfoglalkoztatottság mértéke lesz, melyet ismerve az egyes emberek munkaidejét, a következőképpen számolhatunk ki:

$$\text{foglalkoztatottság} = \text{végzett összmunkaidő} - \text{munkanap} * 480$$

Egy ember foglalkoztatottsága egy negatív vagy egy pozitív szám lesz, annak függvényében, hogy az adott buszvezető alul- vagy felülfoglalkoztatott. Egy járművezető munkaideje segítségével a foglalkoztatottságát esetünkben úgy kapjuk, hogy a munkaidejéből kivonjuk a tervezési időszak alatti munkanapok számát, megszorozva az egy napra elvárt munkaidővel. Esetünkben $21 \cdot 480 = 10\,080$ mivel 21 munkanapunk van a tervezési időszak alatt és napi 8 órát, vagyis 480 percet várunk el minden egyes buszvezetőtől. Így a dolgozók foglalkoztatottsága az alábbiak szerint alakul (3. táblázat):

	Munkaidő	Foglalkoztatottság
1. buszvezető	10 010	-70
2. buszvezető	10 298	218
3. buszvezető	9 980	-100
4. buszvezető	9 971	-109
5. buszvezető	10 355	275
6. buszvezető	10 157	77
7. buszvezető	9 746	-334

3. táblázat A buszvezetők foglalkoztatottsága

Látható hogy a táblázatban az első buszvezetőnk 70 perccel alul van foglalkoztatva, míg a másodiknak 210 perccel több munkaideje van az elvártnál. A következő fázishoz szükséges kezdeti költség innen már könnyedén kiszámolható úgy, hogy minden egyes dolgozónak vesszük a költségszorzóval megnövelt foglalkoztatottságának az abszolút értékét. Ez esetünkben a következőképpen néz ki. Tekintsük azt az esetet, amikor mind a felülfoglalkoztatottság, mind az alulfoglalkoztatottság költségszorzója egyenlően egy. Ekkor a kezdeti megoldásunk költsége a példánál: 1 183. Mivel az egyszerű gráfszínezés általában nem ad elég jó megoldást, és a költség alakulása bizonyos bemenetek esetében nem jelezhető előre, ezért lokális kereséssel alapuló algoritmussal végzünk átszínezéseket a megoldásunkon.

2.5 A gráf átszínezése Tabu Search segítségével

A következő fejezetben bemutatom a problémára alkalmazott Tabu Search módszert, melynek segítségével a kezdeti megoldásunkat felhasználva, hatékony megoldást adtam a problémára. Az eljárást Glover nevéhez kötik [17–19] és napjainkban az egyik legelterjedtebb lokális kereső algoritmus.

Legyen SL a gráfszínezésünk összes lehetséges megoldásainak halmaza, és minimalizáljuk a célfüggvényünket az SL -en. Bármely $sl \in SL$ lehetséges megoldásnak definiáljuk a szomszédságát, mely legyen $N(sl)$. Szomszédságon egy olyan megoldást értünk, amely elérhető sl -ből egy szomszédságként definiált átszínezéssel. Az algoritmus ilyen $sl_0, sl_1, \dots, sl_n$

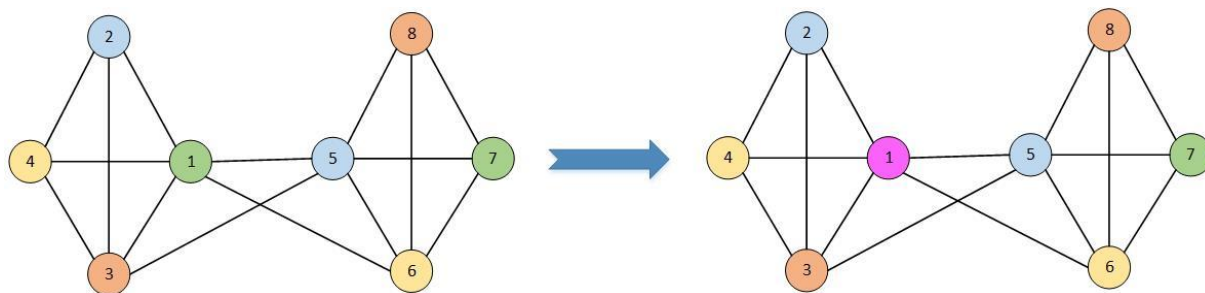
megoldásokat látogat meg a futása során, ahol sl_0 a kezdeti megoldásunk, melyet a gráfszínezés ad, és $sl_{i+1} \in N(sl_i)$. A szomszédság választási módszerek közül a következők lehetségesek:

1. *Best Fit*: Egy $sl \in SL$ megoldás összes szomszédjai közül a lehető legjobbat választjuk egy lépésben. Esetünkben ez azt jelenti, hogy végigmegyünk egy megoldás összes szomszédján és közülük a lehető legjobbat választjuk, ami jobb, mint az aktuális költségünk.
2. *First Fit*: Egy $sl \in SL$ megoldás összes szomszédjai közül a legelsőként megtalált jobbat választjuk. Ez azt jelenti, hogy egy színezés során a legelső olyan szomszédot választjuk, ami jobb az aktuális megoldásnál, vagyis ebben az esetben nem járjuk be az adott állapot összes szomszédját.

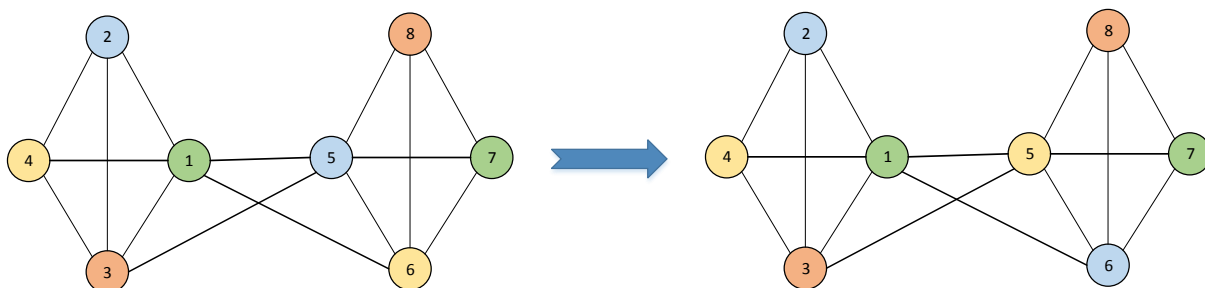
Egy lépésben választott szomszédságokat a tabulistában tároljuk, mely legyen TL . A TL listánkban lévő lépéseket nem hajthatjuk végre egy adott ideig. Az aktuális sl megoldás $N(sl)$ szomszédjainak halmazát a következő szomszédságokat definiálva kaphatjuk meg:

1. *1 csúcs átszínezése*: Az újraszínezés során a gráf egy csúcsának új színt adunk. Vagyis egy műszakot egy buszvezetőtől elveszünk és odaadjuk egy másiknak. Azt hogy melyik járművezetőnek adjuk oda a műszakot, *First Fit* módszerrel úgy választjuk meg, hogy a legelső olyan buszvezető számára osztjuk ki, mely esetén a költség csökken, és a szabályok betartása mellett képes elvégezni a műszakot. A legjobban túlfoglalkoztatott buszvezetőktől próbálunk elsősorban elvenni műszakokat, és azt egy alulfoglalkoztatott embernek odaadni. Miután végrehajtottuk a csúcs újraszínezését, hozzáadjuk a TL tabulistához.
2. *Két csúcs színének cseréje*: A csere során a gráf két csúcsának a színét megcseréljük. Ez azt jelenti, hogy két járművezető között műszakot cserélünk úgy, hogy mindkettőjük számára olyan műszakot adunk, melyet a szabályok betartása mellett képesek elvégezni. Itt szintén *First Fit* módszerrel választjuk a következő szomszédságot. Itt a legjobban túlfoglalkoztatott és az alulfoglalkoztatott buszvezetők között próbáljuk a cseréket végrehajtani. Miután végrehajtottuk a cserét, hozzáadjuk a TL tabulistához.

Az alábbi két ábra a két szomszédságtípusra mutat példát. Az 2. ábrán az látható hogy az 1-es csúcs, melyet kezdetben egy túlfoglalkoztatott dolgozó végez, az átszínezés során átkerül egy másikhoz, aki alulfoglalkoztatott. A 3. ábrán az látható, ahogy egy túlfoglalkoztatott és egy alulfoglalkoztatott buszvezető műszakot cserél úgy, hogy javuljon a költség.



2. ábra 1 csúcs újraszínezése (1-es csúcs)



3. ábra 2 csúcs színének cseréje (5-ös és 6-os csúcs)

A legelső jobb szomszédság választása, azaz a First Fit azért célravezetőbb annál, mintha mindig a globálisan legjobb szomszédot választanánk ki, azaz a Best Fitnél, mert egy lépésben az összes szomszéd vizsgálata a keresési tér mérete miatt túl sok ideig tart. A megoldás során ezért az aktuális megoldás szomszédjai közül mindig *First Fit* módszerrel választunk. Így az algoritmus kódja a következő:

-
1. Legyen s_0 a kezdeti megoldás (gráf színezése)
 2. $TL = \{\}$ és $s = s_0$ és $best = s_0$
 3. *while* kilépési feltétel
 - a. $s^* := N(s)$ szomszédjai közül a legelső jobb ahol $költség(s^*) < költség(s)$ és $move(s, s^*) \notin TL$
 - b. Ha $költség(best) > költség(s^*)$ akkor $best = s^*$
 - c. Ha nem találunk aktuálisnál jobb megoldást, akkor a lehető legjobbat választjuk.
 - d. $TL := TL \cup move(s, s^*)$
 - e. Régi elemek törlése TL -ből
 - f. $s := s^*$
-

A tabulistán az elemek meghatározott időt töltenek. Az algoritmus futása során, ha ez az idő letelik, akkor az elem törlődik a tabulistáról és így újra léphetővé válik az adott szomszédság. Kilépési feltételként több módszer is alkalmazható. Gyakran használtak az iteráció számon

alapuló megállási feltételek, továbbá időkorláton alapuló feltételek. Használható olyan megállási feltétel is a megoldás során, amely azt határozza meg, hogy bizonyos iteráció számon belül minimálisan mennyit javuljon a költség. Esetünkben olyan megállási feltételt definiáltam, hogy az algoritmus álljon meg, ha az aktuálisan megtalált legjobb költségénél bizonyos iteráción belül nem talál jobb megoldást.

3 Eredmények

A heurisztikus megoldó módszer tesztelése során adott eredményeket összevetettem az egészértékű programozási feladat által adott eredményekkel viszonylag kicsi (napi műszakok maximális száma 50) bemenet esetén. Az algoritmus MATLAB környezetben lett lekódolva, az egészértékű programozási feladat megoldása során a modell C++ nyelven lett felépítve és IBM CPLEX szoftvert használtam a megoldására. A teszteket számítógépen futtattam, melyek paraméterei a következők:

- Processzor: Intel Core I7 970 3.2Ghz
- Memória: 14Gb
- Operációs rendszer: Microsoft Windows 7 Enterprise 64bit

A tesztfuttatásokhoz az általam generált inputokat használtam fel, melynek tulajdonságai a következők:

- A műszakok napon belüli elhelyezkedése normális eloszlású.
- A műszakok munkaideje átlagosan 7 és 9 óra között helyezkedik el.
- A műszakok szolgálati ideje 6 és 10 óra között helyezkedik el.

A teszteléshez továbbá felhasználtam valós példát is, mely a Tisza Volán Zrt. szegedi adatbázisához tartozik, és amely a szegedi helyi közlekedéshez tartozó járatokat, műszakokat és az általuk felhasználható buszvezetőket tartalmazza.

Az általam használt bemenetek esetén 1 hónapos tervezési időszakkal számoltam és adtam meg rá a sofőrök ütemezését. A felhasznált teszteseteket 5 méretbe sorolhatóak, mely a tervezési időszak alatti napi maximális műszakszámot tükrözi. Minden generált bemenetből 5 db-ot fogok tesztelni és a futtatások során 25, 50, 75, 100 és 200 méretű bemeneteken tekintjük az eredményeket. A kisebb 25, 50 méretű generált és a Tisza Volánt Zrt.-től kapott valós bemeneteken az egészértékű programozási feladat által adott eredményt használom referenciaként, a nagyobb 75, 100 és 200 méretű inputok esetén az optimum nem ismert, mivel az egészértékű programozási feladat ilyen méretű bemenetek esetén belátható időn belül nem ad megoldást. Ezért számoltam egy elméleti alsó korlátot, melynél adott emberszám mellett jobb eredményt nem tudunk elérni. A tesztelés folyamán a becsült emberszámmal minden esetben megoldható volt a feladatot. Minden méretből 5 darab bemeneten teszteltem a programot. Az algoritmus megállási feltételeként azt állítottam be, hogy álljon meg, ha az aktuálisan általa megtalált legjobb megoldásnál 200 iteráció után sem talál jobb megoldást. Az alábbi táblázat

(4. táblázat) az algoritmus és az egészértékű programozási feladat által produkált futási eredményeket tartalmazza 25 méretű bemeneteken. A bemenet hétköznaponként 25 műszakot, míg a hétvégéken, szombaton 20 illetve vasárnap 15 műszakot tartalmazott.

Név	Alsó korlát	Idő IP	Költség IP	Idő heurisztika (sec)	Költség heurisztika	Gap IP (%)
gen25_1	4807	18.77	4900	9.9842	4807	1.89%
gen25_2	4658	495.2	4691	9.0530	4658	0.7%
gen25_3	1825	154.8	1855	10.9028	1825	1.62%
gen25_4	4371	22.96	4457	8.8626	4371	1.92%
gen25_5	4815	18.11	4911	9.0196	4815	1.95%

4.táblázat: IP és heurisztika eredménye 25 méretű bemeneteken

A megoldás során megállási feltételként 2%-os eltérés volt megengedett az egészértékű programozási feladatnak. Ez azt jelenti, hogy a becsült optimumnál 2%-nál rosszabb eredmény esetén állt meg a megoldó, vagyis az egészértékű megoldásunk ennyire van maximum az alsó korláttól. Az IP megoldó eltérései az optimumtól így 0 és 2% között helyezkednek el. A táblázatból látszik, hogy viszonylag kicsi input esetén egy-két példától eltekintve mind a heurisztika mind az egészértékű programozási feladat elfogadható futásidőket produkált.

Név	Alsó korlát	Idő IP	Költség IP	Idő heurisztika (sec)	Költség heurisztika (sec)	Gap IP (%)
gen50_1	3359	875.1	3465	33.2	3359	1.72%
gen50_2	2581	844.8	2581	122.1	2581	0%
gen50_3	3000	1102.34	3047	20.23	3000	1.54%
gen50_4	1671	1252.78	1696	17.86	1671	1.4%
gen50_5	1925	1529.56	1960	23.76	1925	1.7%

5.táblázat: IP és heurisztika eredménye 50 méretű bemeneteken

Ahogy az 5. táblázatban látható, nagyobb méretű inputokon az IP megoldó futásideje jelentősen megnövekedett, azonban a becsült optimumtól való 2%-os határon belüli megoldást még így is talált. A gen50_2 bemenet esetén az IP megoldó el is érte az elméleti alsó korlátot. Megállapítható, hogy a heurisztika futásideje egy kivétellel, nem növekedett jelentősen az előző 25-ös méretű inputokon tapasztalt futásidőhöz képest, és minden esetben elérte az elméleti alsó korlátot. Az 50 méretű bemenetek hétköznaponként 50, míg hétvégéken 45 és 40 műszakot tartalmaztak.

Név	Alsó korlát	Idő heurisztika (sec)	Költség heurisztika	Kezdeti költség
gen75_1	2861	217.59	2861	18052
gen75_2	4051	54.98	4051	18834
gen75_3	3809	58.96	3809	19438
gen75_4	90	72.51	100	14666
gen75_5	4582	335.76	4582	17677

6. táblázat: heurisztika eredménye 75 méretű bemeneteken

A 6. táblázat a heurisztika futási eredményeit tartalmazza 75 méretű bemenetek esetén. Az algoritmus a kezdeti emberszám becslés alapján megközelítőleg 95-110 emberből oldotta meg a feladatokat. Megállapítható hogy a futásideje az algoritmusnak növekedett az előzőekhez képest. A heurisztika egy esetben képtelen volt elérni az elméleti alsó korlátot, mivel az általa legjobb megoldásként adott megoldás után 200 mélységben nem talált jobb megoldást, így megállt. Elmondható azonban, hogy a gen75_4 bemenet esetén, mivel az összes munkaidő eltérése az emberszámhoz tartozó elvárt munkaidőhöz képest kicsi, valószínűleg nem érhető el az alsó korlát. A 75 méretű bemenetek hétköznaponként 75, míg hétvégén 70 illetve 65 műszakot tartalmaztak.

Név	Alsó korlát	Idő heurisztika (sec)	Költség heurisztika	Kezdeti költség
gen100_1	3261	124.53	3261	19752
gen100_2	987	145.23	987	22139
gen100_3	1663	343.03	1663	20846
gen100_4	3995	429.51	3995	20959
gen100_5	3616	130.01	3616	23086

7. táblázat: heurisztika eredménye 100 méretű bemeneteken

A 100-as méretű generált inputokat jellemzően 145-155 emberrel lehetett optimálisan megoldani az emberszám becslés alapján. Ez a méret már egyes gyakorlati területeken megközelíti a probléma valós méretét. Megfigyelhető, (7. táblázat) hogy amint nő az emberszám, úgy nő a gráfszínezési algoritmus kezdeti megoldásának is a költsége, tovább növelve így az átszínezési lokális keresés futásidejét. Az algoritmus az elméleti alsó korlátot minden esetben elérte, azonban megfigyelhető, hogy egyes esetekben ez jóval tovább tartott. A 100 méretű bemenetek hétköznaponként 100, míg hétvégénként 90 illetve 85 műszakot tartalmaztak.

Név	Alsó korlát	Idő heurisztika (sec)	Költség heurisztika	Kezdeti költség
gen200_1	3564	1540.65	3564	44562
gen200_2	4895	1106.94	4895	42287
gen200_3	3771	1776.34	3771	44836
gen200_4	2930	516.75	3930	43455
gen200_5	2029	1117.01	2029	45758

8. táblázat: heurisztika eredménye 200 méretű bemeneteken

Az 8. táblázatban látható, hogy a heurisztika futásideje a 200-as méretű inputok esetén jelentősen megnövekedett, azonban tekintve, hogy a megoldáshoz 295-305 emberre volt szükség a kezdeti emberszám becslés alapján, így ez a futásidő elfogadható. Az elméleti alsó korlátot minden esetben elérte, azonban elmondható, hogy a kezdeti megoldásunk költsége viszonylag magas lett. A megoldás sebessége - ahogy az a táblázatból is látszik - ilyen méret esetén már nincs összefüggésben a kezdeti megoldás költségével. Valószínűleg attól függ az algoritmus sebessége, hogy mekkora költségjavulással járó szomszédságokat választ, illetve milyen hamar találja meg azokat a szomszédokat, melyek javítanak az aktuális költségen.

Név	Alsó korlát	Idő IP	Költség IP	Idő heurisztika (sec)	Költség heurisztika	Gap IP (%)
volan	4897	21418.56	5012.87	17.86	4897	2.29%

9. táblázat: IP és heurisztika legjobb eredménye valós bemeneten

A valós, Volán Zrt-től kapott bemeneten tesztelve az algoritmust elmondható, hogy az IP megoldó 2%-os Gap esetén nem adott megoldást, hanem csak 2.29%-os megoldást volt képes elérni. Az IP által talált legjobb megoldás és az ahhoz tartozó idő látható a 9. táblázatban. A heurisztika a valós bemeneten hatékonyan viselkedett, melynek oka az lehet, hogy a valós példákban a műszakok munkaideje közötti különbség jóval nagyobb, mint a generáltaknál, vagyis a lokális kereső algoritmus az átadások és a cserék folyamán egy iterációban nagyobb mértékben tud javítani a költségen, mint a generált bemenetek esetén.

Összefoglalás

A műszakkiosztási feladat napjainkban egyre inkább előtérbe kerül, ahogyan egyre több embert kell kezelniük az állami intézményeknek és a nagyvállalatoknak. Ha ezek hatékonyan osztják ki a munkát a dolgozók számára, jelentős költségbeli megtakarítást érhetnek el. A céloom a dolgozat során az volt, hogy a közlekedési társaságoknál fellelhető járművezető műszakkiosztási problémára adjak egy hatékony algoritmust, mely használható alternatívát kínál az egészértékű programozási feladat sokszor valós környezetben elfogadhatatlan futás-ideje szempontjából. A probléma megoldására egy kétlépéses gráfszínezési algoritmust dolgoztam ki, mely a műszakok adott emberszám melletti elosztásában igen hatékony, és képes valós méretű bemenetek esetén is a feladat költséghatékony megoldására.

A dolgozatomban áttekintettem a műszakkiosztási feladat során felmerülő problémákat, megadtam az alapp probléma buszvezetők ütemezése szempontjából fontos szegmenseit, és megvizsgáltam a szakirodalom által a problémára adott módszereket. Bemutattam, hogy a való életben a közlekedési társaságok életében hol helyezkedik el a probléma. A feladat megoldása során a szakirodalom által adott módszerekre építkezve adtam egy heurisztikus módszert, melynek során a problémát gráfként reprezentáltam, és egy emberszám becselő módszer segítségével előre meghatározott emberhalmaz mellett adtam egy kezdeti színezést a gráfunkhoz, melyet végül az alul- és túlfoglalkoztatottság csökkentése érdekében egy Tabu Search algoritmus segítségével módosítottam, mely következtében a kezdeti költség tovább csökkent. A módszer előnye a rugalmassága, ami elősegíti, hogy újabb vállalatfüggő korlátozó tényezők is könnyen beépíthetők legyenek a modellbe.

További cél az algoritmus párhuzamosítása és implementálása C/C++ környezetben, továbbá grafikus processzorra történő optimalizálása, mellyel az algoritmus valószínűleg tovább gyorsítható. Mivel az alul- és túlfoglalkoztatottság csökkentése szempontjából az algoritmus kellően hatékony, céloom az emberszám kialakítás azon eseteinek fejlesztése, mikor szélsőséges bemenetek esetén az emberszám becslése nem sikeres, így a minimálisan a szabályoknak megfelelő 6-1 kialakítású szabadnapmintákon kívül, és az intervallum gráf színezésének polinomiális tulajdonságán kívül más nem garantálja az emberszám helyes megválasztását. Ha mindezek sikeresen végrehajthatóak, akkor a műszakkiosztási problémára használt heurisztikus módszerekre egy valós alternatívát lehetne nyújtani.

Irodalomjegyzék

- [1] J.K. Lenstra, A.H.G.R. Kan, Complexity of vehicle routing and scheduling problems, *Networks*. 11 (1981) 221–227.
- [2] J. Arabeyre, J. Fearnley, F. Steiger, W. Teather, The airline crew scheduling problem: A survey, *Transp. Sci.* 3 (1969) 140–163.
- [3] M. Padberg, R. Rushmeier, K. Hoffman, Recent advances in exact optimization of airline scheduling Problems, *Tech. Rep.* (1995).
- [4] A. Wren, J. Rousseau, Bus Driver Scheduling: An Overview, *Comput. Aided Sched. Public Transport.* (1995) 173–187.
- [5] V. Árgilán, A. Tóth, B. Dávid, C. Kemény, G. Pongrácz, Greedy heuristics for driver scheduling and rostering, *Proc. 2010 Mini-Conference Appl. Theor. Comput. Sci.*, (2010) 101–108.
- [6] G. Pongrácz, C. Kemény, Mohó heurisztika humánerőforrás ütemezésére TDK dolgozat (2010).
- [7] D. Sitompul, S.U. Randhawa, Nurse scheduling models: a state-of-the-art review, *J. Soc. Health Syst.* 2 (1990) 62–72.
- [8] E. Burke, P. De Causmaecker, G. Berghe, H. Van Landeghem, The State of the Art of Nurse Rostering, *J. Sched.* 7 (2004) 441–499.
- [9] V. Mehrotra, Ringing up big business, *OR/MS TODAY*. 8 (1997) 18–25.
- [10] T. Grossman, D. Samuelson, S. Oh, T. Rohleder, Call centers, *Technical Report*, Haskayne Sch. Business, Univ. Calgary. (1999).
- [11] M. Mesquita, M. Moz, A. Paias, J. Paixão, M. Pato, A. Respício, A new model for the integrated vehicle-crew-rostering problem and a computational study on rosters, *J. Sched.* 14 (2010) 319–334.
- [12] 1992. évi XXII. TÖRVÉNY a Munka Törvénykönyvéről, VI. fejezet, A munkaidő és a pihenőidő.
- [13] M. Gamache, A. Hertz, J.O. Ouellet, A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding, *Comput. Oper. Res.* 34 (2007) 2384–2395.
- [14] G. Hajós, Über eine Art von Graphen, *Int Math. Nachrichten.* (1957) 11.
- [15] M. Biró, M. Hujter, Z. Tuza, Precoloring extension. I. Interval graphs, *Discrete Math.* 100 (1992) 267–279.
- [16] D. Brélaz, New methods to color the vertices of a graph, *Commun. ACM.* 22 (1979) 251–256.

- [17] F. Glover, Tabu Search: A Tutorial, Interfaces (Providence). 20 (1990) 74–94.
- [18] F. Glover, Tabu Search - Part I, ORSA J. Comput. 1 (1989) 190–206.
- [19] F. Glover, Tabu Search - Part II, ORSA J. Comput. 2 1 (1989) 4–32.



A projekt az Európai Unió támogatásával,
az Európai Szociális Alap
társfinanszírozásával valósul meg.

Jelen kutatást a Szuperszámítógép, a nemzeti virtuális laboratórium című, TÁMOP-4.2.2.C-11/1/KONV-2012-0010 azonosítószerű projekt támogatta az Európai Unió és az Európai Szociális Alap társfinanszírozása mellett.