ALKALMAZOTT MATEMATIKAI LAPOK

A MAGYAR TUDOMÁNYOS AKADÉMIA MATEMATIKAI TUDOMÁNYOK OSZTÁLYÁNAK KÖZLEMÉNYEI

31. kötet

Szerkesztőség és kiadóhivatal: 1055 Budapest, Falk Miksa u. 12.

Az Alkalmazott Matematikai Lapok változó terjedelmű füzetekben jelenik meg, és olyan eredeti tudományos cikkeket publikál, amelyek a gyakorlatban, vagy más tudományokban közvetlenül felhasználható új matematikai eredményt tartalmaznak, illetve már ismert, de színvonalas matematikai apparátus újszerű és jelentős alkalmazását mutatják be. A folyóirat közöl cikk formájában megírt, új tudományos eredménynek számító programokat, és olyan, külföldi folyóiratban már publikált dolgozatokat, amelyek magyar nyelven történő megjelentetése elősegítheti az elért eredmények minél előbbi, széles körű hazai felhasználását. A szerkesztőbizottság bizonyos időnként lehetővé kívánja tenni, hogy a legjobb cikkek nemzetközi folyóiratok különszámaként angol nyelven is megjelenhessenek.

A folyóirat feladata a Magyar Tudományos Akadémia III. (Matematikai) Osztályának munkájára vonatkozó közlemények, könyvismertetések stb. publikálása is.

A kéziratok a főszerkesztőhöz, vagy a szerkesztőbizottság bármely tagjához beküldhetők. A főszerkesztő címe:

Páles Zsolt, főszerkesztő
1055 Budapest, Falk Miksa u. 12.
A folyóirat e-mail címe: aml@math.elte.hu

Közlésre el nem fogadott kéziratokat a szerkesztőség lehetőleg visszajuttat a szerzőhöz, de a beküldött kéziratok megőrzéséért vagy továbbításáért felelősséget nem vállal.

Az Alkalmazott Matematikai Lapok előfizetési ára évfolyamonként 1200 forint. Megrendelések a szerkesztőség címén lehetségesek.

A Magyar Tudományos Akadémia III. (Matematikai) Osztálya a következő idegen nyelvű folyóiratokat adja ki:

1. Acta Mathematica Hungarica,
2. Studia Scientiarum Mathematicarum Hungarica.

Alkalmazott Matematikai Lapok **31** (2014), 1-40.

ÜTEMEZÉSI FELADATOK AZ AUTÓBUSZOS KÖZÖSSÉGI KÖZLEKEDÉS OPERATÍV TERVEZÉSÉBEN: EGY ÁTTEKINTÉS

ÁRGILÁN VIKTOR, BALOGH JÁNOS, BÉKÉSI JÓZSEF, DÁVID BALÁZS, GALAMBOS GÁBOR, KRÉSZ MIKLÓS, TÓTH ATTILA

A közlekedési társaságok számára lényeges szempont költségeik racionalizálása, amit legkönnyebben az operatív költségeik csökkentésével valósíthatnak meg. Ez a járatok, a járművek optimalizált ütemezésével is elősegíthető. Egy ilyen optimalizálási feladat nagyon komplex, ezért a műveletek ütemezését három fázisra bontva tárgyaljuk. Ezek a járműütemezés, a vezetőütemezés és a műszakkiosztás feladatai. Ezek a részfeladatok általában NP-nehéz problémák, ezért az elmúlt évtizedig közepes méretű feladatok optimális megoldása sem volt lehetséges. A számítási sebesség, az alkalmazott modellek és az azokat megoldó algoritmusok olyan jelentősen fejlődtek, hogy lehetővé vált nagyobb méretű feladatok kezelése is. A cikkben áttekintjük az említett részfeladatok megoldására szolgáló legfontosabb módszereket. Tárgyaljuk azok különböző matematikai modelljeit, hatékony megoldási lehetőségeikkel együtt. A speciális korlátozó feltételek szerepét saját tapasztalatainkon keresztül vizsgáljuk. Utalunk a busz- és vezetőütemezési probléma általunk tanulmányozott és bevezetett megoldási módszereire is.

1. Bevezetés

A közösségi közlekedési szolgáltató vállalatok kiadásainak nagy részét az operatív költségek alkotják. Ez jórészt a járműflotta költségéből, a járművek üzemanyag- és karbantartási költségéből, valamint a járművezetők fizetéséből tevődik össze. Ebből következően az operatív költségekben mutatkozó megtakarítás számottevő javítást adhat költségvetésükben. A leggyakrabban használt módszer ezeknek a költségeknek a csökkentésére egy hatékony, számítógéppel támogatott információs rendszer kialakítása és használata. Az IKT (információs és telekommunikációs technológia) fejlődésének köszönhetően napjainkra szinte minden közösségi közlekedési társaság – modern vállalatirányítási környezetet biztosítva – rendelkezik saját információs rendszerrel.

Ezeknek a rendszereknek a fő funkciói az üzleti alkalmazásokon túl (úgymint könyvelés, számvitel stb.) olyan modulokat is tartalmaznak, amelyek

– előkészítik a járművek ütemezését a vállalat által kiszolgált vonalakhoz,

– illesztik a járművek és a járművezetők műszakjainak ütemezését a vonalakhoz,
– képesek a járműflotta nyomon követésére és monitorozására egy-egy nap során,
– jelzik a diszpécsernek a szokatlan eseményeket (meghibásodás, késés stb.),
– nyomon követik a járműflotta járműveinek állapotát,

és más hasonló tevékenységeket.

Az IKT-környezet alapvető háttérként szolgál a hatékony logisztikai irányításra (lásd pl. [56]), ugyanakkor a folyamatban jelen van egy másik elvárás, amely azt kívánja, hogy találjuk meg a rendszer azon részeit, amelyek az operatív, műveleti költségek szempontjából csökkenthetők.

Jónéhány ipari döntéstámogatási eszköz létezik, amely a logisztikai rendszertervezési és optimalizálási feladatok komplett megoldását célozza. A gyakorlatban ennek ellenére kiderül, hogy sok vállalatspecifikus részlet, korlátozó feltétel van jelen, amelyeket az általános rendszerek nem kezelhetnek egységesen, de amelyek fontosak a közlekedési vállalatok számára. Példaként említjük, hogy amennyiben a közlekedési társaság alternatív üzemanyagú járműveket is alkalmaz flottájánál, akkor ezek ütemezésénél figyelembe kell venni a szaknyelven rádiusznak nevezett, egy tankolással megtehető kilométerek számát, amely jóval kevesebb lehet, mint a hagyományos üzemanyagokkal működő járművek futási teljesítménye. Ilyen eseteket vizsgáltak [62]-ben és [53]-ban. Alternatív üzemanyagú járműveknél fontos tényező a tankolás időtartama is. Amíg hagyományos üzemanyagoknál (pl. dízelolaj) a tankolás kb. 5 perc, addig *CNG (compressed natural gas*) vagy más üzemanyagok esetén a feltöltési idő ennek többszöröse is lehet. Ilyen jellegű feladatok kezelését ismertettük néhány cikkünkben (lásd pl. [9]). Azért, hogy megfelelő járműütemezés készüljön, ezeket a feltételeket számításba kell venni a napi ütemezést elkészítő szoftvernél. Legjobb tudomásunk szerint napjaink ütemező szoftverei nem kezelnek ilyen jellegű feltételeket, vagy azokat csak korlátozott mértékben veszik figyelembe.

Követve a közösségi közlekedési szolgáltatás fejlesztésének általános módszereit, egy fejlesztés fő horizontjai a következők:

– *stratégiai tervezés*, ennek fő eleme a buszok, járatok útvonalának meghatározása (buszvonalak kialakítása),
– *taktikai tervezés*, melynek legfontosabb eredménye a menetrend elkészítése és
– *operatív tervezés*, a szolgáltatás ellátásához a buszok és a vezetők, a műszakok ütemezése.

Az irodalomban létezik másféle felfogás is, a feladatokat lehet másképpen csoportosítani. Borndörfer [14] a stratégiai és a taktikai tervezést közös, ún. *„szolgáltatástervezési”* fázisba vonja össze. Megemlít olyan kapcsolódó – a szolgáltatástervezés körébe sorolható – feladatokat (pl. jegyárazás, tarifatervezés és -kialakítás),

amelyek az utazási igényeket is meghatározhatják. Az ilyen jellegű feladatokat – és a hozzájuk tartozó modelleket – ebben a cikkben nem tárgyaljuk.

Annak, hogy mi csak az említett operatív tervezési részfeladatok tárgyalására szorítkozunk, a terjedelmi korlátokon túl az is az oka, hogy a közösségi közlekedési szolgáltató vállalatoknak a többi említett stratégiai és taktikai tervezési feladatra kevés befolyásuk van: ezek általában az állami kormányzati vagy helyi önkormányzati (pl. városi) szervek által meghatározottak. Azon kívül, hogy a járatok milyen sűrűn kövessék egymást, más előírások is vonatkozhatnak a vonalakra, és azon belül bizonyos járatokra is. Például a vonalak kapacitására, vagy a szolgáltatást ellátó járművek speciális tulajdonságaira is lehetnek előírások. Egy jellemző példa az, hogy melyik vonalon milyen időközzel közlekedjenek alacsonypadlós járművek.

Ezen túlmenően viszont szinte teljesen a közlekedési társaságok döntésétől függ, hogy a saját járműflottájukat hogyan ütemezik, és a járművezetőiket hogyan osztják be műszakokba rövid- és hosszútávon egyaránt.

Az már ebből a bevezetőből is látszik, hogy a költségek csökkentése ebben a környezetben egy komplexen összefonódott problémacsokor, amelynek megoldása globális optimalizálási módszereket igényel. Mivel mindezekre egy teljesen összetett megközelítés megvalósíthatatlannak látszik, ezért olyan részproblémákra osztjuk a feladatot, amelyek eléggé izoláltak ahhoz, hogy ezeket a gyakorlatban kezelni tudjuk. Ezek

- a buszvonalak útvonaltervezése,
- a menetrendtervezés,
- az operatív ütemezés.

Megjegyezzük, hogy ennek ellenére a tervezési fázisok és az ütemezési fázis, ha nem is szorosan, de általában mégis hatnak egymásra annak érdekében, hogy globálisan hatékonyabb – vagy legalább lehetséges – megoldás legyen nyerhető.

A tudományos közösség évtizedekkel ezelőtt felismerte, hogy az operatív tervezési problémák optimalizált megoldása nagy kihívást jelentő, érdekfeszítő feladat. Ezeket a feladatokat tárgyalva sok eredmény született (lásd például a [12, 13, 17, 23, 27, 54] cikkeket), amelyek különböző modelleket és eltérő hatékonyságú algoritmusokat tárgyalva foglalkoztak a témával. Hamar kiderült, hogy a legtöbb részprobléma NP-nehéz, ami időben sokszor reménytelenné teszi az optimális – vagy közel optimális – megoldások megtalálását a gyakorlatban felmerülő problémák esetén. Már közepes méretű feladatok – például néhány százezer fős lakosságú városok – esetén is elég összetett, komplex feladattal állunk szemben, amely még összetettebbé válik, ha a részleteket is figyelembe kell vennünk. Még bonyolultabbá teszi a problémát, ha olyan korlátokat is kezelni kell, amelyeket a jogszabályok, a szakszervezetek, az egyéni igények és egyéb gyakorlati feltételek határoznak meg. Ezek például a járművezetők vezetési idejére, munkaidejére, pihenőidejére és munkaközi szüneteire vonatkozó előírások. Ilyen további korlátozó feltételeket adhatnak a telephelyek kötöttségein, kapacitásán és egyéb feltételein

keresztül más gyakorlati részletek, egészen odáig, hogy a vezetőknél megfelelő mértékben legyenek vegyítve a kedvelt és nem kedvelt műszakok. Ehhez az utóbbihoz kapcsolódóan megemlítjük, hogy például Abbink és szerzőtársai 2007-ben publikált cikke [1] tárgyal egy vasúti műszakkiosztási példát.

Nem célunk sem a hálózati útvonal-, sem a menetrend tervezés tárgyalása. Mindezekhez Desaulniers [24] áttekintő tanulmányát javasoljuk, amely jó bevezetés, kiindulópont lehet ilyen jellegű problémák megoldásának tanulmányozásához.

Annak ellenére, hogy felvázoltuk a (buszos) tömegközlekedési rendszerek általános struktúráját, mi a jelen tanulmányban az operatív tervezés egyes fázisait vizsgáljuk. Ezen belül a buszflotta járműveinek ütemezésére, a járművezetők műszakonkénti ütemezésére és beosztására fogunk koncentrálni. A munkaerő ütemezését további részfeladatokra bontjuk: külön fejezetben tárgyaljuk a járművezetők ütemezését és a műszakok kiosztását a járművezetők között.

A cikk felépítése a következő. Mint látni fogjuk, a legutóbbi időkig kidolgozott eljárások a két ütemezési problémát egymás után végrehajtva oldják meg. A járműütemezés eredményeire támaszkodva keresnek optimális – vagy közel optimális – megoldást a vezető-ütemezési feladatok megoldására. Ezért a két ütemezési problémát elválasztva tárgyaljuk a második és a harmadik fejezetben. Azonban azt is látni kell, hogy ma már egyre nagyobb teret nyernek azok a megközelítések, amelyek a jármű- és vezető-ütemezési problémát együtt kezelik, és egyszerre próbálják megoldani. Figyelembe véve az algoritmusokban bekövetkező javításokat és a hardverek rohamosan növekvő számítási kapacitását, nem kétséges, hogy ezek a módszerek is hamarosan beépülnek az újonnan kifejlesztendő interaktív rendszerekbe. Ezért külön fejezetet szentelünk a napjainkban kibontakozó hatékony integrált modellek kidolgozására szolgáló kutatások áttekintésének.

A következő fejezetekben először általánosan – moduláris szerkezetben – vizsgáljuk a feladatokat, majd alfejezetenként részletesen tárgyaljuk az ismert megoldási módszereket.

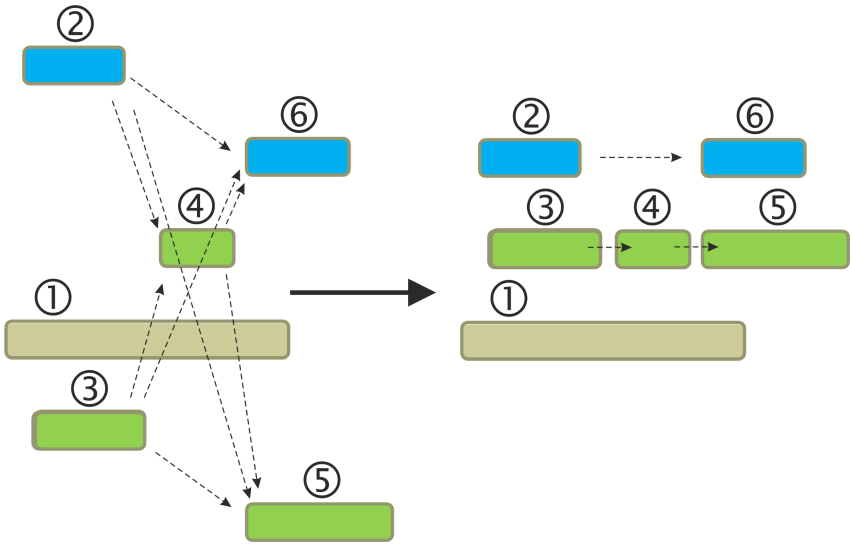
2. A járműütemezési feladat

2.1. A feladatról általában, definíciók, jelölések

A járműütemezési problémánál *(Vehicle Scheduling Problem, VSP)* adott a *járműflotta járműveinek* és a menetrendi *járatoknak* a halmaza. Jelölje ezeket rendre két halmaz, $V = \{v_1, v_2, \ldots, v_{m^*}\}$ és $U = \{u_1, u_2, \ldots, u_{n^*}\}$. Egy V halmazba tartozó minden $v \in V$ jármű azonos típusba tartozik (pl. alacsony padlós és gázüzemű). Ha több – eltérő típusú – járművünk van, akkor a járművek halmazát diszjunkt V_i $(i = 1, \ldots, k)$ részhalmazokra bontjuk. Jelölje továbbá $c_{i,j}$ azt a költséget, amivel az i jármű a j járat által előírt tevékenységet elvégzi. A feladat az, hogy a járműflotta elemeit rendeljük hozzá az egyes járatokhoz úgy, hogy a

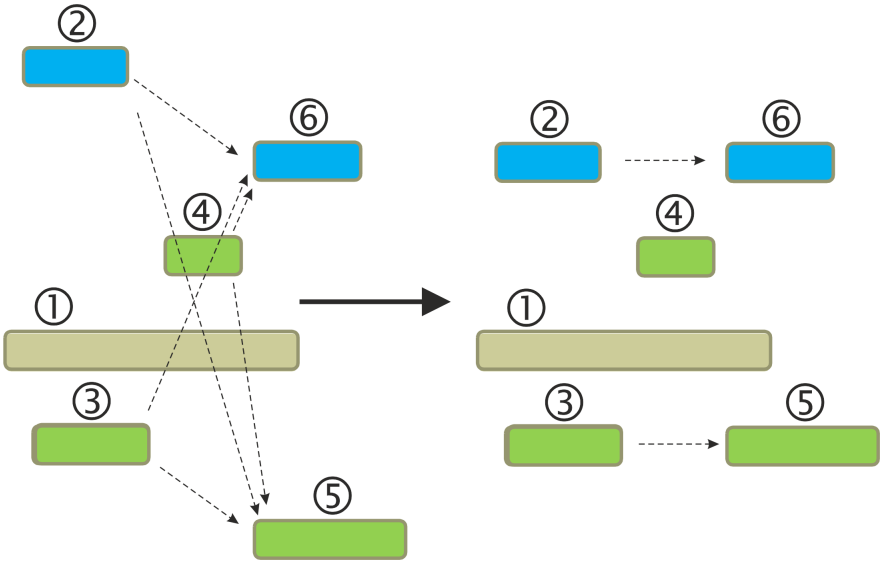
hozzárendelés összköltsége minimális legyen. Az első pillanatban egyszerű hozzárendelési feladatnak tűnő probléma azért bonyolódik meg, mert nem kell minden járművet járathoz rendelni, és egy jármű – a járatok időbeli eltolódása miatt – több járathoz is hozzárendelhető. További korlátozó feltételt jelenthet az, hogy minden járatot pontosan egyszer kell végrehajtani, és – esetleg további feltételek alapján – minden jármű legyen képes végrehajtani azokat a járatokat, amelyekhez a megoldásunk során hozzárendeltük.

Minden $u_i \in U$ menetrendi járatra adott a járat $dt(u_i)$ *indulási* ideje és $at(u_i)$ *érkezési ideje*, valamint $dg(u_i)$ *indulási* és $ag(u_i)$ *érkezési földrajzi helye*. Az u_i és az u_j járatot *kompatibilisnek* (ebben a sorrendben egymás után végrehajthatónak, összefűzhetőnek) nevezzük, ha ugyanaz a jármű képes kiszolgálni egymás után őket, azaz $at(u_i) \leq dt(u_j)$, és $dt(u_j) - at(u_i)$ kisebb, mint a $dg(u_j)$ és $ag(u_i)$ közti távolság megtételéhez szükséges idő. Az 1. és a 2. ábra egy-egy, néhány járatból álló sematikus szituációt mutat. A sematikus ábrákon szaggatott nyilak jelzik a kompatibilis járatpárokat (itt a vízszintes tengely reprezentálja az időt, a földrajzi helyek nincsenek ábrázolva), csak akkor kötünk össze szaggatott nyíllal két járatot, ha egymással kompatibilisek. Az 1. ábrán jelzett sematikus szituációban ilyen módon 3 jármű elég a 6 járat ellátásához, míg a 2. ábráén ehhez már 4 jármű kell.



1. ábra. 6 járatot ábrázoló sematikus feladat. A szaggatott nyilak jelzik az egymással kompatibilis járatokat. Az ábra jobb oldali része azt illusztrálja, hogy a feladat 3 járművel megoldható.

Az ütemezési időszak kezdetén a járművek *depókban* állnak, és az ütemezési időszak végén oda is térnek vissza. Depóba kerülhet egy jármű az ütemezési idő-



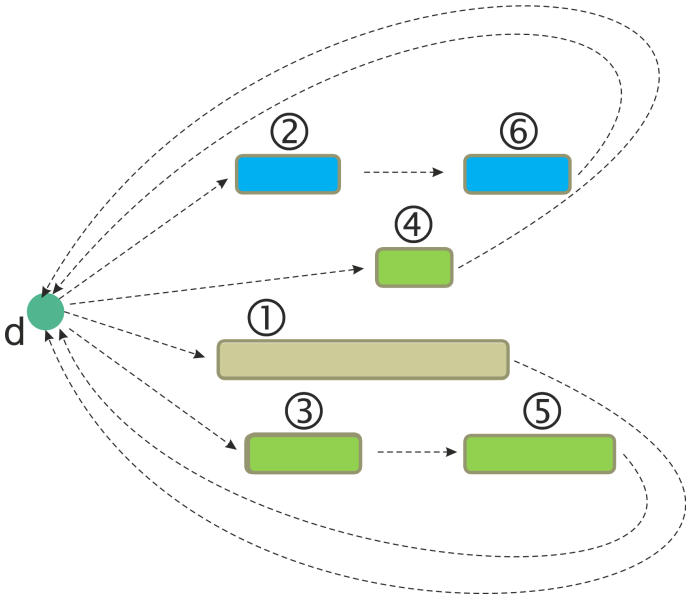
2. ábra. Az 1. ábráéhoz hasonló sematikus szituáció, de itt az ottaninál kevesebb járat kompatibilis egymással, így ez a feladat csak 4 járművel oldható meg.

szak bármely időpontjában, ha hosszabb ideig nem rendelünk hozzá járatot. Depó lehet garázs, parkolóhely vagy telephely, attól függően, hogy a jármű hol parkol.

A depók számától függően beszélhetünk egydepós (*Single Depot Vehicle Scheduling Problem*, SDVSP) és többdepós (*Multiple Depot Vehicle Scheduling Problem*, MDVSP) járműütemezési feladatról. A többdepós esetben az is elő lehet írva, hogy melyik járatot melyik depókhoz tartozó járművek hajthatják végre.

A menetrendi járatokon kívül, amelyek szállítanak utast, megkülönböztetünk olyan járatokat is, amelyek nem szállítanak utast. Ekkor beszélünk *rezsijáratról*. Ez utóbbiak közé tartozik a depóból történő ki- és beállás, valamint két kompatibilis menetrendi járat esetén az első járat érkezési és a második járat indulási földrajzi helye közötti átállás.

Egy jármű *ütemezése* járatoknak egy olyan lánca, amelyben minden egymást követő két menetrendi járat egymással kompatibilis. Általában egy jármű ütemezése a gyakorlatban egy jármű egy napi munkájának előírását jelenti, amit járműműszaknak (szakszóval járműfordának vagy eszközfordának) is nevezünk. A jármű egy érvényes ütemezése egy kiállási járattal kezdődik, és egy beállási járattal végződik. A 3. ábra a 2. ábrának megfelelő sematikus szituáció egy érvényes járműütemezésekké kibővített megoldását mutatja. Itt feltettük, hogy egy depó van (egydepós eset), és a depóból valamennyi járat indulási helyére és érkezési helyéről a depóba vezetnek depókiállási és -beállási járatok).



3. ábra. A 2. ábrán látható sematikus szituáció megoldása egy depó esetén a depó ki- és beállási járatokkal érvényes járműütemezéseket alkot.

A járműütemezés alapfeladata abból áll, hogy adjuk meg a járműflotta járműveinek ütemezését a fenti módon úgy, hogy minden egyes menetrendi járat hozzá legyen rendelve pontosan egy jármű ütemezéséhez, és minden menetrendi járat a megfelelő depók valamelyikéből legyen végrehajtva. (Természetesen lehetnek olyan járművek is, amelyek az adott, (pl. egy napi) ütemezésben nem vesznek részt.) Célfüggvényként kezelhetjük az ütemezésben használt járművek számának a minimalizálását, de definiálható más költségfüggvény is. A cél akkor a költségfüggvény minimalizálása. Ha a költségfüggvény a teljes ütemezés költségét reprezentálja, akkor tartalmaznia kell az egyes depókhoz tartozó átalányköltséget, valamint operatív költséget is. Az átalányköltségen azt a költséget értjük, amely abból keletkezik, hogy egy adott jármű a depóban rendelkezésre áll. Ez lehet a beszerzési, fenntartási, karbantartási stb. költségekből vetített átlag. Az operatív költség általában a megtett távolságokkal arányos, azonban különböző lehet attól függően, hogy melyik depóból származó jármű látja el az adott feladatot. A járatok operatív költsége attól is függhet, hogy rezsi-, vagy menetrendi járatról van-e szó, mivel különböző lehet a kilométerek „egységára”. Több modell képes úgynevezett depókapacitási korlátozó feltételek kezelésére is, azaz lehetséges megoldásnak csak olyan megoldásokat tekint, amely figyelembe veszi minden depó esetén az ahhoz tartozó járművek maximális számát.

A célfüggvényben az ütemezett járművek általános (pl. a fenntartási költségekből adódó) és utazási (menetrendi- és rezsijáratainak) költségei szerepelnek. Némileg könnyíti a feladatot, hogy mint említettük, a költség második komponense általában a megfelelő út hosszával arányos, ugyanakkor a menetrendi járatok és a rezsijáratok kilométereihez eltérő költségek is tartozhatnak. Természetesen a rezsijáratok költségei így függenek a megfelelő földrajzi helyek közötti távolságoktól, következésképpen eltérő lehet a rezsijáratok költsége attól függően, hogy egy adott menetrendi járat mely másik járatot követ.

Több alapvető matematikai modell létezik különböző (SDVSP, MDVSP) feladatok megoldására, amelyeket az előző évtizedekben dolgoztak ki. A következő alfejezetekben áttekintjük a feladatok néhány alapvető megoldási módszerét. A napjainkban talán legszélesebb körűen használt MDVSP-modellekben a probléma egy egészértékű többtermékes hálózati folyam problémaként fogalmazódik meg (lásd [13, 48, 54]). Ebben a modellben az optimális ütemezést egy egészértékű lineáris programozási feladat megoldásaként számíthatjuk ki. A probléma megfogalmazható halmazlefedési vagy halmazpartícionálási feladatként is (lásd például [43, 63]).

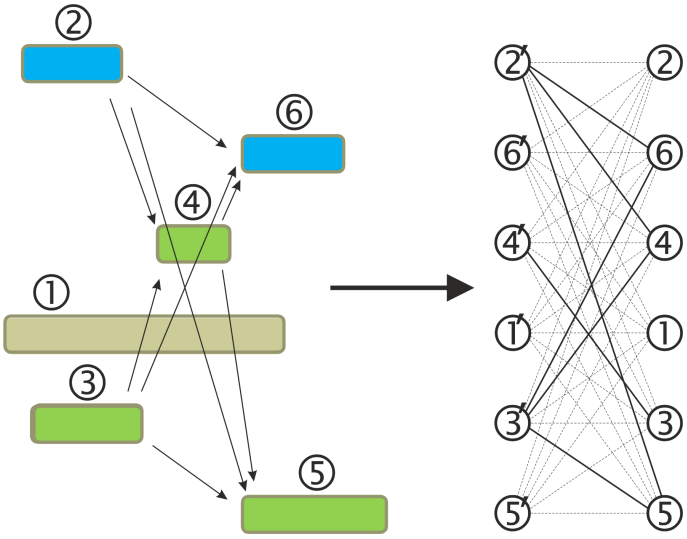
2.2. Az egydepós járműütemezési feladat

Az SDVSP-problémára adott első megoldási módszert Saha [64] publikálta. Az U járathalmaz elemeire egy részleges rendezést definiál, és bevezetésre kerül egy β rendezési reláció, mely szerint akkor szolgálható ki u_2 legálisan u_1 után, ha $ag(u_1) = dg(u_2)$, valamint $at(u_1) \leq dt(u_2)$. A megkötésekből látható, hogy ez a modell nem engedélyezi a különböző járatok között végrehajtott rezsimeneteket, így az itt meghatározott β reláció a fent definiált kompatibilitásnál gyengébb.

Az SDVSP-re több páros gráfon alapuló modellt is publikáltak. Ezekről bővebb áttekintés található Bunte és Kliewer [16] munkájában. Mi az alábbiakban Bertossi és társai [12] modelljét és megoldását ismertetjük, ahol szintén párosítási problémaként modellezik a feladatot. Vegyünk egy $G = (G_1, G_2, E)$ teljes páros gráfot, ahol G_1 és G_2 csúcshalmazok, minden $i \in G_1$ csúcs egy-egy járat érkezésének (érkezési földrajzi helyének), míg minden $j \in G_2$ csúcs egy-egy járat indulásának (indulási földrajzi helyének) felel meg. E jelöli az élhalmazt, valamint $|G_1| = |G_2| = |U| = n^*$, és $|E| = (n^*)^2$. Tegyük fel továbbá, hogy E két E_1 és E_2 részhalmazra osztható, amelyek:

$$E_1 = \{(i,j) \mid u_i \text{ és } u_j \text{ kompatibilis járatpár}\},$$
$$E_2 = E \setminus E_1.$$

A fenti gráf a következő módon értelmezhető (lásd a 4. ábrán egy egyszerű, sematikus feladat illusztrációját). Az $(i,j) \in E_1$ élek két kompatibilis u_i és u_j járat érkezési és indulási földrajzi helyei közti lehetséges rezsijáratot jelképezik,



4. ábra. Az 1. ábrán látható sematikus szituációhoz tartozó páros gráf Bertossi és szerzőtársai SDVSP-modelljében [12]. A folytonos, vastagabb vonallal rajzolt élek az E_1 élhalmazt, a szaggatott, vékonyabb vonallal rajzolt élek az E_2 élhalmazt ábrázolják itt.

míg minden $(i, j) \in E_2$ él két egymás utáni rezsijáratnak felel meg. Ezek közül az első az u_i járat érkezési földrajzi helyéről a depóba, majd a másik a depóból az u_j járat indulási földrajzi helyére. Ezek segítségével látható, hogy a G gráf egy M teljes párosítása egy lehetséges járműütemezést fog adni, melyben a járművek száma $|M \cap E_2|$.

Az (i, j) élekhez $c_{i,j}$ költségeket rendelve az SDVSP feladata megfeleltethető egy minimális költségű teljes párosítás keresésének a G gráfban.

- Ha $c_{i,j} = 1$ minden $(i, j) \in E_2$ esetén, és $c_{i,j} = 0$ egyébként, úgy a feladat megoldásával megkapjuk a járatok teljesítéséhez szükséges minimális eszközszámot.
- Ha $c_{i,j}$ értékei az adott rezsijáratok végrehajtásához szükséges költségek lesznek, akkor a feladat megoldása a minimális operatív költséget adja.

Természetesen az előbbi költségek valamely kombinációja is használható. A gyakorlati életben a probléma kiegészül még a járművek darabszámára adott korláttal. Legyen ez a korlát k. Ekkor egy korlátos párosítási feladatot kapunk, ahol a feladat a minimális költségű párosítást megkeresni a gráfban az $|M \cap E_2| \leq k$ feltétel mellett. A probléma formálisan a következőképpen adható meg: legyen x bináris változók vektora, ahol $x_{i,j} = 1$, ha az (i, j) él az M párosításhoz tartozik, különben $x_{i,j} = 0$, és legyen c a költségvektor. Az X lehetséges megoldások halmazát az

alábbi feltételeknek megfelelő vektorok határozzák meg:

$$\sum_j x_{i,j} = 1, \quad i = 1, 2, \ldots, n^*,$$

$$\sum_i x_{i,j} = 1, \quad j = 1, 2, \ldots, n^*,$$

$$\sum_{(i,j)\in E_2} x_{i,j} \leq k.$$

A célfüggvény

$$min_{(i,j)}\{cx \mid x \in X, \quad x_{i,j} \in \{0, 1\}\}.$$

A fenti feladat minimális költségű hálózati folyamproblémaként is megoldható (lásd [16]). A hálózat konstruálása ekkor annyiban tér el a fent definiált gráftól, hogy további csúcsokat és éleket vezetünk be: a járatokat jelző csúcsokat a hálózatban kettébontjuk a járat indulását és a járat érkezését reprezentáló csúcsra, melyeket egy, a járatot jelző éllel kötünk össze. Ezeknek a járat éleknek az alsó és felső korlátja is 1 lesz, így biztosítva azt, hogy minden járat pontosan egy alkalommal kerüljön teljesítésre. Ennek az a következménye, hogy az ezeken az éleiken felmerülő költségek egy konstans többletként jelentkeznek, ami a feladat célfüggvényére nincs hatással. A járatok csúcsaihoz hasonlóan a depót jelképező csúcs helyett is egy depóindulási, illetve depóérkezési csúcsot vezetünk be. Ezeket egy 0 költségű éllel kötjük össze, mely a hálózat depókörfolyam éle lesz. Ha a fentiek szerint korlátozzuk a rendelkezésre álló eszközök számát, úgy ennek az élnek a kapacitása k lesz. Ezt a fajta formalizmust használva Ahuja és munkatársai bebizonyították, hogy a feladat erősen polinomiális időben megoldható [3].

2.3. A többdepós járműütemezési feladat

A járműütemezési feladat megoldására leggyakrabban használt modell az úgynevezett többdepós járműütemezési probléma (*Multiple Depot Vehicle Scheduling Problem*, MDVSP) modellje. A valós életben a különböző menetrendi járatokra és a járművekre speciális igények vonatkozhatnak. Az eltérő járműtípusok, valamint a járművek tartózkodási helye alapján a járműveket különböző depókba oszthatjuk, így a járatok kiszolgálása – az igényektől függően – különböző depókból történhet.

A többdepós járműütemezési problémát Bodin és szerzőtársai definiálták [13], majd Bertossi és szerzőtársai mutatták meg róla, hogy NP-nehéz feladat [12]. A modell értékét az adja, hogy tartalmazza a valós életbeli járműütemezési probléma legfontosabb komponenseit. A matematikai modellek ismertetése során jelöléseinkben a Löbel által használt terminológiát követjük [54].

A többdepós járműütemezési feladatnál minden menetrendi járat esetén a felhasználó megadja azokat a depókat, ahonnan az adott járat kiszolgálható. A gyakorlatban ez jelentheti például azt, hogy bizonyos járatok csak csuklós busszal

láthatók el, és az is megadható, hogy mely esetben mely telephelyhez tartozzanak. Ezeket az előírásokat a telephely és az állomás elhelyezkedése, valamint a forgalom jellemzői határozhatják meg.

A következőkben az MDVSP különböző megoldási technikáit mutatjuk be. Ezek egy része többtermékes folyamproblémára vezeti vissza a feladatot, és egészértékű programozási (IP) feladatként oldja meg azt. A különbség az alapul szolgáló hálózat felépítésének módszerében rejlik. Ezek alapján megkülönböztetünk kapcsolatalapú és idő-tér hálózati modellt. Egy másik megközelítésben a problémát halmazpartícionálási, illetve halmazlefedési feladatként modellezik. Ezt a modellt elemezték például Ribeiro és Soumis [63], vagy Hadjar és szerzőtársai [43].

2.3.1. A kapcsolatalapú többtermékes hálózati modell

A kapcsolatalapú többtermékes hálózati folyam (*connection-based multicommodity network flow*) modell széleskörűen használt az MDVSP-probléma megoldására. Sok, a témába vágó kutatás fókuszált ennek alkalmazására. Intenzíven tanulmányozták a generált egészértékű programozási feladat megoldási módszereinek fejlesztésében rejlő lehetőségeket. Számos megközelítés alapul heurisztikus, közelítő módszerekre (lásd [23, 54, 57]), míg mások pontos megoldást szolgáltató, egzakt algoritmusokat tárgyaltak (lásd [49, 55]).

Mielőtt rátérünk a modell leírására, a korábban bevezetett jelölések mellé új definíciókat is meg kell adnunk. Jelölje D a depók halmazát, és $D_u \subseteq D$ egy u menetrendi járat depóhalmazát: ez azokat a depókat tartalmazza, amelyekből ki lehet kiszolgálni az u járatot. Jelölje $U_d \subseteq U$ azoknak a járatoknak a halmazát, amelyek kiszolgálhatók a d depóból. Hasonlóan, minden $d \in D$ esetén definiálunk két, $dt(d)$ és $at(d)$ – depóindulási és depóérkezési – csúcspontot a hálózatban; ezek szimbolizálják azt, hogy a jármű a d depóból indul, és oda érkezik vissza. A hálózat csúcspontjainak N halmazát ezek után a következő módon definiáljuk

$$N = \{dt(u) \mid u \in U\} \cup \{at(u) \mid u \in U\} \cup \{dt(d) \mid d \in D\} \cup \{at(d) \mid d \in D\}.$$

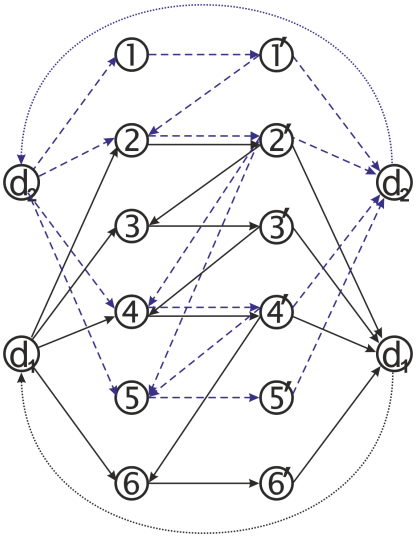
A hálózat éleinek definíciójához vezessük be a következő jelöléseket. Egy d depóhoz tartozó menetrendi járatok halmazához rendeljük a következő irányított éleket:

$$E_d = \{(dt(u), at(u)) | u \in U_d\}, \quad \forall d \in D.$$

Rendeljünk irányított élt a d depó minden u járatának érkezési idejét reprezentáló csúcsból az u-val kompatibilis – szintén d-beli – járatok indulási időpontjához rendelt csúcsba:

$$B_d = \{(at(u), dt(u')) \mid u, u' \in U_d \text{ kompatibilis járatok}\}, \quad \forall d \in D.$$

B_d élei rezsijáratok. További, a d depóhoz tartozó nem menetrendi járatok, amelyekhez éleket kell rendelni a hálózatban, alkotják a depóhoz tartozó kiállási



5. ábra. A kapcsolatalapú modell hálózata egy kétdepós feladat példája esetén. (A földrajzi helyek mellett ezen az ábrán az indulási és érkezési időpontokat sem reprezentáljuk, de csak azokat a csúcsokat kötjük össze éllel, ahol a megfelelő járatok kompatibilisek.)

és beállási élek halmazát:

$$R_d = \{(dt(d), dt(u)), (at(u), at(d)) \mid u \in U_d\}, \quad \forall d \in D.$$

Ha korlátot szeretnénk előírni az egyes depókban rendelkezésre álló járművek számára, akkor szükségünk van az ún. „*depókörfolyam*" élek halmazának definiálására is:

$$K_d = \{(at(d), dt(d))\}, \quad \forall d \in D.$$

Ezek alapján meg tudjuk adni a hálózat d depóhoz tartozó éleinek a halmazát:

$$A_d = E_d \cup B_d \cup R_d \cup K_d, \quad \forall d \in D,$$

és a gráf összes éleinek halmaza

$$E = \cup_{d \in D} A_d.$$

A kapcsolatalapú modell hálózatának felépítését az 5. ábra szemlélteti.

Ezen előkészületek után most már készen állunk arra, hogy definiáljuk az MDVSP-feladatot a $G = (N, E)$ hálózaton. Ehhez definiálunk egy egészértékű x vektort, amely egy többtermékes folyamként is tekinthető. A vektor dimenziója

megegyezik a hálózat éleinek számával. A vektor $e \in E$ élhez tartozó komponensét x_e^d-vel jelöljük, ha az e él a d depóhoz tartozik ($e \in A_d$). Az x_e^d komponens értéke egy adott ütemezésben 1 lesz, ha az adott él benne van az ütemezésben, különben 0. Ez alól csak a depókörfolyam élek a kivételek, mert azok többször is szerepelhetnek egy ütemezésben.

Az x vektor segítségével olyan korlátozó feltételeket definiálunk, amelyek a feladat követelményeit biztosítják.

Ütemezésünkben biztosítani kell azt, hogy minden menetrendi járatot pontosan egyszer hajtsunk végre. Ez azt jelenti, hogy egy lehetséges megoldásban csak olyan járműütemezések szerepelhetnek, amelyeknek – a depókörfolyam élektől eltekintve – nincs közös élük. Ezt a következő feltételekkel érhetjük el:

$$\sum_{d \in D_u, e=(dt(u),at(u)) \in E_d} x_e^d = 1, \quad \forall u \in U.$$

Biztosítanunk kell azt is, hogy az ütemezési időszak végére minden jármű visszaálljon egy depóba. Ez más megfogalmazásban azt jelenti, hogy ha egy adott depóhoz tartozó jármű egy – depótól eltérő valamelyik – csúcsra (állomásra) megérkezik, azt el is kell hagynia.

$$\sum_{e \in n^+} x_e^d - \sum_{e \in n^-} x_e^d = 0, \quad \forall u \in U, \forall n \in N,$$

ahol n^+ jelöli az $n \in N$ csúcsból induló élek halmazát, és hasonlóan, n^- az $n \in N$ csúcsba futó élek halmaza.

Ha vannak depókapacitást korlátozó feltételek, akkor azokat a depókörfolyam élekhez kell kapacitásként, vagyis a folyamértékre vonatkozó felső korlátként előírni. Ez azt jelenti, hogy ha k_d a d depóhoz tartozó azonos típusú buszok száma, akkor a feltételrendszerhez hozzá kell adni az

$$x_{at(d),dt(d)}^d \leq k_d$$

feltételt.

Bármely, a fenti feltételeket kielégítő folyam a feladat egy lehetséges megoldása lesz.

Amennyiben optimális megoldást szeretnénk kapni, definiálnunk kell egy célfüggvényt, amelynek a fenti feltételeket kielégítő optimális megoldását keressük. Ez az élekhez nemnegatív, valós értékű súlyokat rendelve történhet. Az él súlya az adott járat költségét reprezentálja. Amennyiben egyszerűen a járművek számát szeretnénk minimalizálni (ekkor a flottaminimalizálási feladatnak nevezett problémáról van szó), akkor a kiállási élekhez a többi él költségéhez viszonyítva nagyon nagy súlyokat kell rendelni. Ha c_e jelöli az e élhez rendelt költséget, akkor a feladat célfüggvénye:

$$\min \sum_{e \in E} c_e x_e.$$

Ez a fenti korlátozó feltételekkel együtt tekintve egy egészértékű (IP) programozási feladatot határoz meg, amelynek megoldása – bizonyos értelemben – rutinfeladat. Ennek megoldásával kapcsolatban a legfőbb felmerülő probléma az, hogy a hálózatnak túl sok éle lehet. Gondoljunk csak egy több ezer menetrendi járatot tartalmazó hálózatra. Ez már egy közepes, néhány százezer lakosú városnál is előfordulhat, és egy ilyen esetben a lehetséges rezsiátmenetek száma milliós nagyságrendű is lehet. Ennek az oka az, hogy a kapcsolatalapú hálózat minden élt tartalmaz, ahol akár csak elméletileg is lehetséges (rezsi)átmenet (az adott két járat elméletileg kompatibilis egymással). A végső megoldásba ezeknek ugyan csak egy csekély hányada kerül be, de nem lehetséges ezek elhagyása, mert azzal esetleg elveszíthetjük a feladat optimális megoldását is.

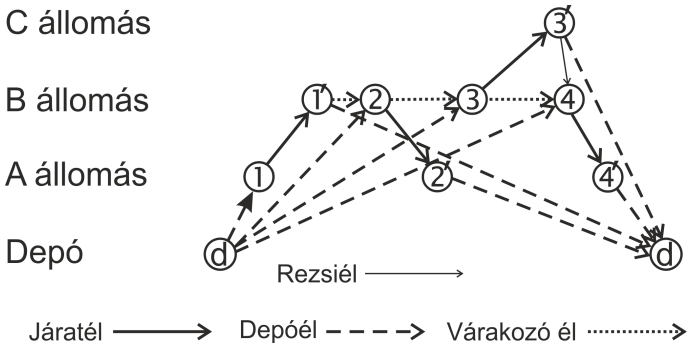
2.3.2. Az idő-tér hálózati modell

Az élek számának csökkentése egy új, módosított modellel történhet. Ezt tárgyaljuk a következőkben. Az *idő-tér hálózati (time-space network, TSN) modellt* Kliewer és szerzőtársai vezették be [48] közúti közösségi közlekedési feladatok kapcsán. Annak ellenére, hogy az idő-tér modellt légiközlekedési feladatoknál (repülőjáratok ütemezésénél) már használták korábban (lásd pl. [44]), a járműütemezés területén [48] az első idő-tér módszert tárgyaló publikáció.

A modell fő vívmánya, hogy nagyobb méretű, a gyakorlatban előforduló problémákat is meg lehet vele oldani.

A modell két dimenziót használ, ezek az idő és a tér. A tér szó jelentése itt az, hogy melyik földrajzi helyről (állomásról) van szó, míg az időt idővonalak reprezentálják, amelyek egyes állomásokhoz (földrajzi helyekhez) tartoznak. Az idővonalak az indulási és érkezési időpontokat tartalmazzák. Minden egyes állomáshoz tartozik egy idővonal, és az állomás minden lehetséges indulási és érkezési időpontjához egy csúcspontot definiálunk annak idővonalán. (Ha több, különböző járat indulási, vagy érkezési időpontja egybeesik, azokhoz egy „összevont" csúcspont tartozik.) Könnyű észrevenni, hogy a két modell közötti alapvető különbség az, hogy az időpontok itt helyekhez vannak kötve. Így a hálózat N csúcshalmazát az állomásokhoz tartozó indulási és érkezési időpontok adják. Minden $d \in D$ depó esetén hasonlóan definiálhatjuk E_d-t, mint a kapcsolatalapú modellben. Természetesen a depókhoz is idővonalakat adunk meg, így hasonlóan definiálható R_d is.

A legfőbb különbség azonban a két modell között a rezsijáratok definíciójában rejlik. Az idő-tér hálózati modellben ugyanis az idővonalak használatával lehetőség nyílik arra, hogy „összegyűjtsük", összevonjuk több rezsijárat lehetséges folyamát. Így nem feltétlenül szükséges minden egyes lehetséges rezsijáratot külön éllel reprezentálni, hanem megfelelő rezsiátmenetekhez tartozó éleket összevonhatunk egyetlen éllé. A szerzők [48]-ban az úgynevezett utolsó-első egyezési stratégiát alkalmazták. Ennek alapgondolata egy kétfázisú összevonási stratégia:



6. ábra. Egy egyszerű példa idő-tér hálózatra [10].

– Az első fázisban minden egyes járat érkezését jelképező csúcspontból az összes többi állomás esetén csak a másik állomás első olyan járatához vezető, rezsijáratot jelentő élt húzzuk be, amely járatunkkal időben az első kompatibilis járat a másik állomáson. Ezen éleket nevezzük első egyezésnek. Csak ezeket az éleket tekintjük a lehetséges rezsiélek közül.
– Az első fázis után egy adott állomásnak az indulást jelképező csúcspontjaiba mindegyik másik állomásról érkező rezsijáratok közül már csak ez első egyezésnek megfelelő rezsijáratokat hagytuk meg. A második fázisban tovább csökkentjük ezen élek számát. Most kerül sor arra, hogy összevonjuk a beérkező rezsiéleket. A második fázisban egy adott állomás időpontjaihoz végignézzük az összes többi állomásról érkező, első egyezés éleket, és az oda egy azonos, de másik állomásról érkező, első egyezést jelentő élek közül csak a legkésőbb indulót hagyjuk meg. Ezt nevezzük utolsó-első egyezést jelképező élnek. Elhagyva a többi, első egyezést jelképező élet, az élek számának további csökkentése lehetséges.

Ezért a B_d halmaz az összes rezsijáratnak csak egy részét fogja tartalmazni. Ezen a módon csökkenthető a különböző állomások között rezsiélek száma. Azonban, hogy teljessé tegyük a modellt, ahhoz mindegyik állomáshoz be kell vezetni az állomáson belüli csúcspontokat összekötő, úgynevezett várakozó élek W_d halmazát, minden $d \in D$ depó esetén. Ezek az élek mindig az állomás idővonalát követik, éllel összekötve az egymást követő (indulási) időpontokat, összegyűjtve azok folyamát. Ebben az esetben az A_d élhalmaz definíciója a következő lesz:

$$A_d = E_d \cup B_d \cup R_d \cup K_d \cup W_d, \quad \forall d \in D,$$

és a gráf összes éleinek halmaza

$$E = \cup_{d \in D} A_d.$$

A 6. ábra mutat egy egyszerű példát egy idő-tér hálózatra.

Ebben az esetben az IP-modell a kapcsolatlapú modelléhez hasonló lesz. Az egyetlen különbség, hogy több él folyamát egyetlen élbe gyűjthetjük össze, így az $x_e^d \in \{0,1\}$ feltétel helyett az „$x_e^d \geq 0$, x_e^d egész" feltételt kell szerepeltetnünk.

2.3.3. A halmazpartícionálási modell

Ebben az alfejezetben Ribeiro és Soumis [63] cikke alapján megadjuk a probléma egy halmazpartícionálási megfogalmazását. Az MDVSP a 2.3.1. alfejezetben megfogalmazott G gráfon megadott körfolyamok segítségével újrafogalmazható. Minden $d \in D$ esetén legyen H_d azon G-beli p utak halmaza, amelyek $dt(d)$-ből indulnak, és oda is térnek vissza. Legyen

$$H = \bigcup_{d \in D} H_d$$

az összes G-beli ilyen utak halmaza. Minden $p \in H_d$ esetén vezessünk be egy y_p^d bináris változót, amelyet a következőképpen definiálunk:

$$y_p^d = \begin{cases} 1, & \text{ha a } p \in H_d \text{ út szerepel a megoldásban,} \\ 0, & \text{különben.} \end{cases}$$

Definiáljuk továbbá $a_{e,p}^d$-t az alábbi módon:

$$a_{e,p}^d = \begin{cases} 1, & \text{ha a } p \in H_d \text{ út tartalmazza az } e \text{ élt,} \\ 0, & \text{különben.} \end{cases}$$

Legyen c_p a $p \in H_d$ úthoz rendelt költség. Ekkor a modell a következőképpen fogalmazható meg. Minimalizáljuk a

$$\sum_{d \in D} \sum_{p \in H_d} c_p y_p^d$$

célfüggvényt a

$$\sum_{d \in D} \sum_{p \in H_d, e=(dt(u),at(u)) \in E_d} a_{e,p}^d y_p^d = 1, \quad \forall u \in U$$

és

$$y_p^d \in \{0,1\}, \quad \forall d \in D, \forall p \in H_d$$

feltételek mellett.

Ha a d depóban korlátos, k_d számú jármű van, akkor ebben az esetben a feltételeket ki kell egészíteni a következő egyenlőtlenségekkel:

$$\sum_{p \in H_d} y_p^d \leq k_d, \quad \forall d \in D.$$

2.3.4. Heurisztikus algoritmusok

Az MDVSP-re számos heurisztikus megközelítést publikáltak. Kliewer és társai egy ún. *változófixálási (variable fixing) heurisztikát* adnak meg [47], amely az idő-tér modellen alapul. Ennek alapötlete, hogy egyszerűsített problémák megoldásával (az említett cikkben külön SDVSP-problémákat oldanak meg az eredeti MDVSP minden depójára) olyan járatsorozatokat találjon, melyek minden egyszerűsített problémában egymás után következnek. Ezeket a sorozatokat leköti, és a felépítendő idő-tér modellben együtt kezeli őket.

Suhl és szerzőtársai [69] egy kerekítéses heurisztikát alkalmaztak. Módszerük alapötlete, hogy az MDVSP-feladat LP-relaxációjának értéke és az optimális egész megoldás értéke közötti különbség meglehetősen kicsi, néha nulla, és az LP-relaxáció optimális megoldásában sok változó kap már egész értéket. Az algoritmus egy előre meghatározott korlát elérésekor leállítja az IP-megoldót (ez lehet a bejárt csúcsok számának korlátja, vagy az aktuális feladat értéke és az LP-relaxáció értéke közötti különbség), majd az így „kézben lévő" részmegoldáson végrehajt egy kerekítési algoritmust. Az algoritmus által meghatározott két kerekítési tartomány $[0, r_l]$ és $[r_u, 1]$, ahol $0 \leq r_l \leq r_u \leq 1$. Legyen x_j egy változó, és $x_j - \lfloor x_j \rfloor = f_j$, ahol $0 \leq x_j \leq 1$. A heurisztika az x_j értékét az alábbi szabályok szerint kerekíti:

$$x_j := \begin{cases} \lfloor x_j \rfloor, & \text{ha} \quad f_j \in [0, r_l], \\ \lceil x_j \rceil, & \text{ha} \quad f_j \in [r_u, 1]. \end{cases}$$

Egymás után több kerekítési iterációt is végrehajthatunk, feltéve, hogy az így létrejött LP még mindig lehetséges megoldást ad. Ha nem tudtunk egy változót sem kerekíteni, akkor a kezdeti kerekítési intervallumok növelhetők. Az így kapott feladatra újra lefuttatjuk a korlátozás-szétválasztás módszerére alapuló IP-megoldót. A folyamatot addig ismételjük, amíg minden változót fixáltunk, vagy a feladatnak nincs lehetséges megoldása.

Pepin és szerzőtársainak dolgozata több heurisztikus módszert hasonlít össze [61]. A publikációban öt különböző heurisztikus megközelítést vizsgálnak:

- egy szétválasztás és vágás (*branch-and-cut*) típusú megoldást,
- Lagrange-relaxációra alapuló heurisztikát,
- oszlopgenerálást,
- egy nagy szomszédsági keresést,
- valamint egy tabu-keresést.

A szétválasztás és vágás típusú megoldás alkalmazásánál a feladat idő-tér modelljét építik fel, majd oldják meg CPLEX-szel, míg az oszlopgenerálásnál lényegében szintén CPLEX-szel kapnak eredményt.

A probléma folyammegmaradási feltételének Lagrange-relaxálásával a kapott feladat SDVSP-részproblémákkal lesz ekvivalens, amit egy aukciós algoritmussal oldanak meg. Az így kapott alsó korlátot szubgradiens módszerrel javítják.

A nagy szomszédsági keresés egy kezdeti megoldásból kiindulva minden iterációban r különböző járműműszakot választ ki, több választási stratégia valamelyikével. Az így kiválasztott műszakokat újraoptimalizálják, az oszlopgenerálást használva. A választási stratégiák valószínűsége minden iterációban változik attól függően, hogy az előzőek mennyire voltak hatékonyak.

A tabukeresés szintén egy kezdeti megoldásból indul, és minden iterációban az aktuális megoldás egy szomszédjára tér át. Kétféle módon definiálják a szomszédságot: 1-mozgatás, és csere-mozgatás alapján. Az 1-mozgatással olyan szomszédok érhetőek el, melynél a v_i eszköz valamely járatát a szomszédban egy másik, v_j eszköz hajtja végre. A csere-mozgatás olyan szomszédokat ad, melyeknél, ha a t_k járatot a v_i eszköz, valamint a $t_{k'}$ járatot a v_j eszköz hajtotta végre az eredeti ütemezésben, úgy a szomszédban a t_k járatot a v_j, valamint a $t_{k'}$ járatot a v_i eszköz hajtja végre (valamely k, k', i és j értékekre).

3. A járművezető-ütemezési feladat

A munkaerő költsége egy nagyon fontos, kiemelt összetevője az egész operatív ütemezés költségének, így a dolgozók munkájának beosztása, ütemezése manapság központi kérdés. Tipikus személyzetbeosztási alkalmazások jelennek meg a kórházak üzemeltetésénél (pl. nővérek), a call-centerekben (operátorok), légitársaságoknál (stewardessek, pilóták), és a közlekedési vállalatoknál (járművezetők munkájának ütemezésére). A beosztások elkészítésekor a feltételek erősen szakmaspecifikusak, így a problémák megfogalmazása, a modellek és megoldási módszerek is meglehetősen változatosak [28].

A járművezető-ütemezés során (amelyet röviden vezetőütemezésnek is fogunk nevezni) adott az ellátandó, operatív feladatok halmaza. Ezt kell olyan módon műszakokba rendezni az adott időperiódusra (általában egy napra) vonatkozóan, hogy minden feladat hozzá legyen rendelve egy műszakhoz, és a műszakok minden – a járművezetők számára előírt – szabálynak megfeleljenek. A hozzárendelést úgy kell megadni, hogy eközben minimalizáljuk az ütemezés során kialakított műszakok költségeinek összegét.

Sok olyan feltétel van, amelyet a vonatkozó jogszabályok és az adott közlekedési vállalat dolgozóira vonatkozó szabályok, előírások határoznak meg. Ilyenek a napi munkaidő-beosztásra vonatkozó szabályok, a maximális munkaidő és a napi pihenőidő minimális hossza, a kellő számú és idejű szünet előírása egy adott vezetési idő után, két munkabeosztás között kötelezően előírt pihenőidő stb. A menetrendi járatokon és a rezsimeneteken kívül sokféle technikai és adminisztrációs feladat is van, amelyeknek pontosan meghatározott időbeli hossza van. Ilyenek az utasok ki-

és beszállási ideje a járatok végállomásain: a tankolás, a parkolás, a műszakkezdet és -befejezés stb. Megjegyezzük, hogy a vezetési idő és a munkaidő különböző fogalmak, így ezekre eltérő szabályok vonatkozhatnak, miként arra is, hogy melyik technikai tevékenység ideje melyikbe számít bele. Ehhez járulhat még hozzá az, hogy különböző vállalatoknál további korlátozó feltételek és szabályok is előfordulhatnak (szolgálati idő, tengelyen töltött idő stb.).

A különböző tevékenységekhez tartozó dolgozói költségeknek (a fizetéseknek és egyéb járulékoknak) megfelelően az egyes feladatokhoz költségeket tudunk rendelni. Így az egyes műszakok költségeit úgy számítjuk ki, hogy összeadjuk a különböző tevékenységek költségeit.

A legnépszerűbb megközelítés a probléma megoldására a halmazpartícionálás (lásd pl. [30]), valamint az annak relaxációját jelentő halmazlefedési modell (pl. [66, 73]) vizsgálata.

A megoldás előállítása tipikusan kétféle módon történhet: vagy a korlátozó feltételeket kielégítő egy lehetséges megoldás előállításával [27] és annak iteratívan történő javításával, vagy nagyszámú legenerált lehetséges megoldás közül a legjobbak kiválasztásával [50]. Különböző jellegű algoritmusokat alkalmaztak a probléma megoldására. Ezek közül érdemes kiemelni az egészértékű programozási modelleket (lásd pl. [72]), az evolúciós elvekre épülő metaheurisztikus eljárást [50], vagy a Fuzzy-módszeren alapuló [52] algoritmusokat.

A helyes modell kialakítását nagymértékben nehezíti az, hogy a különböző közlekedési társaságok más-más elveket, belső szabályokat alkalmaznak a vezetőütemezésnél. Ilyen lehet például annak előírása, hogy mely pontokon (mely földrajzi helyeken) történhet vezetőcsere egy adott járművön, ez mennyi időt igényel, milyen szabályok vonatkoznak a szünetekre stb. Az ilyen különbségek és a nagyszámú, nehéz korlátozó feltétel miatt a legtöbb megoldási módszer csak a legalapvetőbb korlátozó feltételeket veszik figyelembe. Ilyen lehet például a maximális munkaidő, a néhány rövid és egy hosszú (étkezési) munkaközi szünet.

Egy valós életből származó példa esetén az összes szünet kezelése, ütemezése azonban nem egyszerű: az egy műszakon belül előírt szünetek száma ugyanis függhet a műszak és/vagy a már ledolgozott munkaidő hosszától. A munkaidő előrehaladásával egyre gyakrabban kell a szüneteket kiadni (általában egyre rövidebb munkaszakaszok után, de ezekre is vonatkozhatnak bonyolultabb szabályok); több különböző variáció van a szünetek hosszára. Emellett sok szabály nem a munkaidőre vonatkozik, hanem a vezetési időre, és a kettő eltérhet egymástól. A munkaidőbe beleszámolódhatnak más, előírt technikai idők is, például a végállomásokon történő fel- és leszállási idők [29]. A fenti korlátozó feltételeket azért soroltuk fel, hogy érzékeltessük azt, hogy mennyire bonyolult egy ilyen rendszer, és próbáltuk felvázolni azt is, hogy milyen elvárásokat támasztanak egy matematikai modell elkészítése és alkalmazása során az egyes felhasználók.

A vezetőütemezés központi logisztikai probléma a tömegközlekedésben, hiszen a járművezetőkkel kapcsolatos költségek a teljes, közlekedéssel kapcsolatos költsé-

gek nagy részét alkotják. A vezetőütemezés alapproblémája a következő módon határozható meg: Adottak a járművek közlekedései/tevékenységei, fix indulási és érkezési idejükkel és helyükkel. A feladat az, hogy mindegyikhez járművezetőket rendeljünk minimális költséggel, átfedések nélkül, kielégítve a szabályokat és az előírásokat. Ennek klasszikus matematikai megfogalmazása egy halmazlefedési problémát eredményez, amely problémáról ismert, hogy NP-teljes [37].

A vezetőütemezési problémára *CSP-ként (Crew Scheduling Problem)* fogunk hivatkozni, bár az angol nyelvű szakirodalomban emellett szokás Driver *Scheduling Problem*-nek vagy *Duty Scheduling Problem*-nek is nevezni. Az irodalomban számos CSP-megoldási módszer és alkalmazás található. A következőkben ezeket tekintjük át röviden. Részletesebb áttekintéshez például a [28] tanulmány ajánlható az ez iránt érdeklődőknek.

3.1. Modellek és algoritmusok a CSP-re

Az alfejezetet az alapvető jelölések összefoglalásával kezdjük. Egy olyan földrajzi helyet, ahol egy *feladat* (*task*) kezdődik, vagy befejeződik, és a vezető elhagyhatja, vagy elfoglalhatja a járművet, *váltási helynek* (*relief point*) nevezünk. Amikor egy jármű egy váltási helyhez ér, *lehetséges váltási pontról* (*relief oppurtunity*) beszélünk. A jármű két egymást követő lehetséges váltási pont közötti feladatait *munkaszakasznak* (*work piece*) nevezzük. A vezetőütemezési feladat megoldására két általános megközelítés létezik.

3.1.1. A generálás és kiválasztás módszere

A módszerre az angol elnevezés (Generate and Select) rövidítése alapján GaS-sel fogunk hivatkozni. GaS-technikát használtak az [50, 52, 72] cikkekben. A módszer a következőképpen foglalható össze: A generálási lépésben állítsunk elő nagyszámú szabályos műszakot, majd a kiválasztási lépésben keressünk egy olyan részhalmazt a generált műszakokból, amely minimális költségű és lefedi a feladatokat.

Az első fázis jelentős számítási időt igényel. A számítási igény nagyban függ a járatok mennyiségétől, a szabályok számától és bonyolultságától. Emellett a szabályok ellenőrzésének számítási igénye is nagyban befolyásolja ennek a fázisnak – és így az egész probléma – komplexitását.

A GaS kiválasztási fázisa hagyományosan egy halmazlefedési vagy halmazpartícionálási feladatként fogalmazható meg. Jelölje n' a lehetséges műszakok és m' a feladatok számát, $x_j \in \{0,1\}$ és $a_{i,j} \in \{0,1\}$ az i. feladathoz és a j. műszakhoz tartozó változókat ($i = 1,2,\ldots,m'$, $j = 1,2,\ldots,n'$), ahol

$$x_j = \begin{cases} 1, & \text{ha a } j. \text{ műszakot kiválasztjuk,} \\ 0, & \text{különben.} \end{cases}$$

$$a_{i,j} = \begin{cases} 1, & \text{ha az } i. \text{ feladatot tartalmazza a } j. \text{ műszak,} \\ 0, & \text{különben.} \end{cases}$$

Ekkor a feladat a következő módon írható fel:

$$\min \left(w_1 \sum_{j=1}^{n'} c_j x_j + w_2 \sum_{j=1}^{n'} x_j \right) \tag{1}$$

figyelembe véve az alábbi korlátozó feltételeket:

$$\sum_{j=1}^{n'} a_{i,j} x_j \geq 1, \quad i = 1, 2, \ldots, m', \tag{2}$$

ahol c_j a j. műszak költsége, w_1 és w_2 különböző súlyok.

Az (1) célfüggvény a (2) feltételrendszerrel egy halmazlefedési feladatot generál, melynél – mint az észrevehető – átfedés is lehetséges. Ez azt jelenti, hogy ugyanazon feladathoz elvileg több műszak is hozzárendelhető. Ebben az esetben a gyakorlatban gondoskodnunk kell az esetleges átfedések kezeléséről, vagy ezek számának minimalizálásáról is [25, 66].

A partícionálási modell a lefedési feladat olyan megszorítása, amikor átfedés nem lehetséges. Ez annyiban módosítja a feltételrendszert, hogy az egyenlőtlenségek helyett egyenlőségeket írunk elő. A probléma az, hogy ebben az esetben a lehetséges megoldások létezése nem garantált. Megjegyezzük, hogy mindkét feladatról bebizonyították, hogy NP-nehéz [37].

A probléma megoldására számos eljárás létezik. Például egészértékű programozási módszereket használnak [72]-ben, heurisztikus megoldási technikákat alkalmaznak [50]-ben.

3.1.2. A konstruktív megközelítés módszere

A konstruktív megközelítés egyetlen megoldást épít fel irányítottan egy optimalizálási eljárás segítségével. Ennek kerete rendszerint egy hagyományos iteratív eljárás, amely egy induló megoldást generál, és iteratívan javítja azt. Ezzel a módszerrel találkozhatunk a [2, 27, 40, 71] cikkekben.

3.1.3. A szabályok ellenőrzése

A legkritikusabb részfeladat annak ellenőrzése, hogy a kapott megoldás lehetséges megoldás-e. Ez azt jelenti, hogy – mindkét megközelítés generáló folyamata során – a megoldáshoz tartozó összes műszakról el kell döntenünk, hogy azok

teljesítik-e a feltételeket. Miután – még egy kisméretű probléma esetén is – a lehetséges műszakok száma nagyon nagy, ezért a technikák többségénél különböző egyszerűsítések kerülnek végrehajtásra azért, hogy a futási időt csökkentsék. Egy alkalmazott megoldás az, hogy a probléma méretét csökkentik azzal, hogy lehetséges váltási pontokat hagynak el, vagy azzal, hogy csak számításilag kezelhető számú műszakot generálnak le. Ezek az egyszerűsítések természetesen befolyásolhatják és korlátozhatják a módszernél az optimalizálás sikerét.

3.2. A CSP korlátozó feltételeiről

A CSP megoldása során a korlátozó feltételek valóban döntő szerepet játszanak a megoldási módszer szempontjából. Ezek erősen befolyásolják a megoldás milyenségét és a számítási időt. Erős korlátozó feltételekkel rendelkező valós problémák esetén nehéz kérdés, hogy megtalálják az egyensúlyt a fenti szempontok között. Mivel általában a probléma mérete nagy, és az előírt szabályok rögzítettek, a fő cél az, hogy találjunk egy hatékony módszert, amely gyakorlati szempontból jó minőségű, kivitelezhető megoldást nyújt.

A fő korlátokat nemzetközi (pl. EU-s szabályok) és nemzeti (minisztériumi, önkormányzati stb.) szinten határozzák meg. A helyi közlekedési vállalat is további „kemény” és „puha” igényeket (szigorúan betartandó vagy ajánlott) definiálhat. Fontos igény a valós, gyakorlati alkalmazások mellett a rugalmasság. Sok esetben az adaptálhatóság és a futási idő a legfontosabb szempontok, ha a megoldás minősége megfelel bizonyos követelményeknek (általában valamilyen korlátozó feltételeknek). Abban az esetben, ha döntéstámogató az ütemezés, egyes mérnöki követelmények minőségére vonatkozóan a megoldást nem kell, vagy nem is lehet formalizálni. Ekkor a hosszútávú tervezés ütemezése interaktív mechanizmusként valósul meg, azaz interaktív, mérnöki beavatkozás lehetséges vagy szükséges is. Egy módszer alkalmazásának lehetősége nagymértékben függ attól, hogy miként lehet a modellben a szükséges szabályokat megfogalmazni, formalizálni, milyen paramétereket képes kezelni, és ezáltal a különböző megoldásokat előállítani.

Ez a rugalmasság központi kérdés az optimalizálási folyamat többi szakaszához való viszony tekintetében is: a megoldás értékelése önmagában nem, csak a járműütemezés, valamint a vezetőbeosztás eredményének értékelésével együtt lehetséges.

3.3. A célfüggvényről

A CSP megoldásának minősége csak részben függ az eredményezett műszakok számától. Általánosabban, egy megoldás költségének a tartalmazott műszakok költségeinek összegét tekinthetjük. Természetesen egy járművezető foglalkoztatásának van egy általános költsége, ezért minimalizálva az összköltséget, a vezetőütemezés alacsony szinten fogja tartani a műszakok számát is. Másrészt, költségeket rendelhetünk a műszakokhoz a bennük szereplő feladatok szerint is. Ezen kívül meg lehet határozni egyéb jellemzőket, mint például a műszak típusa, a vár-

ható időtartama, amelyek járulékosan, büntetésként szerepelhetnek a költségben is. Mindazonáltal, a legáltalánosabb értelemben értékelve a megoldást, a műszakok összes munkaidejét lehet teljes költségként tekinteni. (Egyes helyeken a bérezés nem munkaidő szerint történik, ilyenkor természetesen a műszak költségét is ehhez kell igazítani.)

4. Integrált stratégiák

Ebben a fejezetben a jármű- és járművezető-ütemezési feladat *integrált megközelítéseit* (*vehicle and crew scheduling problem, VCSP*) tekintjük át röviden, megemlítve az irodalomban tárgyalt legfontosabb modelleket, algoritmusokat. Az érdeklődőknek kiindulópontként a [68] cikket ajánljuk, a részletesebb elmélyüléshez javasoljuk a kitűnő áttekintést adó [24] tanulmányt.

A járművezető-ütemezés a hagyományos megközelítést követve a járműütemezés fázisa után hajtódik végre. (Ezért ezt *szekvenciális megközelítésnek* is nevezzük.) Ez általában hatékonyan végrehajtható, ha a váltási helyek között sok olyan van, amely megengedett váltási pont is egyben, és a kialakított járműütemezés gyakran érint ilyen váltási helyet. Azonban, ha a kialakított járműütemezés túl „sűrű", sok helyen nincs elég idő a járművezető-váltásra, vagy a kialakított járműütemezés hosszú ideig nem érint megengedett váltási helyet, akkor ronthatunk az előző fázis eredményén, veszíthetünk a hatékonyságból. A járműütemezés és a járművezető-ütemezés egyszerre történő elvégzése emiatt indokolt lehet, és ez napjaink egyik fontos kutatási területe.

A VCSP feladata a következőképpen fogalmazható meg. Adott menetrendi járatok egy halmaza, adott a járműflotta, melynek járművei különböző depókhoz tartoznak, adottak továbbá a járművezetőkre vonatkozó szabályok. A feladat a járműveknek és a járművezetőknek olyan érvényes ütemezését adni, hogy az valamennyi – járműre és járművezetőre – vonatkozó szabálynak megfeleljen, és minimális költségű legyen.

Ez a feladat egy, az MDVSP-hez hasonló egészértékű programozási feladatként írható fel, kibővítve olyan feltételekkel, amelyek egyrészt a vezetőütemezésnek felelnek meg, másrészt pedig kapcsolatot teremtenek a két ütemezés között. Ekkor a célfüggvényben is a két ütemezés költségének összege jelenik meg (lásd pl. [24]).

Mint korábban láttuk, az MDVSP- és a VCSP-feladat kapcsán is elmondhatjuk, hogy mindkettő NP-nehéz feladat.

A következőekben ismertetett módszerek esetén néhánynál a járműütemezés részfeladata egydepós, és így polinomiális időben megoldható [13].

A VCSP feladatában a lehetséges műszakok száma igen nagy, különösen a többdepós esetben. Éppen ezért az 1990-es évek végéig nem övezte akkora érdeklődés a problémát a kutatók körében, mint a járműütemezési vagy járművezető-ütemezési

problémát. A számítógépek számítási sebességének növekedése és az egyre kifinomultabb optimalizálási módszerek alkalmazása révén azonban az utolsó évtizedben megnőtt az érdeklődés ezek irányában, szaporodnak az ilyen tárgyú publikációk. A kis és közepes méretű problémák mára megoldhatóvá váltak.

Már az 1980-as évek elején Bodin és szerzőtársai komolyan kritizálták a szekvenciális megközelítést [13]. Észak-amerikai tömegközlekedési példákon keresztül bizonyították azon érvelésüket, hogy a járművezetők (bér)költsége sokszor magasabb, mint a járművekkel kapcsolatos költségek. Extrém esetekben a bérköltség akár 80%-át is adhatja a kétféle költség összegének. Ebből az következik, hogy ezt a költséget nem lehet másodlagos kérdésnek tekinteni. Sőt, alapvetően figyelembe kell venni már az első fázisban, vagy kombinált, integrált módon. Ball és szerzőtársai [8] ugyancsak integrált modellt javasoltak.

Az ezt követő cikkekben különböző heurisztikus módszereket adtak meg a kutatók, amelyekben a VCSP heurisztikus megoldása során figyelembe vettek bizonyos, járművekre vonatkozó feltételeket is. Az 1997 előtt használt módszerekről egy jó áttekintés található Gaffi és Nonato 1999-es közleményében [35], vagy Freling 1997-es PhD-dolgozatában [32].

Az első, valóban integrált jármű- és vezetőütemezési megközelítés csak 1995-ben született Freling és szerzőtársai révén [31]. Erről Gaffi és Nonato [35] bizonyította be, hogy hatékony lehet akkor is, amikor a megengedett váltási pontok távol esnek egymástól, például helyközi vagy távolsági tömegközlekedés esetén. A módszer szintén jól működhet azokban az esetekben, amikor egy vezető egy járművet használhat csak a műszakja során.

A legnépszerűbb megközelítés az integrált VCSP-problémára kétségkívül az egészértékű programozási modell alkalmazása. Freling és szerzőtársai [31] modellje az egydepós esetre három részből állt: az SDVSP-t kvázihozzárendelési, a VCSP-t halmazpartícionálási feladatként fogalmazták meg. A kettőt korlátozó feltételekkel kötötték össze, biztosítva a kompatibilitást. A feladat megoldására Lagrange-relaxációt és oszlopgenerálási módszert kombináló, közelítő megoldást adtak. Ez aztán további publikációkat inspirált, lásd [33, 34, 46].

Freling és szerzőtársai a [33, 34] cikkeikben szintén az egydepós integrált feladatot tekintették, a buszok számára vonatkozó felső korlát nélkül. Ugyancsak oszlopgenerálást – és ezen belül Lagrange-relaxációt – használtak az ott adódó ún. mesterprobléma megoldására, a feltételek relaxációjával. A [34] cikkben valós, kb. 150–250 járatból álló feladatok megoldásáról számoltak be, kezelhető megoldási időben. Megmutatták, hogy csak kis javítás érhető el a feladatok szekvenciális, egymás után történő megoldásához képest. Ez a javítás szignifikánsabb, ha a járművezetők nem válthatnak járművet (egy szünetet követően).

Az első pontos megoldást adó algoritmust 1999-ben publikálta Haase és Friberg [41] az egydepós esetre. Modelljükben mindkét halmazpartícionálási megközelítés integrált matematikai megfogalmazását adták: mindkét részfeladatot halmazpartícionálási feladatként fogalmazták meg. Eljárásukban a járművek ütemezése Ribeiro és Soumis 1994-es [63], míg a járművezetők ütemezése Desro-

chers és Soumis 1989-es [26] modelljein alapult. A szétválasztás, vágás és árazás (branch-and-cut-and-price) típusú algoritmusukban oszlopgenerálást és vágásgenerálást alkalmaztak. Az oszlopgenerálási mesterprobléma az LP-relaxációnak felelt meg, míg az árazásra (pricing) a legrövidebb út problémák megoldása szolgált. Algoritmusukkal csak kisméretű példákat tudtak megoldani, legfeljebb 20 járatból állókat.

Egy másik érdekes egzakt algoritmust adtak meg az egydepós esetre Haase és szerzőtársai 2001-ben [42]. A járművezető-ütemezési problémát egy többtermékes hálózati folyam problémaként megfogalmazva oldották meg úgy, hogy beleágyazták az egydepós, járművekre vonatkozó feltételeket. Ezt olyan módon tették, hogy mindkét fázist figyelembe véve, a VCSP-feladatra garantált, pontos optimumot adott az algoritmusuk. Itt a szétválasztás és árazás (branch-and-price) típusú algoritmusra számos gyorsítási technikát alkalmaztak. Két algoritmusváltozatot is tárgyaltak a szétválasztáskor adódó ágak kezelésére: egy egzakt és egy heurisztikus változatot. Véletlen feladatokon végeztek számítógépes szimulációs teszteket. Átlagosan kb. 1,5 órás (bár 10-ből 6 esetben 3 órás) futási idővel tudtak 150 járatméretű példákig pontos megoldást szolgáltatni. Ennél nagyobb, 350 járatméretű példákig heurisztikus megoldást alkalmaztak, 2 órán belüli CPU-időt használva. Itt a feladat olyan egyszerűsített megfogalmazását tekintették, ahol a költségben csak a szükséges buszok száma szerepelt, és ők sem vettek figyelembe korlátot ezek számára. Így megfogalmazásuk gyakorlatilag szintén egydepós feladatként fogható fel. Fontos kiemelni ugyanakkor, hogy a feladatok nem valós, hanem a tömegközlekedési feladatokat szimuláló véletlen adatokból származtak.

A többdepós esetben Gaffi és szerzőtársa [35] tárgyalták először az integrált feladatot, heurisztikus módszert használva. Lagrange-relaxációt és az oszlopgenerálás módszerét használták, csak a többdepós esetre módosítva a megfogalmazást. Ők is az egyidejű vezető- és járműütemezés előnyei mellett érveltek olasz tömegközlekedési példákon keresztül, amelyek nem városi (nem helyi) közlekedési példák voltak. De újra felhívja a figyelmünket a számítási idő fontosságára, kritikus voltára, hogy egy 257 járatból álló, olasz tömegközlekedési példa esete 24 óránál hosszabb számítási időt adott, átlagosan 2-6 óra volt a futási idő, bár a maiaknál lényegesen lassabb, 180 MHz-es PC gépen.

Huisman és szerzőtársai 2005-ben [46] a korábbi egydepós [34, 42] modelleket és algoritmusokat sikeresen terjesztették ki a többdepós problémára. Ez volt a többdepós probléma első általános matematikai megfogalmazása. Két megközelítést is javasoltak: az egyik a [32, 34], a másik a [42] tanulmányokban ismertetett módszer általánosítása a többdepós esetre. Mindkettő esetén az első fázisban egy alsó korlátot számítanak ki az optimumra. Az első fázis LP-relaxácón alapszik, oszlopgenerálást és Lagrange-relaxációt használ. A második fázis ad egy lehetséges egészértékű megoldást a feladatra. A [34]-ben használt Lagrange-relaxációt alkalmazva először egy járműütemezést generál, amelyből aztán a [33, 34] cikkekben tárgyalt módszereket használva járművezető-ütemezést készít. Összehasonlító

teszteket kb. 650 járatig készítettek, amelyek eredményei felülmúlják a szokásos szekvenciális ütemezést használó módszereket. Teszteket végeztek valós és szimulációs tesztadatokon egyaránt. A két megközelítés között nem mutatkozott szignifikáns eltérés, de az első valamennyivel jobbnak bizonyult.

Valós, nagyméretű, többdepós, heterogén járműflotta esetén az eddig megemlített módszerek integrálása nehéz lenne egy alkalmazási rendszerbe. Így ezek helyett a szekvenciális, esetleg kézi módon történő integrálást alkalmazták a 2000-es évek elejéig.

A [22] cikkben rövidebb megoldási időt tudtak produkálni és nagyobb feladatokat tudtak megoldani a szerzők, kisebb méretű részfeladatokra vágva a feladatokat, és azokat – önmagukban, integrált módon – külön oldva meg. A többdepós feladatra Borndörfer és szerzőtársai [15] is Lagrange-relaxáción alapuló megközelítést használtak. A járműütemezési és a járművezető-ütemezési problémáknál használt közelítő megoldásokat alkalmazták, ezek információit felhasználva egy branch-and-bound típusú algoritmusban. Egészértékű programozási technikát használtak, különböző korlátozó feltételekkel összekapcsolva a járműveket és járművezetőket a rezsi és a kiállási, valamint beállási járatoknál. Heurisztikus módon, Lagrange-relaxációt használva, majd az egészértékű megoldásokat egy korlátozás és szétválasztás típusú technikával nyerve nagyméretű, 1500-as járatszámú problémát is kezelni tudtak.

Az integrált feladat egy heurisztikus megközelítését adja Laurent és Hao dolgozata [51] is. Habár az általuk tárgyalt feladat általánosabb, mint az egydepós eset, de feltételezik, hogy az összes jármű ugyanabban a depóban parkol. Ugyanakkor a járművek típusaik szerint nem kell, hogy homogén flottát alkossanak. Egy mohó, véletlen adaptív keresést alkalmaznak (Greedy Randomized Adaptive Search, GRASP). Hasonlóan a legtöbb módszerhez egynapos időhorizontot alkalmaznak, de a járművezetőkre vonatkozó feltételek egyszerűsítettek: csak a napi műszak „átmérőjére" (a fellépés és lelépés között eltelt időre) vonatkozó korlátot, a napi teljes munkaidőre vonatkozó korlátot, és azt a paramétert veszik figyelembe, hogy a járművezetőknek megengedett-e vagy nem a napon belüli járműváltás.

Mesquita és Paias 2008-ban [58] két matematikai megfogalmazást adtak a problémára. Mindkét modell többtermékes hálózati modellt tartalmaz a járműütemezésre, míg a járművezető-ütemezési rész vagy halmazparticionálási vagy kombinált halmazparticionálási és lefedési megfogalmazás. A relaxált LP-t oszlopgenerálással oldották meg, amelynek részproblémája korlátozó feltételekkel ellátott legrövidebb út feladat. Amennyiben a relaxált LP megoldása nem egésznek adódott, egészértékű megoldást kerestek korlátozás és szétválasztás alapú hagyományos IP-megoldóval. A [38]-ban alkalmazott eljárás egy ehhez hasonló megközelítést használt, a szerzők korábbi idő-tér hálózati modelljét alkalmazva a járműütemezési komponensben.

A részben integrált modellek közé tartozó modellt tárgyal Gintner, Kliewer és Suhl 2008-as cikke [39]. Ez annyiban hasonlít a tradicionális szekvenciális meg-

közelítésekre, hogy a járművek optimalizált ütemezését végzi el először, majd – ennek eredményét felhasználva – a járművezetőkét. A járművek ütemezési fázisa az idő-tér hálózati MDVSP-modellt használja, de annyiban különbözik az eredeti megközelítéstől, hogy nem egyetlen optimális megoldást ad, hanem többet, minimális járműszámmal és minimális költséggel. Ezek után a VCSP-t halmazpartícionálási feladatként oldja meg, és Lagrange-relaxációt alkalmaz. A klasszikus megközelítéssel összehasonlítva a módszer jobb járművezető-ütemezéseket eredményez.

Steinzen és szerzőtársai 2010-ben [68] egy teljesen integrált VCSP-megközelítést adnak. A mögöttes, járműütemezést kezelő modell az idő-tér hálózati modell. A megoldás itt is az oszlopgenerálás és a Lagrange-relaxációs módszert kombinálva történik. Az oszlopgenerálás részfeladatát korlátozó feltételekkel ellátott – egy idő-tér hálózaton alapuló – legrövidebb út feladat megoldásával modellezték. Egy heurisztikus, szétválasztás és árazás típusú módszerrel generáltak lehetséges megoldásokat. A numerikus tesztjeik azt mutatják, hogy ez a módszer felülmúlja a korábbiakat az ismertetett tesztfeladatokon, amelyeket 640 járatot tartalmazó példaméretekig tekintettek. Az összehasonlítás alapjait a [15, 38, 45, 46] cikkek alkották, amelyek közül valamennyit felülmúlta tesztpéldáikon az algoritmusuk. Az alkalmazott tesztpéldáik (kizárólag) a Steinzen weblapján található példák [67] voltak.

5. A műszakkiosztási feladat

A járművezetők műszakkiosztási problémája is az általános műszakkiosztási feladatok közé tartozik. Itt a feladat az, hogy egy tervezési periódusban munkavállalókat – számos általános és speciális feltétel figyelembevételével – rendeljünk hozzá a napi műszakokhoz. A járművezetők műszakkiosztási feladatának feltételei általában megfelelnek más, tipikus alkalmazások feltételeinek, amelyek egyes személyzetbeosztási [17, 18] és nővérbeosztási [19, 11] feladatoknál, illetve a call-centerek üzemeltetése esetén az operátori műszakkiosztásnál [65] is megtalálhatók.

A járművezetők műszakkiosztási feladatában adottak a járművezetői műszakok, amelyeket a vezetőütemezés során meghatároztunk. Ezek napi beosztásokat jelentenek. Ezek a műszakok a tervezési periódus különböző napjain eltérőek lehetnek. Adottak továbbá a rendelkezésre álló járművezetők.

A műszakkiosztás általában hosszabb periódusra történik, amelyet *tervezési időszaknak* nevezünk. A tervezési időszak tipikusan néhány hétből álló időintervallum, de ez lehet néhány nap, hónap, vagy akár egy egész év is. A műszakkiosztási feladat lényege, hogy egy tervezési időszakra az adott műszakokat rendeljük valós alkalmazottakhoz úgy, hogy az előírt szabályokat betartjuk. Ez a feladat több alkalmazási területen fordul elő. A leggyakoribb alkalmazások a légiközlekedésben

(pilóta és személyzet), vasúti közlekedésben, call-centerekben dolgozók, nővérek és buszsofőrök munkájának tervezése.

A módszernek a tervezési időszak minden műszakjához járművezetőt (konkrét személyt) kell rendelnie. Van néhány általános szabály, rendelkezés, amely korlátozó feltételként veendő figyelembe. Ilyenek például a maximális heti munkaórák száma, vagy a szabadnapok előírt minimális száma egy adott időszakban, ezen belül a vasárnapok száma, stb. A szabályok teljesen különbözhetnek az alkalmazási területtől függően (légi, vasúti közlekedés, városi tömegközlekedés stb.).

Gyakran további speciális helyi szabályok is előfordulnak, amelyeket szintén korlátozó feltételként kell figyelembe venni a beosztás elkészítéskor. Ilyen például az, hogy milyen kategóriájú vezetői jogosítvánnyal kell rendelkeznie a vezetőnek bizonyos típusú buszok vezetéséhez.

A hozzárendelés költsége a közlekedési cégek esetén nagyban függhet a vállalatvezetés „filozófiájától". Ez lehet egyetlen cél is, de több összetevő is alkothatja. Utóbbi esetben ezekből több is (pl. súlyozott módon) megjelenhet a célfüggvényben. Ilyen lehet a járművezetők számának minimalizálása, a szerződésben szereplő megállapított munkaóráktól való eltérések összegének minimalizálása (akár túlóráról, akár az abban szereplőnél kevesebb tényleges munkaidőről van szó, azaz a túlfoglalkoztatást és az alulfoglalkoztatást egyaránt kerülni kell, a lehetséges mértékben). A fenti okok miatt több célfüggvényű optimalizálási módszerek használata is indokolt lehet [59, 60]. Miután az alapfeladat általánosan tekintve hozzárendelési probléma, a szakirodalomban megtalálható néhány – a hozzárendelési feladatra alapozott – általános megközelítés is a műszakkiosztási feladatra. Ilyenek például a többtermékes folyam algoritmusok [17], a logikai programozás alkalmazása korlátozó feltételekkel [74], vagy az evolúciós módszert használó algoritmusok [59].

A feladatot több részfeladatra lehet bontani, ahol az egyes lépéseket egymás után, esetleg iteratívan lehet végrehajtani. Bizonyos környezetben nem minden lépés szükséges, illetve össze is lehet vonni őket. A buszközlekedésben leginkább előforduló lépések a következők.

– Igényfelmérés. Első lépésben meghatározható, hogy mennyi emberre van szükség. Ez függ a műszakok számától, egymáshoz képest időbeni viszonyuktól, méretüktől, típusuktól, továbbá befolyásolják a rendelkezésre álló alkalmazottak „tulajdonságai" (pl. szerződések, jogosítványok).

– Szabadnap kiosztás. A szabadnapok kiosztása főleg akkor szükséges, amikor nem teljesen kötöttek a jövőbeni műszakok. Ennek végrehajtása során némi rugalmasságot biztosítani kell. Ez a lépés természetesen a definiált szabályoktól függ, attól, hogy mennyi és milyen szabadnapot kell kiosztani.

– Műszaksorozatok készítése. Ebben a lépésben az adott műszakokból (esetleg másképp meghatározott feladatokból) bizonyos időintervallumra (jellemzően egy hétre vagy hónapra) adott hosszúságú sorozatokat hoznak létre.

– Műszaksorozat-ember hozzárendelés. Végül itt az önálló műszakokat, vagy műszaksorozatokat tényleges alkalmazottakhoz rendelik.

A feladat egy részletesebb, általánosabb felbontása megtalálható [28]-ban.

5.1. Korlátozó szabályok

A szigorú, „kemény" szabályok főleg az EU, minisztériumok, önkormányzatok, szakszervezetek stb. által előírt szabályok, ezek betartása kötelező az ütemezés során. Az ajánlott, „puha" szabályok főleg személyi igények során kerülnek előtérbe (néhány szabadnap egymás után következzen, néhány szabadnap essen hétvégére stb.). Ezek betartása nem kötelező, de minősítik a megoldás „ jóságát", súlyozva bekerülhetnek a célfüggvénybe a kiértékelésnél.

Jellemző szigorú szabályfajták, amelyek a legtöbb alkalmazási területen előfordulnak:

– Adott az egymást követő két műszak közötti minimális pihenőidő.
– Maximálva van a heti munkaidő.
– Adott a heti minimális pihenőidő.
– Szabályozott, hogy legfeljebb hány egymást követő napon dolgozhat a munkavállaló szabadnap nélkül.
– Adott, hogy legalább hány szabadnapja legyen egy munkavállalónak egy hónapon belül.
– Adott, hogy hány szabadnapnak kell esnie bizonyos típusú napra (pl. hétvégére, vasárnapra stb.) adott időn (egy hónapon) belül.
– Adott, hogy bizonyos munkavállalók csak bizonyos típusú műszakot kaphatnak, vagy milyen típusból mennyit kaphatnak.

Jellemző „puha" szabályok, igények:

– Kerüljük az olyan kiosztást, ahol két szabadnap között csak egy munkanap van.
– Előnyben részesítjük, ha két szabadnap egymás után következik.
– Lehetőleg a szabadnapok egyenletesen legyenek elosztva a tervezési időszakban.
– Egyenletesen legyenek a műszaktípusok kiosztva az alkalmazottak között, vagy pont fordítva, lehetőleg azonos típusú műszakot kapjon egy alkalmazott egy adott időszakon belül (hét vagy hónap).

A buszos járművezetőkre vonatkozó szabályok egyfajta osztályozása megtalálható [60]-ban.

5.2. Kiértékelés

A műszakkiosztás minőségének meghatározása nem egyértelmű a szakirodalomban. Természetesen a cél mindenhol a költség minimalizálása, de a költség meghatározásához több tényezőt kell figyelembe venni. Egyrészt cél a szükséges alkalmazottak számának minimalizálása, feltételezve, hogy kevesebb ember alkalmazásával kisebb a költség [17, 18, 74]. Ugyanakkor több helyen az alkalmazottak száma kötött, ilyenkor a kiosztás minőségének növelése a cél, amit a puha szabályokhoz való illeszkedés határoz meg. Ezen szabályok megfelelően súlyozott kiértékelése határozza meg a kiosztás minőségét. Természetesen gyakran az emberek számának minimalizálása és a puha szabályok figyelembe vétele együttesen jelenik meg az optimalizálásban [19, 59], esetleg a be nem osztott alkalmazottakat tartalékba helyezik betegségek esetére [60]. Ugyanakkor a valós életben a kiosztás költségét nagyban befolyásolja az alkalmazottak szerződése, amely meghatározza a ledolgozandó órák számát. Ettől való eltérés alul-, illetve túlfoglalkoztatást jelent. Az előbbi azért költséges, mert a szerződésben foglalt órák alapján kapja a fizetését, holott nem dolgozott annyit, az utóbbi meg túlórát jelent, amelynek bérezése általában magasabb az alap órabérnél. Így ezek súlyozott minimalizálása a költség csökkenését jelenti [6]. Az nyilvánvaló, hogy az alul- és túlfoglalkoztatás csökkentése csak úgy érhető el, ha a kiosztás során felhasznált alkalmazottak száma és összetevője változik.

5.3. Megoldási módszerek

Ugyan a műszakkiosztási feladat az élet eltérő területein fordul elő, a főbb betartandó szabályok, illetve az optimalizálás során figyelembe vett költség- és célfüggvények nagyon hasonlóak. Így a különböző területen alkalmazott megoldási módszerek is elég általánosak a többi alkalmazási környezetbe való illesztéshez. A módszerek közötti különbség egyrészt inkább abból adódik, hogy hol húzzuk meg a határt a megoldás minősége, valamint a feladat mérete és a futási idő között.

Másrészt az alkalmazott optimalizálási algoritmusok változatossága jellemzi a szakirodalmat. Egyik megközelítés, hogy a feladatot visszavezetik halmazlefedési vagy partícionálási feladatra, ahol legenerálnak sok műszakkiosztást, majd ezekből kiválasztják azokat, amelyekkel a költség a legkisebb. Ezt az irányt követték például Gamache és szerzőtársai, akik 1999-es cikkükben [36] a halmazpartícionálást oszlopgenerálással oldották meg.

A feladat felírható folyamproblémaként is. Cappanera és szerzőtársai 2004-ben egy többtermékes folyamproblémaként oldottak meg légiközlekedés ütemezési feladatot, ahol a többtermékesség az adta, hogy háromféle kiosztást készítettek a kapitányoknak, pilótáknak és légi utaskísérőknek [17]. Jellemző még a kombinált megoldási módszerek használata. Yunes és társai [74] rámutattak, hogy az egészértékű programozási feladatként történő felírás csak kis feladatokra működik elfogadható időn belül, míg a pusztán korlátozó feltételes logikai programozás-

ként való felírás már hatékonyabb, de még mindig nem elfogadható valós méretű feladatokra. Azt azonban kimutatták, hogy ennek a két módszernek a kombinálásával készített hibrid módszer, ahol oszlopgenerálást alkalmaztak és az oszlopok generálását végezték korlátozó feltételes logikai programozással, már képes volt nagy feladatokra is hatékony megoldásokat adni elfogadható idő alatt.

További kombinált módszert alkalmaztak Caprara és szerzőtársai. Itt a megoldást egy konstruktív heurisztika adja, amely a kiosztás építése közben a mohó műszakválasztáshoz felhasználja egy Lagrange-féle relaxált egészértékű programozási feladat megoldását [18]. Iteratív kombinált heurisztikát alkalmaz Bellanti [11] és Nurmi [60]. Előbbi egy kezdeti kiosztást generál mohó módon, majd ezt javítja szomszédsági keresés módszerrel, ehhez tesztelt iteratív helyi keresést, illetve tabulistás módszert [11]. Nurmi és szerzőtársai [60] a megoldást két fázisban végzik: először a szabadnapokat osztják ki, majd utána a műszakkiosztást. Mindkét feladathoz ugyanazt a módszert alkalmazzák, amely egy mohó, populációalapú helyi keresés.

Egy kiosztás kiértékelése több tényezőből áll össze, így az optimalizálás gyakran több célfüggvényű optimalizálás. Erre példa Moz és szerzőtársai [59] evolúciós módszere, ahol két szempont szerint történik az optimalizálás. Ez a kettő az emberek elvárt számától való eltérés minimalizálása és a túlórák egyenletes elosztása. A generált megoldások kiértékelésénél e két cél szerinti Pareto-dominanciát használják fel.

5.4. Szétválasztási stratégiák

A fenti részfeladatokat közelebbről megvizsgálva láthatjuk, hogy különböző opciók lehetségesek az ütemezési rendszer struktúrájának megtervezésére. Két alapvető kérdés, amely fontos ennek a struktúrának a megtervezésével kapcsolatban:

– Hol legyen a határvonal a vezetőütemezés és a műszakkiosztás feladatai között, mely szabályokat melyik részfeladatnál vegyünk figyelembe? Alapvetően a legfőbb elv ezzel kapcsolatban az, hogy a dolgozókra vonatkozó olyan szabályok, rendelkezések, amelyek az egy napon belüli műszak kialakításában veendők figyelembe, azok a vezetőütemezés, míg a további feltételek a műszakkiosztás feladatkörébe tartoznak. Sajnos, elméletileg ennek a szabálynak a figyelembevételével is a vezetőütemezéssel a napi ütemezéseknek egy olyan szerkezetét kaphatjuk, amelyekből a műszakkiosztás már nem eredményezhet a szabályoknak megfelelő megoldást. A gyakorlatban ez a probléma inkább csak kisebb városok társaságainál életszerű, ott fordulhat elő realisztikus módon (jóval százezer fő alatti lakosság esetén), mivel a kisebb kombinációs lehetőség okozhat hozzárendelési problémákat egy olyan ütemezésben, ahol az ütemezés szerkezetének előírtnak kell lennie. Mivel a feladat ebben az esetben kisebb méretű, ez lehetővé teszi napi ütemezés helyett heti periódusok kialakítását, így csökkentve annak a kockázatát, hogy nem kapunk lehetséges megoldást.

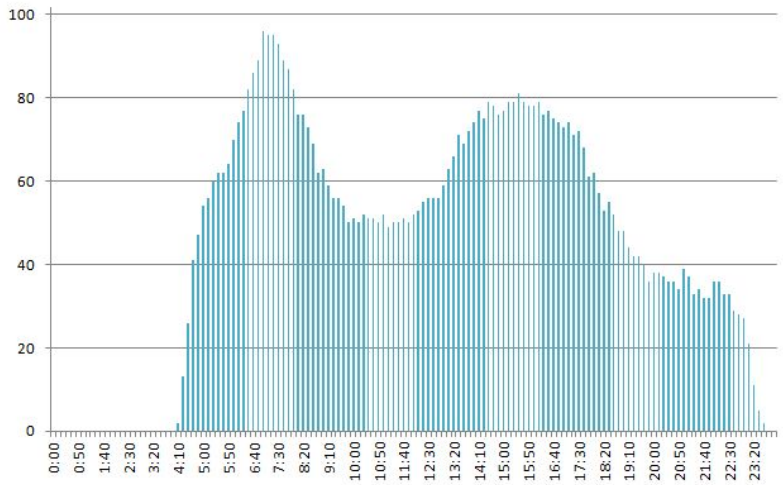
– A sorrend a járműütemezés és a járművezetők ütemezése között: mivel a járművek napi műszakjai menetrendi előírásokon alapulnak, így természetes módszerként adódik első fázisban az optimális járműütemezést elkészíteni, és azután a vezetőket hozzárendelni a járművekhez (megengedve a műszak során az eszközökön a vezetőcserét) olyan módon, hogy az összes, az emberekre vonatkozó szabály ki legyen elégítve. Ez a módszer ugyan hatékony, viszont robusztus számítógépes optimalizálási hátteret igényel. Éppen ezért a gyakorlatban, ahol nem használnak optimalizálási eszközöket automatizált tervezésre, vagy annak támogatására, az alkalmazott mérnöki „heurisztikák" megcserélik a két lépést: első lépésként (nem automatizált tervezéssel) gyakran a menetrend alapján emberműszakokat hoznak létre, figyelembe véve a járművezetőkre vonatkozó szabályokat, majd azután megfelelő járműveket rendelnek a kialakított műszakokhoz. Ennek a megközelítésnek az előnye az, hogy a feladat komplexitása jelentősen csökken, mivel a járművezető a műszakja alatt nem cserélhet járművet. Komoly hátrány ugyanakkor, hogy ez magasabb költségű ütemezéseket eredményezhet. Jellemzően, az egy napon használt járművek száma ilyenkor szignifikánsan nagyobb. A fentiek alapján egy optimalizálásorientált automatizált ütemezési rendszer esetén feltételezzük, hogy az járműütemezéssel indul, vagy ha nem, akkor ezzel egyidejűleg figyelnie kell a járműszabályokat is.

6. Néhány saját eredmény

Végül röviden megemlítjük, hogy a jelen tanulmány szerzői további (részben más társszerzőkkel közös) cikkeikben a közösségi buszközlekedés operatív tervezési feladataira adott – általában különböző gyakorlati kihívásokból fakadó – megoldásaikat tárgyalják az alábbi cikkekben, közleményekben: [4, 5, 6, 7, 9, 10, 20, 21, 70, 71].

Az itt tárgyalt felosztás alapján megvalósított, implementált fázisokat az [5] és a [10] publikációkban egy keretrendszerben mutattuk be. A gyakorlatban is egy olyan, nagy modulokból álló keretrendszernek nevezhető rendszert fejlesztettünk, amelynek moduljai megengedik olyan eljárások beágyazását, amelyek megfelelnek a fent említett követelményeknek. Amint az [10]-ben tárgyalt, ez a keretrendszer lehetőséget ad arra, hogy egy-egy moduljába különböző opciókat lehessen beépíteni. Így egy-egy modulon belül alternatív módszerek alkalmazhatók opcionális megoldásként, és néhány esetben kitérünk ezeknek a lehetőségeknek a tárgyalására és vizsgálatára is.

A járműütemezési feladatra a [20] és a [21] publikációkban az MDVSP-heurisztikák kapcsán tárgyalt, a [47] cikkben megadott változófixálási heurisztika néhány módosítását, továbbfejlesztését tárgyaltuk, különböző hasonló típusú, további heurisztikákat tárgyalva és elemezve.



7. ábra. Egy tipikus példa: egy munkanapon belül melyik időszakban hány jármű (és járművezető) szükséges minimálisan a járatok ellátásához. Ennek az ún. „kétpúpú teve" diagramnak a „púpjai", a reggeli és a délutáni csúcs (iskola- és munkakezdési, illetve befejezési időpontok) jellegzetesek munkanapokon.

A járműütemezési és jármű-hozzárendelési feladatokhoz kapcsolódóan tapasztalatunk szerint a konkrét, gyakorlati ütemezésben a szakirodalomban tárgyalt és alkalmazott modellek hátrányaként lehet megemlíteni, hogy csak azokat a szabályokat veszik figyelembe, amelyek a menetrendi járatokkal, az azokhoz megkívánt busztípusokkal és kapacitásokkal, valamint a menetrendi járatok közötti rezsijáratokkal kapcsolatosak. Nem lehet viszont olyan specifikus feltételeket beépíteni, amelyek valós alkalmazási környezetből származnak. A tömegközlekedés operatív feladatainak tervezésénél ilyen tipikus, járműspecifikus korlátozó feltételek a tankolási vagy parkolási előírások. Tankolási követelmények beépítését tárgyaltuk [7]-ben és [9]-ben. Az ezekben tárgyalt módszer előnye, hogy heterogén járműflotta tankolási feladatait is kezeli. Erre szükség lehet, amennyiben hosszú tankolási idejű, egy tankolással a dízel üzemenyagú járművekhez képest kis távolságot megtenni képes járművek (is) vannak a flottában.

Egy járművezető-barát, azaz olyan módszert tárgyalunk [5]-ben a jármű- és vezetőütemezési feladat iteratív megoldására, ahol a járművezetőknek nem kell a nap során járművet cserélni. Így csak egyazon járművet vezetik a nap során a műszakjuk alatt, kivéve osztott műszak esetén. Ez olyan napközbeni, több órás otthoni pihenő utáni visszatérést tartalmazó műszak, amely tulajdonképpen két,

független műszakrészből áll. A 7. ábra mutatja, hogy az osztott szolgálat miért szükséges általában munkanapokon: a reggeli és a délutáni csúcs járatainak ellátásához több jármű (és járművezető) szükséges, mint egyébként napközben.

A [71] publikáció a vezetőütemezésre ad a gyakorlatban használható heurisztikákat, míg [70] a vezetőütemezéshez kapcsolódó tevékenységekre egy általános keretmunkát.

A [6] publikációban a műszakkiosztási feladatra adott hosszú távú, több hetes intervallumokra adott megoldásainkat ismertetjük.

Köszönetnyilvánítás.

A kutatást a „Szuperszámítógép, a nemzeti virtuális laboratórium" című, TÁMOP-4.2.2.C-11/1/KONV-2012-0010 azonosítószámú projekt támogatta az Európai Unió és az Európai Szociális Alap társfinanszírozása mellett. A tanulmány a TÁMOP-4.2.1/B-09/1/KONV-2010-0003 Mobilitás és környezet: Járműipari, energetikai és környezeti kutatások a Közép- és Nyugat-Dunántúli Régióban projekt támogatásával jött létre, a projekt a Magyar Állam és az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Hivatkozások

[1] E. Abbink, J. van 't Wout, and D. Huisman: *Solving Large Scale Crew Scheduling Problems by using Iterative Partitioning*, In Proceedings of ATMOS2007 – 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization and Systems, 96–106, 2007.

[2] U. Aickelin, E.K. Burke, and J. Li: *Improved Squeaky Wheel Optimization for Driver Scheduling*, In Proceedings of the 9th International Conference on Parallel Problem Solving from Nature – PPSN IX, Lecture Notes in Computer Science, 4193, 182–191, Springer, 2006.

[3] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin: *Network Flows*, Chapter IV of the Handbooks in Operations Research and Management Science, Volume 1: Optimization, (eds. G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J. Todd), 211–369, North Holland, 1989.

[4] V. Árgilán, J. Balogh, J. Békési, B. Dávid, M. Krész, and A. Tóth: *A flexible system for optimizing public transportation*, In Proceedings of the 8th International Conference on Applied Informatics, Vol. **2**, 181–190, 2010.

[5] V. Árgilán, J. Balogh, J. Békési, B. Dávid, M. Krész, and A. Tóth: *Driver scheduling based on "driver-friendly" vehicle schedules*, In Operations Research Proceedings 2011, Selected Papers of the International Conference on Operations Research (OR 2011), 323–328, Springer, 2012.

[6] V. Árgilán, B. Dávid, Cs. Kemény, G. Pongrácz, and A. Tóth: *Greedy heuristics for driver scheduling and rostering*, In Proceedings of the 2010 Mini-Conference on Applied Theoretical Computer Sciences, Koper, Slovenia, 13-14 October, 2010, University of Primorska Press, 101–108, 2011. (ISBN: 978-961-6832-10-6)

[7] V. Árgilán, J. Balogh, J. Békési, B. Dávid, G. Galambos, M. Krész, and A. Tóth: *An Assignment Model for Real-World Vehicle Scheduling Problem with Refueling*, közlésre benyújtva, 2013.

[8] M. Ball, L. Bodin, and R. Dial: *A matching based heuristic for scheduling mass transit crews and vehicles*, Transportation Science, **7**, 4–31, 1983.

[9] J. Balogh, J. Békési, G. Galambos, and M. Krész: *Model and Algorithm for a Vehicle Scheduling Problem with Refueling*, In Proceedings of the 9th Workshop on Models and Algorithms for Planning and Scheduling Problems, 229–231, 2009.

[10] J. Békési, A. Brodnik, D. Pash, and M. Krész: *An integrated framework for bus logistic management: case studies*, In Logistik Management, Physica-Verlag, 389–411, 2009.

[11] F.C.G. Bellanti, F. Della Croce, and R. Tadei: *A greedy-based neighborhood search approach to a nurse rostering problem*, European Journal of Operational Research, **153(1)**, 28–40, 2004.

[12] A.A. Bertossi, P. Carraresi, and G. Gallo: *On Some Matching Problems Arising in Vehicle Scheduling Models*, Networks, **17**, 271–281, 1987.

[13] L. Bodin, B. Golden, A. Assad, and M. Ball: *Routing and Scheduling of Vehicles and Crews: The State of the Art*, Computers and Operations Research, **10**, 63–211, 1983.

[14] R. Borndörfer: *Discrete Optimization in Public Transportation*, IB-Report 08–56, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany, 2008.

[15] R. Borndörfer, A. Löbel, and S. Weider: *A bundle method for integrated multi-depot vehicle and duty scheduling in public transit*, In Computer-aided Systems in Public Transport, (eds. M. Hickman, P. Mirchandani, and S. Voß), Lecture Notes in Economics and Mathematical Systems, 600, 3–24, Springer-Verlag, Heidelberg, 2008.

[16] S. Bunte and N. Kliewer: *An overview on vehicle scheduling models*, Journal of Public Transport, **1(4)**, 299–317, 2009.

[17] P. Cappanera and G. Gallo: *A Multicommodity Flow Approach to the Crew Rostering Problem*, Operations Research, **52(4)**, 583–596, 2004.

[18] A. Caprara, P. Toth, D. Vigo, and M. Fischetti: *Modeling and Solving the Crew Rostering Problem*, Operations Research, **46**, 820–830, 1998.

[19] B.M.W. Cheng, J.H.M. Lee, and J.C.K. Wu: *A nurse rostering system using constraint programming and redundant modeling*, IEEE Transactions in Information Technology in Biomedicine, **1(1)**, 44–54, 1997.

[20] B. Dávid: *Heuristics for the Multiple-Depot Vehicle Scheduling Problem*, In Proceedings of the 2010 Mini-Conference on Applied Theoretical Computer Science, Koper, Slovenia, 13–14 October, 2010, University of Primorska Press, 2011, pp. 23–28. (ISBN: 978-961-6832-10-6)

[21] B. Dávid and M. Krész: *Application Oriented Variable Fixing Methods for the Multiple Depot Vehicle Scheduling Problem*, Acta Cybernetica-Szeged, **21(1)**, 53–73, 2013.

[22] S.W. de Groot and D. Huisman: *Vehicle and crew scheduling: Solving large real-world instances with an integrated approach*, In Computer-aided Systems in Public Transport, (eds. M. Hickman, P. Mirchandani, and S. Voß), Lecture Notes in Economics and Mathematical Systems 600, 43–56, Springer-Verlag, Heidelberg, 2008.

[23] M. Dell' Amico, M. Fischetti, and P. Toth: *Heuristic algorithms for the multiple depot vehicle scheduling problems*, Management Science, **39**, 115–125, 1993.

[24] G. Desaulniers and M.D. Hickman: *Public Transit*, In Handbook in OR & MS, (eds. C. Barnhart and G. Laporte), Vol. **14**, Chapter 2, Elsevier B.V., 2007.

[25] M. Desrochers, J. Gilbert, M. Sauve, and F. Soumis: *CREW-OPT: subproblem modeling in a column generation approach to urban crew scheduling*, In Computer-aided transit scheduling, (eds. M. Desrochers and J.-M. Rousseau), Lecture Notes in Economics and Mathematical Sytems, **386**, 395–406, Springer-Verlag, Berlin, 1992.

[26] M. Desrochers and F. Soumis: *A column generation approach to the urban transit crew scheduling problem*, Transportation Science, **23(1)**, 1–13, 1989.

[27] T.G. Dias, J.P. de Sousa, and J.F. Cunha: *Genetic algorithms for the bus driver scheduling problem: a case study*, Journal of the Operational Research Society, **53(3)**, 324–335, 2002.

[28] A.T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier: *Staff scheduling and rostering: A review of applications, methods and models*, European Journal of Operational Research, **153**, 3–27, 2004.

[29] *European Union.* Regulation (EC) No. 561/2006 of the European Parliament and of the Council of 15 March 2006 on the harmonisation of certain social legislation relating to road transport and amending Council Regulations (EEC) 3821/85 and (EC) 2135/98 and repealing Council Regulation (EEC) 3820/85. *Official Journal of the European Union*, L **102**, 11.04.2006.

[30] J.C. Falkner and D.M. Ryan: *EXPRESS: set partitioning for bus crew scheduling in Christchurch*, In Computer-aided transit scheduling, (eds. M. Desrochers and J.-M. Rousseau), Lecture Notes in Economics and Mathematical Systems, 386. 359–378, Springer-Verlag, Berlin, 1992.

[31] R. Freling, C.G.E. Boender, and J.M.P. Paixao: *An integrated approach to vehicle and crew scheduling*, Technical Report 9503/A, Econometric Institute, Erasmus University Rotterdam, Rotterdam, 1995.

[32] R. Freling: *Models and Techniques for Integrating Vehicle and Crew Scheduling*, PhD thesis, Tinbergen Institute, Erasmus University Rotterdam, 1–151, 1997.

[33] R. Freling, A.P.M. Wagelmans, and J.M.P. Paixao: *An overview of models and techniques for integrating vehicle and crew scheduling.* In Computer-Aided Transit Scheduling, (ed. N.H.M. Wilson), Lecture Notes in Economics and Mathematical Systems, **471**, 441–460, Springer, Berlin, 1999.

[34] R. Freling, D. Huisman, and A.P.M. Wagelmans: *Models and algorithms for integration of vehicle and crew scheduling.* Journal of Scheduling, **6**, 63–85, 2003.

[35] A. Gaffi and M. Nonato: *An integrated approach to the extra-urban crew and vehicle scheduling problem*, In Computer-Aided Transit Scheduling, (ed. N.H.M. Wilson), Lecture Notes in Economics and Mathematical Systems, **471**, 103–128, Springer, Berlin, 1999.

[36] M. Gamache, F. Soumis, G. Marquis, and J. Desrosiers: *A column generation approach for large-scale aircrew rostering problems*, Operations Research, **47(2)**, 247–263, 1999.

[37] M.R. Garey and D.S. Johnson: *Computers and Interactability: A Guide to the Theory of NP-Completness*, Freeman, San Fransisco, 1979.

[38] V. Gintner, I. Steinzen, and L. Suhl: *A time-space network based approach for integrated vehicle and crew scheduling in public transport*, In Proc. EWGT2006 Joint Conf., (eds. M. Binetti, F. Civitelle, E. De Liddo, M. Dell'orco, M. Ottomanlli), Bari, Italy, 371–377, 2006.

[39] V. Gintner, N. Kliewer, and L. Suhl: *A Crew Scheduling Approach for Public Transit Enhanced with Aspects from Vehicle Scheduling*, In Computer-aided Systems in Public Transport, (eds. M. Hickman, P. Mirchandani, and S. Voß), Lecture Notes in Economics and Mathematical Systems, **600**, 25–42, Springer-Verlag, Heidelberg, 2008.

[40] N. Guerinik and M.V. Caneghem: *Solving Crew Scheduling Problems by Constraint Programming*, In Proceedings of the 1st Conference of Principles and Practice of Constraint Programming, pp. 481–498, 1995.

[41] K. Haase and C. Friberg: *An exact branch and cut algorithm for the vehicle and crew scheduling problem*, In Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems, **471**, (ed. N.H.M. Wilson), 63–80, Springer, Berlin, 1999.

[42] K. Haase, G. Desaulniers, and J. Desrosiers: *Simultaneous vehicle and crew scheduling in urban mass transit systems*, Transportation Science, **35(3)**, 286–303, 2001.

[43] A. Hadjar, O. Marcotte, and F. Soumis: *A Branch-and-Cut Algorithm for the Multiple Depot Vehicle Scheduling Problem*, Tech. Rept. G–2001–25, Les Cahiers du Gerad, Montreal, 2001.

[44] C. Hane, C. Barnhart, E.L. Johnson, R.E. Marsten, G.L. Nemhauser, and G. Sigismondi: *The fleet assignment problem: Solving a large integer program*, Mathematical Programming, **70(2)**, 211–232, 1995.

[45] D. Huisman: *Integrated and Dynamic Vehicle and Crew Scheduling*, PhD thesis, Erasmus University of Rotterdam, Rotterdam, 2004.

[46] D. Huisman, R. Freling, and A.P.M. Wagelmans: *Multiple-depot integrated vehicle and crew scheduling*, Transportation Science, **39**, 491–502, 2005.

[47] N. Kliewer, T. Mellouli, and L. Suhl: *Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice*, OR Spectrum, **27(4)**, 507–523, 2005.

[48] N. Kliewer, T. Mellouli, and L. Suhl: *A time-space network based exact optimization model for multi-depot bus scheduling*, European Journal of Operational Research, **175**, 1616–1627, 2006.

[49] A. Kokott and A. Löbel: *Lagrangean Relaxations and Subgradient Methods for Multiplie-Depot Vehicle Scheduling Problems*, ZIB-Report 96–22, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, Germany, 1996.

[50] R.S.K. Kwan, A.S.K. Kwan, and A.S.K. Wren: *Evolutionary Driver Scheduling with Relief Chains*, Evolutionary Computation, **9**, 445–460, 2001.

[51] B. Laurent and J-K. Hao: *Simultaneous Vehicle and Crew Scheduling for Extra Urban Transports*, In New Frontiers in Applied Artificial Intelligence, (eds. N.T. Nguyen, L. Borzemski, A. Grzech, and M. Ali), Lecture Notes in Computer Science Volume, **5027**, 466–475, 2008.

[52] J. Li: *A Self-Adjusting Algorithm for Driver Scheduling*, Journal of Heuristics, **11**, 351–367, 2005.

[53] J-Q. Li and K.L. Head: *Sustainability provisions in the bus-scheduling problem*, Transportation Research, Part D, **49**, 50–60, 2009.

[54] A. Löbel: *Optimal Vehicle Scheduling in Public Transit*, Ph.D. thesis, Technische Universitaet at Berlin, 1997.

[55] A. Löbel: *Vehicle Scheduling in Public Transit and Lagrangian Pricing.* Management Science, **44**, 1637–1649, 1998.

[56] M. Meilton: *Selecting and implementing a computer aided scheduling system for a large bus company*, Algorithms: Combinatorial Analysis. In Computer-Aided Scheduling of Public Transport, (eds. S. Voss and J.R. Daduna), 203–214, Springer-Verlag, Berlin, 2001.

[57] M. Mesquita and J. Paixao: *Multiple Depot Vehicle Scheduling Problem: A New Heuristic Based on Quasi-Assignment Algorithms*, In Computer-Aided Transit Scheduling, (eds. M. Desrochers and J.-M. Rousseau), Lecture Notes in Economics and Mathematical Systems, **386**, 167–180, Springer-Verlag, Berlin, 1992.

[58] M. Mesquita and A. Paias: *Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem*, Computers and Operations Research, **35(5)**, 1562–1575, 2008.

[59] M. Moz, A. Respcio, and M. Vaz Pato: *Bi-objective Evolutionary Heuristics for Bus Drivers Rostering*, Working Paper 1–2007, Centro de Investigaao Operacional, Universidade de Lisboa, 2007.

[60] K. Nurmi, J. Kyngas, and G. Post: *Driver Rostering for Bus Transit Companies*, Engineering Letters, **19(2)**, 125–132, 2011.

[61] A.-S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman: *Comparison of Heuristic Approaches for the Multiple Depot Vehicle Scheduling Problem*, Journal of Scheduling, **12(1)**, 17–30, 2009.

[62] A. Rabl: *Environmental benefits of natural gas for buses*, Transportation Research, Part D, **7**, 391–405, 2002.

[63] C.C. Ribeiro and F. Soumis: *A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem*, Operations Research, **42(1)**, 41–52, 1994.

[64] J.L. Saha: *An algorithm for bus scheduling problems*, Operational Research Quarterly, **21(4)**, 463–474, 1972.

[65] M. Segal: *The operator-scheduling problem: A network-flow approach*, Operations Research, **24**, 808–823, 1974.

[66] B.M. Smith and A. Wren: *A bus crew scheduling system using a set covering formulation*, Transportation Research, 22A, 97–108, 1988.

[67] I. Steinzen: *Instances for integrated vehicle and crew scheduling problems with multiple depots*, Online elérhető: http://dsor.uni-paderborn.de/index.php?id=bustestset& L=0.

[68] I. Steinzen, V. Gintner, L. Suhl, and N. Kliewer: *A Time-Space Network Approach for the Integrated Vehicle- and Crew-Scheduling Problem with Multiple Depots*, Transportation Science, **4(3)**, 367–382, 2010.

[69] U.U. Suhl, S. Friedrich, and V. Waue: *Progress in solving large scale multi-depot multi-vehicle-type bus scheduling problems with integer programming*, Wirtschaftinformatik Proceedings, Paper **81**, 2007. http://aisel.aisnet.org/wi2007/81

[70] A. Tóth and M. Krész: *A flexible framework for driver scheduling*, In Proceedings of the 11th International Symposium on Operational Research in Slovenia – SOR'11, 341–346, 2011. (ISBN: 978-961-6165-35-8)

[71] A. Tóth and M. Krész: *An efficient solution approach for real-world driver scheduling problems in urban bus transportation*, Central European Journal of Operations Research, **21**(Supplement 1), 75–94, 2013. DOI: 10.1007/s10100-012-0274-3.

[72] A. Wren, S. Fores, A.S.K. Kwan, R.S.K. Kwan, M.E. Parker, and L. Proll: *A flexible system for scheduling drivers*, Journal of Scheduling, **6(5)**, 437–455, 2003.

[73] A. Wren and B.M. Smith: *Experiences with a crew scheduling system based on set covering*, In Computer-aided transit scheduling, (eds. J.R. Daduna and A. Wren), Lecture Notes in Economics and Mathematical Systems, **308**, 104–118, Springer-Verlag, Berlin, 1988.

[74] T. Yunes, A. Moura, and de C. Souza. *Hybrid Column Generation Approaches for Solving Real World Crew Management Problems*, Transportation Science, **39(2)**, 273–288, 2005.

(Beérkezett: 2013. október 4.)

ÁRGILÁN VIKTOR
BALOGH JÁNOS
BÉKÉSI JÓZSEF
DÁVID BALÁZS
GALAMBOS GÁBOR
KRÉSZ MIKLÓS
TÓTH ATTILA
Szegedi Tudományegyetem
Juhász Gyula Pedagógusképző Kar
Informatika Alkalmazásai Tanszék
6701 Szeged, Pf. 396.
{gilan,balogh,bekesi,davidb,galambos,kresz,attila}@jgypk.u-szeged.hu

SCHEDULING PROBLEMS IN THE OPERATIVE PLANNING OF PUBLIC BUS TRANSPORTATION

Viktor Árgilán, János Balogh, József Békési, Balázs Dávid, Gábor Galambos, Miklós Krész, Attila Tóth

Optimization of operative costs is an important aim of public transportation companies. Their planning process can be aided by the optimization of scheduling of vehicle journeys, vehicle schedules, and driver shifts. The problem is a complex optimization problem. For this reason, we deal with the whole problem in three separate phases: vehicle scheduling, driver scheduling and driver rostering. Each sub-problem is NP-hard, and there was no possibility to obtain the optimal solution for large or even middle-sized problems until the last decade. Because of the increase in computational speed and the development of new scheduling methods, researchers today can handle larger problems also. We deal with the mathematical models and solution of these efficient methods. We also refer to our solution methods and the role of the special constraints is analyzed based on our experiences.

Alkalmazott Matematikai Lapok **31** (2014), 41–98.

FOKSOROZATOK PÁRHUZAMOS LESZÁMLÁLÁSA

IVÁNYI ANTAL, KÁSA ZOLTÁN

Az egyszerű gráfok foksorozatainak tesztelése, előállítása és leszámlálása gazdag irodalommal rendelkezik. Cikkünkben párhuzamos algoritmusokat ismertetünk, melyek segítségével leszámláltuk az egyszerű gráfok ($G(n)$) és az izolált csúcsot nem tartalmazó egyszerű gráfok ($G_z(n)$) foksorozatait $n = 24, \ldots, 31$ csúcs esetén.

1. Bevezetés

A gyakorlatban gyakran előforduló probléma különböző objektumok rangsorolása (példák találhatóak [89, 95]-ben). Az egyik módszer az, hogy az objektumokat páronként összehasonlítjuk, az összehasonlítás alapján – rendszerint különböző, de esetenként azonos számú – pontokat rendelünk az összehasonlított objektumokhoz, és az összegyűjtött pontjaik száma (vagy a nyert és vesztett pontok számának különbsége) alapján rangsoroljuk őket. Az azonos pontszámú eset irányítatlan [14, 15, 16, 17, 20, 22, 23, 24, 25, 26, 33, 34, 35, 39, 44, 47, 48, 53, 56, 57, 59, 60, 62, 63, 61, 66, 76, 77, 81, 82, 89, 91, 92, 93, 94, 95, 97, 99, 100, 105, 106, 110, 111, 112, 113, 114, 118, 119, 120, 121, 122, 126, 130, 131, 132, 133, 134, 135, 136, 137, 143, 144, 158, 159, 162, 164, 166, 167, 169, 170, 171, 172, 173, 184, 190, 191], míg a különböző pontszámú eset irányított gráfokkal [1, 4, 6, 7, 8, 9, 10, 11, 12, 13, 18, 36, 21, 27, 28, 29, 30, 31, 36, 37, 38, 40, 43, 50, 51, 52, 55, 58, 64, 67, 68, 69, 70, 71, 72, 73, 75, 80, 83, 84, 85, 86, 87, 88, 88, 90, 97, 98, 104, 103, 108, 109, 117, 123, 124, 129, 137, 138, 139, 140, 141, 142, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 160, 163, 165, 168, 172, 174, 175, 176, 179, 180, 185, 186, 189] kapcsolatos problémákra vezet.

Különösen gazdag irodalommal rendelkezik az az eset, amikor az eredmények egyszerű gráfként reprezentálhatóak (azaz az összehasonlítás eredményeképpen vagy mindkét objektum nulla, vagy mindkettő egy pontot kap), és a probléma a potenciális foksorozatok [71, 187] tesztelése, rekonstruálása és leszámlálása. Havel 1955-ben [78], Erdős és Gallai 1960-ban [56], Hakimi 1962-ben [74], Tripathi és munkatársai 2010-ben [183] javasoltak egy-egy algoritmust annak eldöntésére, hogy egy nemnegatív egészekből álló sorozat lehet-e egy egyszerű gráfhoz tartozó foksorozat. Az algoritmusaik futási ideje legrosszabb esetben $\Omega(n^2)$.

2007-ben Takahashi [177], 2009-ben Hell és Kirkpatrick [79], 2011-ben Iványi és munkatársai [95], 2012-ben pedig Király [107] egymástól függetlenül javasolták az Erdős–Gallai-algoritmus lineáris futási idejű változatát, 2012-ben pedig Iványi [87] a Havel–Hakimi-algoritmus tesztelő változatát, melyek futási ideje legrosszabb esetben $\Theta(n)$.

A klasszikus Havel–Hakimi és Erdős–Gallai-tételekre számos új bizonyítás ismert [32, 45, 116, 181, 182, 183]. Kiterjesztéseik $(0, b)$-gráfokra [46, 145] és (a, b)-gráfokra [84, 85, 87, 97, 151] szintén ismertek. Bemutatjuk a Havel–Hakimi-algoritmus tesztelő változatait [87], és összehasonlítjuk a korábbi lineáris tesztelő algoritmusokkal.

Léteznek korai párhuzamos eredmények is, lásd például [3, 115, 149, 168, 174]. Az Erdős–Gallai-enumerating algoritmus (EGE1) [95] párhuzamos verzióját használtuk fel a $24, \ldots, 31$ csúcsú egyszerű gráfok foksorozatainak leszámlálására.

Megemlítjük, hogy más szerzők is felhasználták algoritmusunkat különböző, foksorozatokkal kapcsolatos problémák megoldására [16, 41, 54].

Legyenek l, m, n, u egész számok, $u \geq l$, $m \geq 1$, $n \geq 1$. Ha $l \leq b_i \leq u$ $(i = 1, \ldots, m)$, akkor az egész számokból álló $b = b_1, \ldots, b_m$ sorozatot (l, u, m)-*korlátosnak* nevezzük. Ha b $(0, n-1, n)$-korlátos, akkor n-korlátosnak, ha az $n - 1 \geq b_1 \geq \cdots \geq b_n \geq 0$ feltételt is kielégíti, akkor n-*szabályosnak* nevezzük. Ha egy n-szabályos sorozat elemeinek összege páros, akkor n-*páros*. Ha létezik olyan n csúcsú egyszerű gráf, amelynek b a foksorozata, akkor azt mondjuk, hogy a b sorozat n-*gráfos* [71, 188]. Ha nincs ilyen gráf, akkor b *nemgráfos*. Egy n-szabályos b sorozat első i elemét az i-hez tartozó *fejnek*, az azt követő $n - i$ elemét pedig az i indexhez tartozó *faroknak* nevezzük.

Cikkünk fő célja a lineáris Erdős–Gallai-algoritmus párhuzamos megvalósításának bemutatása. Bár a probléma önmagában is érdekes, számunkra a fő motivációt Frank András [65, Research problem 2.3.1] monográfiájában szereplő kérdés megválaszolása jelenti: „*Döntsük el, hogy egy n egész számból álló sorozat lehet-e egy n csapatból álló labdarúgó bajnokság végeredménye.*" A potenciális futballsorozatok tesztelésének és rekonstruálásának fontos részproblémája a döntetlen sorozatok kezelése. Mivel az „Ez a sorozat gráfos?" és az „Ez a sorozat egy labdarúgó döntetlen sorozat?" kérdések ekvivalensek (lásd [87, 99, 115, 121, 165]), ezért a gyors válasz létfontosságú számunkra.

Cikkünk felépítése a következő. A bevezető 1. rész után a 2. részben bemutatjuk a párhuzamos programunk alapjául szolgáló lineáris ugró Erdős–Gallai-algoritmust (EGLJ), majd a . részben ismertetjük az EGLJ-algoritmus leszámláló változatát. A . részben a feladat szabályos sorozatokon alapuló felbontását, míg az . részben a páros sorozatokon alapuló felbontását mutatjuk be. A 6. részben az EGLJ-algoritmus párhuzamos változatát (EGP), a 7. részben pedig a grafikus és nem grafikus sorozatok átugrásának lehetőségeit elemezzük, végül a 8. részben összegezzük az eredményeket.

2. Lineáris ugró Erdős–Gallai-algoritmus (EGLJ)

Korábbi cikkeinkben [87, 89, 95] ismertettük a klasszikus Havel–Hakimi (HH) [74, 78] és Erdős–Gallai-algoritmusokat (EG) [56], valamint ezek különböző javításait. Fontos megjegyezni, hogy HHL lineáris csak teszteli a megvizsgált sorozatokat, de nem állít elő a gráfos sorozatoknak megfelelő gráfot. Ugyanakkor a helyreállító változat mindenképpen $\Omega(n^2)$ időt igényel.

2011-ben [95] bemutattuk a HH-algoritmus (HH) néhány gyorsított változatát, majd 2012-ben [87] a lineáris Havel–Hakimi-algoritmust (HHL).

Jelen cikkünkben táblázatokkal, valamint grafikonokkal összehasonlítjuk a HH- és EG-algoritmusok különböző változatainak futási idejét.

A cikk programjaiban a [49] tankönyvben leírt pszeudokód konvenciókat követjük.

A HH-algoritmus gyorsított változatai közül csak a HHT-algoritmus kódját ismételjük meg, mivel közreadjuk annak elemzését, mennyi esély van arra, hogy a HHT már az első lépésben felismer egy nemgrafikus n-szabályos sorozatot. Itt és a továbbiakban n a sorozat hosszát (a gráf csúcsainak számát) jelöli, $b = b_1, \ldots, b_n$ pedig a vizsgálandó $(0, 1, n)$-szabályos sorozatot.

2.1. Algoritmus. HAVEL-HAKIMI-TESTING (n, b)

Bemenet: n: csúcsok száma ($n \geq 1$);
$b = b_1, \ldots, b_n$: a megvizsgálandó n-szabályos sorozat.
Kimenet: 0 vagy 1: 1, ha b gráfos, és 0 egyébként.
Munkaváltozó: i: ciklusváltozó.

```
1. for i = 1 to n − 1                          // 1–6. sor: b elemeinek tesztelése
2.     if i + b_i > n vagy b_{i+b_i} = 0        // 2–3. sor: b nem grafikus
3.         return 0
4.     for j = i + 1 to i + b_i
5.         b_j = b_j − 1
6.     b_{i+1}, ..., b_n rendezése nemnövekvő sorrendbe
7. return 1                                    // 7. sor: b grafikus
```

Ebben a HHT-algoritmusban az eredeti HH-algoritmust kiegészítettük a 3. sorban végzett, a bemenő sorozat minden elemére legfeljebb konstans időt igénylő ellenőrzésével, amely jelzi, ha a sorozat rövidsége, vagy a nullák nagy száma miatt már a legnagyobb fokszámú csúcsot sem tudjuk a szükséges számú másik csúccsal összekötni.

Ez az ellenőrzés a bemenetként szóbajövő [89, 95]

$$R(0, n-1, n) = \binom{2n-1}{n-1} \tag{1}$$

$(0, n-1, n)$-szabályos sorozat közül pontosan azokat szűri ki, amelyekben $b_1 = n-1$ és $b_n = 0$, vagy $b_1 = n-2$ és $b_{n-1} = 0$ vagy $\dots$ vagy $b_1 = 1$ és $b_2 = 0$. Így az (1) képlet segítségével azt kapjuk, hogy a kiszűrt sorozatok $F(n)$ száma

$$F(n) = \sum_{i=1}^{n-1} F_i(n) = \sum_{i=1}^{n-1} \binom{2i-1}{i-1}.$$

Az 1. táblázat tartalmazza a csapatok n számát, az $n-1$ ($F_{n-1}(n)$), az $n-2$ ($F_{n-2}(n)$) és az $n-3$ ($F_{n-2}(n)$) elemmel kezdődő, valamint az összesen kiszűrt sorozatok $F(n)$ számát $n = 1, \dots, 32$ csapat esetén.

A következő 2.2. algoritmus EG leggyorsabb, általunk ismert soros változata (ez a [125] dolgozatban leírt algoritmus egyszerűsített változata). Az algoritmus alapja a következő állítás.

2.1. Tétel. *Legyen $n \geq 1$ és $b = b_1, \dots, b_n$ egy n-szabályos sorozat.*

A b sorozat akkor és csak akkor gráfos, ha elemeinek összege páros, továbbá ha $i > w$, akkor

$$H_i \leq i(i-1) + H_n - H_i$$

és ha $i \leq w_1$, akkor

$$H_i \leq i(i-1) + i(w_i - i) + H_n - H_{w_i}.$$

Bizonyítás. Lásd [95]. □

2.2. Tétel. (Iványi, Lucz, Móri, Sótér, 2011 [95]) *Az* Erdős-Gallai-Linear-Jumping $\Theta(n)$ *idő alatt dönti el, hogy egy n-szabályos $b = b_1, \dots, b_n$ sorozat gráfos-e.*

Bizonyítás. Lásd [95]. □

2.2. Algoritmus. Erdős-Gallai-Linear-Jumping (n, F)

Bemenet. n: a foksorozat hosszát adja meg;
b: az ellenőrizendő foksorozat.
Kimenet. ha a b sorozat grafikus, akkor 1, egyébként pedig 0.

```
1. H_1 = b_1                          // 1. sor: H_1 számítása
2. for i = 2 to n                     // 2–3. sor: H értékeinek számítása
3.       H_i = H_{i-1} + b_i
4. if H_n páratlan                    // 4–5. sor: paritás ellenőrzése
5.       return 0                     // 4. sor: hibás sorozat elutasítása
6. w = n                              // 5. sor: súlypont kezdeti értékének beállítása
7. i = 1                              // 7–15. sor: sorozat ellenőrzése
```

```
8.  while b_i = b_i + 1 and i ≤ n − 1        // 8–9. sor: ellenőrzőpont ellenőrzése
9.      i = i + 1
10.     while w > 1 and f_i ≤ i                  // 10–11. sor: súlypont frissítése
11.         w = w − 1
12.     if w < i                  // 12–13. sor: súlypont ellenőrzőpont előtt van
13.         while w > 1 and i < 1            // 13–14. sor: súlypont frissítése
14.     if w > i                  // 14-15. sor: súlypont ellenőrzőpont után van
15.         if H_i > H_n − H_i + i(i − 1) // 15–16. sor: rossz sorozat elutasítása
16.             return 0
17.     else if H_i > H_n − H_w + i(i − 1) // 17–18. sor: rossz sorozat elutasítása
18.             return 0
19. return 1                                 // 19. sor: jó sorozat elfogadása
```

Egy (szabályos) páros $s = s_1, \ldots, s_n$ sorozatot *nullamentesnek* nevezünk, ha $s_n > 0$. A 2. táblázatban láthatjuk a tesztelt nullamentes sorozatok számát ($E_z(n)$), valamint az egy sorozatra jutó átlagos tesztelés idejét mikroszekundumban az EGL ($T_{\mathrm{EGL}}(n)/E_z(n)$), EGLJ ($T_{\mathrm{EGLJ}}(n)/E_z(n)$) és HHL ($T_{\mathrm{HHL}}(n)/E_z(n)$) esetén, ahol $n = 10, \ldots, 19$. Az $n = 1, \ldots, 9$ értékek nem szerepelnek a táblázatban, mivel a futási idejüket a mérés során programunk nullára kerekítette.

Az 1. ábrán láthatjuk az EGL, EGLJ és HHL átlagos futási idejét a csúcsok számának függvényében. A háromszögek mutatják az $(n, T(n))$ párokat a lineáris Erdős–Gallai-algoritmus (EGL) esetében, a négyzetek a lineáris ugró Erdős–Gallai-algoritmus (EGLJ) esetében, míg a gyémántok a lineáris Havel–Hakimi-algoritmust (HHL) jelzik.

A 3. táblázatban található a nullamentes gráfos sorozatok $G_z(n)$ száma, valamint a nullamentes sorozatok ellenőrzésének átlagos műveletszáma az EGL ($O_{\mathrm{EGL}}(n)/E_z(n)$), EGLJ ($O_{\mathrm{EGLJ}}(n)/E_z(n)$) és HHL ($O_{\mathrm{HHL}}(n)/E_z(n)$) esetén, ahol $n = 2, \ldots, 19$.

A 2. ábrán láthatjuk az EGL, EGLJ és HHL átlagos műveletszámát a csúcsok számának függvényében. A (honlapon lévő változatban zöld) háromszögek mutatják az $(n, O(n))$ párosokat a lineáris Erdős–Gallai-algoritmus (EGL) esetében, a (piros) négyzetek a lineáris ugró Erdős–Gallai-algoritmus (EGLJ) esetén, míg a (kék) gyémántok a lineáris Havel–Hakimi-algoritmust (HHL) jelzik.

Műveletnek számítottunk minden összehasonlítást, összeadást, kivonást, szorzást, osztást, maradékképzést és értékadást. Kivételt képeztek a ciklusok törzsében szereplő indexváltozók. Például a $H[i] - i \cdot (i-1) > R$ parancs elvégzéséhez három műveletre van szükség: a $H[i] - i \cdot (i-1)$ kivonásra, az $i \cdot (i-1)$ szorzásra, valamint a $H[i] - i \cdot (i-1) > R$ összehasonlításra. Az $i - 1$ típusú kivonásokat *nem* számoltuk, mert az i a hozzá tartozó ciklus indexváltozója.

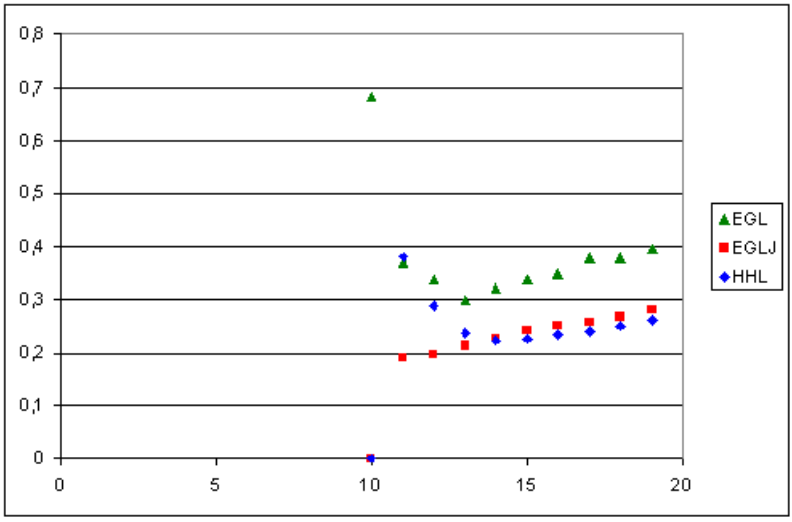
Példaként tekintsük részletesen az $(1, 1)$ nullamentes sorozat tesztelését. A példa az EGL [95], az EGLJ [125], valamint a HHL [87] pszeudokódján alapul.

n	$F_{n-1}(n)$	$F_{n-2}(n)$	$F_{n-3}(n)$	$F(n)$	$\frac{100F(n)}{R(n)}$
2	1	—	—	1	33.3333
3	3	1	—	4	40.0000
4	10	3	1	14	40.0000
5	35	10	3	49	38.8889
6	126	35	10	175	37.8788
7	462	125	35	637	37.1212
8	1 716	462	126	2 353	36.5657
9	6 435	1 716	462	8 788	36.1497
10	24 310	6 435	1 716	33 098	35.8289
11	92 378	24 310	6 435	125 476	35.5742
12	352 716	92 378	24 310	478 192	35.3671
13	1 352 078	352 716	92 378	1 830 270	35.1955
14	5 200 300	1 352 078	352 716	7 030 570	35.0507
15	20 058 300	5 200 300	1 352 078	27 088 870	34.9269
16	77 558 760	20 058 300	5 200 300	104 647 630	34.8198
17	300 540 195	77 558 760	20 058 300	405 187 825	34.7263
18	1 166 803 110	300 540 195	77 558 760	1 571 990 935	34.6439
19	4 537 567 650	1 166 803 110	300 540 195	6 109 558 585	34.5707
20	17 672 631 900	4 537 567 650	1 166 803 110	23 782 190 485	34.5053
21	68 923 264 410	17 672 631 900	4 537 567 650	92 705 454 895	34.4465
22	269 128 937 220	68 923 264 410	17 672 631 900	361 834 392 115	34.3933
23	1 052 049 481 860	269 128 937 220	68 923 264 410	1 413 883 873 975	34.3450
24	4 116 715 363 800	1 052 049 481 860	269 128 937 220	5 530 599 237 775	34.3008
25	16 123 801 841 550	4 116 715 363 800	1 052 049 481 860	21 654 401 079 325	34.2604
26	63 205 303 218 876	16 123 801 841 550	4 116 715 363 800	84 859 704 298 201	34.2232
27	247 959 266 474 052	63 205 303 218 876	16 123 801 841 550	332 818 970 772 253	34.1889
28	973 469 712 824 056	247 959 266 474 052	63 205 303 218 876	1 306 288 683 596 310	34.1572
29	3 824 345 300 380 220	973 469 712 824 056	247 959 266 474 052	5 130 633 983 976 530	34.1277
30	15 033 633 249 770 500	3 824 345 300 380 220	973 469 712 824 056	20 164 267 233 747 100	34.1003
31	59 132 290 782 430 700	15 033 633 249 770 500	3 824 345 300 380 220	79 296 558 016 177 800	34.0747
32	232 714 176 627 630 000	59 132 290 782 430 700	15 033 633 249 770 500	312 010 734 643 808 000	34.0507

1. táblázat. Csapatok n száma, az $n-1$ és $n-2$, valamint $n-3$ elemmel kezdődő, valamint összesen kiszűrt sorozatok $F(n)$ száma $n = 1, \ldots, 32$ csapat esetén.

n	$E_z(n)$	$\frac{T_{\mathrm{EGL}}(n)}{E_z(n)}$	$\frac{T_{\mathrm{EGLJ}}(n)}{E_z(n)}$	$\frac{T_{\mathrm{HHL}}(n)}{E_z(n)}$
10	21 942	0.683620	0.000000	0.000000
11	83 980	0.369136	0.190521	0.381083
12	323 554	0.336883	0.194712	0.287433
13	1 248 072	0.299662	0.213128	0.237967
14	4 829 708	0.319895	0.226101	0.222788
15	18 721 080	0.338281	0.241371	0.226643
16	72 714 555	0.348197	0.251665	0.233406
17	282 861 360	0.379355	0.255846	0.240789
18	1 101 992 870	0.377512	0.267014	0.249460
19	4 298 748 300	0.394.319	0.281491	0.261416

2. táblázat. Nullamentes sorozatok száma, valamint az átlagos futási idő mikroszekundumban az EGL, EGLJ, valamint HHL esetén.

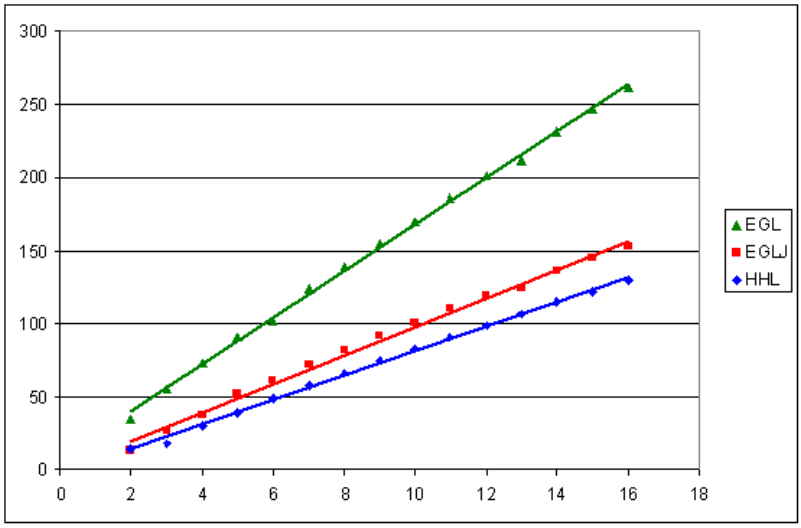


1. ábra. Az EGL, EGLJ és a HHL átlagos futási ideje.

A HHL-nek 14 műveletre van szüksége: 1 összehasonlításra az 1. sorban, 1 összehasonlításra a 3. sorban, 1 értékadásra az 5. sorban, 5 műveletre a 6. és 7. sorban (1 $i = 1$ értékadás, 1 összeadás az i növelésénél, 2 összehasonlítás $i < n$-re, 1 $H_1 = s_1$ értékadás), 1 maradékképzésre, 1 összehasonlításra a 8. sorban, 1

n	$G_z(n)$	$\frac{O_{\mathrm{EGL}}(n)}{E_z(n)}$	$\frac{O_{\mathrm{EGLJ}}(n)}{E_z(n)}$	$\frac{O_{\mathrm{HHL}}(n)}{E_z(n)}$
2	1	35.000	13.000	14.000
3	2	55.000	26.500	18.000
4	7	73.000	37.667	29.889
5	20	91.000	51.429	39.357
6	71	101.609	61.473	48.591
7	240	123.495	72.480	57.553
8	871	139.162	82.042	66.123
9	3 148	154.944	91.751	74.552
10	11 655	170.421	100.929	82.749
11	43 332	185.885	110.047	90.824
12	162 769	201.209	118.930	98.758
13	614 718	212.177	124.720	106.591
14	2 330 537	231.659	136.373	114.739
15	8 875 768	246.785	144.939	121.976
16	33 924 858	261.846	153.411	129.552

3. táblázat. Csúcsok n nullamentes gráfos sorozatok $G_z(n)$ száma, valamint az egy nullamentes páros sorozatra jutó átlagos műveletszám az EGL, EGLJ és a HHL-algoritmus és $2, \ldots, 16$ csúcs esetén.



2. ábra. Az EGL, EGLJ és a HHL átlagos műveletszáma nullamentes páros sorozatok tesztelése esetén.

értékadásra a 10. sorban, 2 kivonásra és egy értékadásra a 13. sorban, valamint 1 összehasonlításra a 14–22. sorban.

Az EGLJ-nek 13 műveletre van szüksége: 1 értékadásra az 1. sorban, 5 műveletre a 2–3. sorban (ciklusváltozó kezdeti értékének beállítása és növelése, 1 összehasonlítás, két H_i értékadás), 1 maradékképzésre és 1 összehasonlításra az 5–8. sorban, 1 értékadásra a 9. sorban, 4 műveletre a 10–28. sorban (ciklusváltozó kezdeti értékének beállítása és növelése, 1 összehasonlítás a 11. sorban és 1 összehasonlítás a 17. sorban).

Az EGL-nek 35 műveletre van szüksége: 1 értékadásra az 1. sorban, 9 műveletre a 2–3. sorban (ciklusváltozó kezdeti értékének beállítása és növelése kétszer, összehasonlítása kétszer, 2 összeadás a H_i esetében, 2 értékadás a H_i esetében) 1 maradékképzésre, 1 összehasonlításra a 4. sorban, 1 értékadásra a 7. sorban, 7 műveletre a 8–12. sorban (ciklusváltozó kezdeti értékének beállítása és növelése kétszer, összehasonlítása kétszer, elágazás tesztelése kétszer), 4 műveletre a 13–14. sorban (ciklusváltozó kezdeti i értékének beállítása és csökkentése, 1 összehasonlítás, 1 értékadás), 11 műveletre a 15–23. sorban (ciklusváltozó kezdeti értékének beállítása és növelése, 9 összehasonlítás).

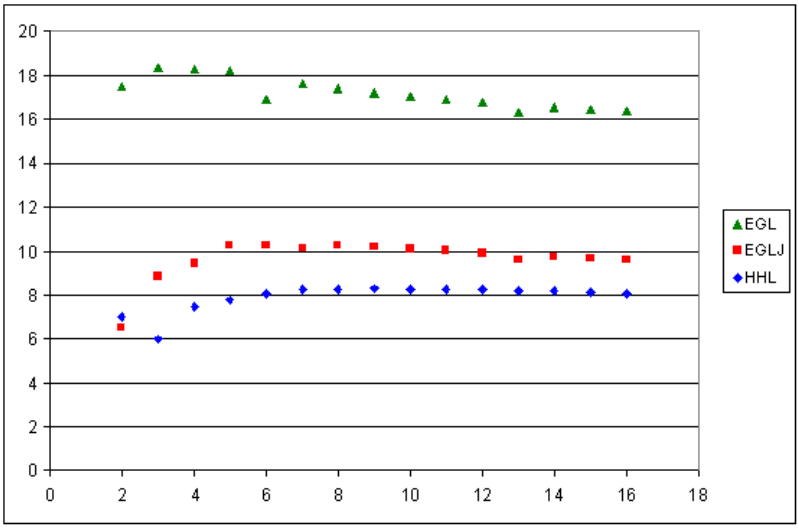
A 4. táblázatban található az amortizált műveletek számának és a nullamentes páros sorozatok $E_z(n)$ számának hányadosa az EGL ($O_{\mathrm{EGL}}(n)/E_z(n)$), az EGLJ ($O_{\mathrm{EGLJ}}(n)/E_z(n)$) és a HHL ($O_{\mathrm{HHL}}(n)/E_z(n)$) algoritmusok esetén, ahol $n = 2, \ldots, 16$.

n	$\frac{O_{\mathrm{EGL}}(n)}{E_z(n)}$	$\frac{O_{\mathrm{EGLJ}}(n)}{E_z(n)}$	$\frac{O_{\mathrm{HHL}}(n)}{E_z(n)}$
2	17.500	6.500	7.000
3	18.333	8.833	6.000
4	18.250	9.417	7.472
5	18.200	10.286	7.781
6	16.935	10.246	8.099
7	17.642	10.154	8.222
8	17.395	10.255	8.265
9	17.216	10.195	8.284
10	17.042	10.093	8.275
11	16.899	10.004	8.257
12	16.767	9.911	8.230
13	16.321	9.593	8.199
14	16.547	9.741	8.196
15	16.452	9.663	8.132
16	16.365	9.588	8.097

4. táblázat. Az amortizált műveletek száma a nullamentes sorozatokra nézve az EGL, EGLJ, valamint HHL-algoritmusokra $n = 2, \ldots, 16$ csúcs esetében.

A 3. ábrán láthatjuk az EGL, EGLJ és HHL amortizált műveletszámát. A (zöld) háromszögek mutatják az $(n, O(n))$ párosokat a lineáris Erdős–Gallai-algoritmus esetében (EGL), a (piros) négyzetek a lineáris ugró Erdős–Gallai-algoritmus esetében, míg a (kék) gyémántok a lineáris Havel–Hakimi-algoritmust jelzik.

A 4. táblázat és a 3. ábra alapján kijelenthető, hogy az EGL, EGLJ és HHL-algoritmusok CAT (konstans időben amortizált) algoritmusok (lásd [161]).



3. ábra. Az amortizált műveletek száma a nullamentes sorozatokra nézve az EGL, EGLJ, valamint HHL esetében.

3. Leszámláló Erdős–Gallai-algoritmus (EGE)

A gráfelmélet egyik klasszikus problémája különböző gráfok – többek között az egyszerű gráfok – foksorozatainak leszámlálása. Példaként tekinthető a *The On-Line Encyclopedia of Integer Sequences* [170], amely $n = 1, \ldots, 31$ csúcsra tartalmazza az egyszerű gráfok foksorozatainak számát (az $n = 20, \ldots, 23$ értékeket 2011. júliusában adta meg Nathann Cohen, míg az $n = 24, \ldots, 29$ értékek a 2011. november 15-én elért eredményeink [95]).

Az új, gyorsított EGL-algoritmust alkalmaztuk a nagyobb értékű n-ek meghatározására.

A szabályos sorozatok ellenőrzését és a gráfos sorozatok leszámlálását tűztük ki célul. Az n-szabályos sorozatok $R(n)$ száma

$$R(n) = \binom{2n-1}{n}, \tag{2}$$

a nullamentes sorozatok $R_z(n)$ száma pedig

$$R_z(n) = \binom{2n-2}{n}. \tag{3}$$

A (2) és a (3) képleteket egyszerűen megkaphatjuk az ismétléses kombinációkra vonatkozó képlettel [19], vagy egyszerű közvetlen bizonyítással [95].

n	$R(n)$	$E(n)$
39	13608507434599516007800	6804253717317430635800
40	53753604366668088230810	2687680218336850574761 0
41	21239229042439586081442 0	1061961452122668536716 20
42	83945524310594554512366 0	4197276215531073370304 40
43	33187765425118777365354 00	1659388271256207997204920
44	13124252690842425594480900	6562126345421738821981380
45	51913710643776705684835560	2595685532188940489189964 0
46	20539772472102957466608852 0	1026988623605168456907261 60
47	81285057017258512527430776 0	4064252850862966793525176 80
48	32175335069331494542108015 50	1608766753466582789006321 550
49	12738806129490428451365214 300	6369403064745230349484448 700
50	50445672272782096667406248 628	2522283613639107993635473 3752
51	19980442743337222601600122 0056	9990221371668617621330382 8904
52	79153292406297458767877406 4068	3957664620314874178190202 69060
53	31362625293061257247649538 3760	1568131264653063110341743 393432
54	12428892245768720464809261 509160	6214446122884360719139487 166608
55	49263609265046928387789436 527216	2463180463252346516736443 1087664
56	19529502244357889468016526 6232892	9764751122178944925225528 3306556
57	77432763284647070522311140 6467256	3871638164232353564359010 03613848
58	30706095785291079689882004 04956360	1535304789264553992010916 827363440
59	12178349853827309571919303 301013360	6089174926913654800993284 900277200
60	48307454420181661301946569 760686328	2415372721009083068053943 0271558520

5. táblázat. A szabályos és páros sorozatok száma $n = 39, \ldots, 60$ esetén.

Ascher 1987-ben a következő explicit formula felhasználásával határozta meg a páros sorozatok $E(n)$ számát (pontosabban a mu torere nevű maori játék bizonyos állapotainak számát):

3.1. Lemma. (Ascher [5], Sloane, Pfoffe [172]) *Ha $n \geq 1$, akkor a páros sorozatok $E(n)$ száma*

$$E(n) = \frac{1}{2}\left(\binom{2n-1}{n} + \binom{n-1}{\lfloor n/2 \rfloor}\right). \tag{4}$$

Bizonyítás. Lásd [5]. □

A (2) és (4) képletek felhasználásával meghatároztuk az $R(n)$ és $E(n)$ értékeket $n = 1, \ldots, 100$ esetén. Az $n = 1, \ldots, 38$ eredményeket [95]-ben publikáltuk, az $n = 39, \ldots, 60$ értékek a 5. táblázatban találhatóak, míg a további értékek $n = 100$-ig az OEIS-ben, valamint az azokat előállító program [127]-ben található.

A következő lemma miatt elegendő a nullamentes páros sorozatok ellenőrzése.

3.2. Lemma. (Iványi, Lucz, Móri, Sótér [95]) *Ha $n \geq 2$, akkor az n-gráfos sorozatok $G(n)$ száma meghatározható az $(n-1)$-gráfos sorozatok $G(n-1)$ és az*

n-gráfos nullamentes sorrozatok $G_z(n)$ számából:

$$G(n) = G(n-1) + G_z(n),$$

és ha $n \geq 1$, akkor

$$G(n) = 1 + \sum_{i=2}^{n} G_z(i).$$

Bizonyítás. Lásd [95]. □

A [95] cikkben található 12. és 13. lemmából azt kapjuk, hogy $R(n) = \Theta(4^n/\sqrt{n})$ és $E(n) = \Theta(4^n/\sqrt{n})$. A 14. következmény [95] azt állítja, hogy ha n tart a végtelenhez, akkor $E(n)/R(n)$ 1/2-hez tart. Szimulációs kísérleteink alapján úgy gondoljuk, hogy a következő állítás szintén igaz: $E_z(n) = \Theta(4^n/\sqrt{n})$, és ha n tart a végtelenhez, akkor $E_z(n)/R(n)$ 1/4-hez tart.

A [42, 95] cikkekben szereplő 22. tétel szerint azonban

$$\frac{4^n}{cn} < G(n) < \frac{4^n}{(\log n)^C \sqrt{n}},$$

ahol c és C pozitív konstansok.

A 2., a 3. és a 4. táblázatban lévő adatok szerint ha n nő, akkor a lineáris EGL, EGLJ és HHL-algoritmusok átlagos költsége csökken. A gráfos sorozatok ellenőrzésének ideje hosszabb, ezért ha n tart a végtelenhez, akkor a nemgráfos sorozatok ellenőrzésének átlagos ideje és a $G(n)/E_z(n)$ sorozat nullához tart.

Ezeket az eredményeket figyelembe véve azt kapjuk, hogy elegendő számunkra a szabályos sorozatok negyedének ellenőrzése. A 6. táblázatban található a nullamentes gráfos sorozatok száma, valamint a szabályos sorozatok számával elosztott nullamentes, szabályos sorozatok számával elosztott nullamentes gráfos és a szabályos sorozatok számával elosztott gráfos sorozatok száma. Burns eredményéből [42, 95] következik, hogy az utolsó két oszlopban található sorozatok nullához tartanak.

Az EGE párhuzamosított EGP változatát (lásd a következő fejezetben) felhasználva meghatároztuk $G(n)$ értékeit $n = 29$-ig. Ezen értékek megtalálhatóak a [95]-ben lévő 2. táblázatban.

Megjegyezzük, hogy $G_z(n)$ megadja az izolált csúcsokat nem tartalmazó egyszerű gráfok számát. 2006-ban Gordon Royle [159] vetette fel a következő kérdést: igaz-e, hogy ha n tart a végtelenhez, akkor $G_z(n+1)/G_z(n)$ 4-hez tart?

Hasonló kérdés vonatkozik a $G(n)/G(n+1)$ sorozat határértékére. Mindkét kérdést szimulációval vizsgáltuk a [90, Conjecture 12, Conjecture 13, Table 4] cikkben.

n	$G_z(n)$	$E_z(n)/R(n)$	$G_z(n)/R(n)$	$G(n)/R(n)$
1	0	0.000000	0.000000	1.000000
2	1	0.333333	0.333333	0.666667
3	2	0.200000	0.200000	0.400000
4	7	0.257143	0.200000	0.314286
5	20	0.222222	0.158730	0.246032
6	71	0.238095	0.153680	0.220779
7	240	0.230769	0.139860	0.199301
8	871	0.236053	0.135454	0.188500
9	3148	0.235294	0.129494	0.179391
10	11655	0.237524	0.126166	0.173375
11	43332	0.238095	0.122852	0.168260
12	162769	0.239188	0.120384	0.164278
13	614198	0.245769	0.118108	0.160821
14	2330537	0.240783	0.116188	0.157882
15	8875768	0.241379	0.114439	0.155271
16	33924859	0.241946	0.112880	0.152950
17	130038230	0.242424	0.111448	0.150844
18	499753855	0.242860	0.101137	0.148926
19	1924912894	0.243243	0.108920	0.147158
20	7429160296	0.243590	0.107789	0.145521
21	28 723 877 732	0.2439024	0.106729	0.143997
22	111 236 423 288		0.105733	0.142569
23	431 403 470 222		0.104793	0.141228
24	1 675 316 535 350		0.103903	0.139961
25	6 513 837; 679 610		0.103058	0.138762
26	25 354 842 100 894		0.102254	0.137625
27	98 794 053 269 694		0.101486	0.136542
28	385 312 558 571 890		0.100752	0.135509
29	1 504 105 116 253 904		0.100049	0.134521
30	5 876 236 938 019 300		0.100752	0.135509
31	22 974 847 474 172 374		0.100049	0.134521

6. táblázat. A csúcsok n és a nullamentes gráfos sorozatok $G_z(n)$ száma, illetve a nullamentes páros sorozatok $E_z(n)$ számának, a nullamentes gráfos sorozatok $G_z(n)$ számának, valamint a gráfos sorozatok $G(n)$ számának és a szabályos sorozatok $R(n)$ számának hányadosa $n = 1, \ldots, 31$ csúcs esetén.

Megmutattuk, hogy a nullamentes gráfos sorozatok száma $n = 30$ csúcs esetén

$$G_z(30) = 5\,876\,236\,938\,019\,300,$$

míg $n = 31$ csúcs esetén

$$G_z(31) = 22\,974\,847\,474\,172\,374.$$

Tripathi és Vijay [95, 182, 6. lemma és 7. tétel] eredményeit felhasználva lényegesen csökkenthetjük a nullamentes páros sorozatok átlagos ellenőrzési idejét. Tripathi és Vijay [182] eredményével kapcsolatban beláttuk [95], hogy a tesztelt sorozatok egyenletes eloszlása esetén az ellenőrző pontok számának várható értéke körülbelül $n/2$.

A 3.3. lemma felhasználásával tovább gyorsíthatjuk az EGE-algoritmust. Ha $b = (b_1, \ldots, b_n)$ egy szabályos sorozat, akkor $c = (c_1, \ldots, c_n)$ *lexikografikusan i-kisebb, mint* b, ha

$$c_j = b_j \quad \text{for} \quad j = 1, \ldots, i,$$

és

$$\sum_{j=i+1}^{n} c_j < \sum_{j=i+1}^{n} b_j.$$

3.3. Lemma. *Legyen i egész szám, melyre $1 \le i \le n$. Ha $b = (b_1, \ldots, b_n)$ egy nemgráfos sorozat és $c = (c_1, \ldots, c_n)$ lexikografikusan i-kisebb, mint b, akkor c szintén nemgráfos.*

Bizonyítás. Lásd [90]. □

A következőekben bemutatandó Erdős–Gallai-Enumerating (EGE) az EGL egy leszámláló változata. Az algoritmus lexikografikus sorrendben megvizsgálja a nullamentes páros sorozatokat, ami lehetővé teszi a legtöbb alapvető művelet végrehajtását $O(1)$ átlagos időben.

– H_i (összegzett fokszámok): a legtöbb esetben a b sorozat utolsó eleme az egyetlen, amely változik, így ilyenkor elegendő a H utolsó értékét frissíteni a b utolsó értékének változása szerint.

– C_i (ellenőrző pontok): ha módosítjuk a sorozat i. elemét, akkor az előtte lévő pontok ugyanazok maradnak, tehát az összes ellenőrző pont előtt ugyanazok maradnak, így elegendő csak az i. index előtti elemet frissíteni és az összes többit utána.

– W_i (súlypontok): minden alkalommal, amikor az ellenőrző algoritmus megkap egy sorozatot ellenőrzésre, frissíti a súlypontokat, azonban sosem kezd 1-ről vagy n-ről. Azt az utolsó értéket használja, amelyet akkor használt, amikor a sorozat aktuális indexét ellenőrizte. Különböző súlypontokkal rendelkezik minden pontra minden egyes i index esetén, és csak eltolja ezeket az értékeket balra, vagy jobbra.

Tegyük fel, hogy n, b, H, c, C és W globális változók, így a **return** utasítás nem igényel többletidőt.

Az EGE fontos tulajdonsága, hogy a következő problémákat $\Theta(1)$ átlagos időben megoldja:

– egy nullamentes sorozat generálása;

– egy sorozathoz tartozó H összegzett fokszámok frissítése;

– egy sorozathoz tartozó C ellenőrző pontok frissítése;

– egy sorozathoz tartozó W súlypontok frissítése.

Noha az EGE a részproblémák többségét sorozatonként átlagosan $\Theta(1)$ időben megoldja, az ellenőrző pontoknál lévő munka több időt igényel, ezért a teljes futási idő $\Theta(E(n))$.

Az Erdős–Gallai-Enumerating program [95] 8. tételén alapul.

3.1. Algoritmus. Erdős–Gallai-Enumerating (n, G_z)

Bemenet. n: csúcsok száma ($n \geq 4$);
$b = b_1, \ldots, b_n$: n-szabályos sorozat.
Kimenet. G_z: az n hosszú nullamentes gráfos sorozatok száma.
Munkaváltozók. i és j: ciklusváltozók;
$H = H_1, \ldots, H_n$: H_i a tesztelt b sorozat első i elemének össze;
$W = W_1, \ldots, W_n$: W_i az aktuális b_i-hez tartozó súlypont, b olyan elemeinek maximális indexe, amelyre igaz, hogy nem kisebbek, mint i;
y: az aktuális b_i vágópontja, azaz i és w maximuma.

1. **for** $i = 1$ **to** n // 1–8. sor: kezdeti értékek beállítása
2. $b_i = n - 1$
3. $H_i = i(n-1)$
4. $W_i = n$
5. $C_i = 0$
6. $G_z = 1$
7. $c = 0$
8. $b_{n+1} = -1$
9. **while** $b_2 \geq 2$ **or** $b_1 \geq 3$ // 9. sor: utolsó sorozat volt?
10. **if** $b_n \geq 3$ // 10–14. sor: a következő sorozat előállítása
11. New3(n, b, H, c, C, W)
12. **else if** $b_n = 2$
13. New2(n, b, H, c, C, W)
14. **else** New1(n, b, H, c, C, W)

15. CHECK(n, b, H, c, W, L) // 15. sor: paraméterek frissítése
16. $G_z = G_z + L$ // 16. sor: G_z növelése
17. **return** G_z // 17. sor: végeredmény

Az algoritmus négy eljárást használ. NEW1, NEW2, valamint NEW3 egy új sorozatot generálnak (ahol b_n 1, 2, illetve 3) és frissítik a fő paramétereket, míg a CHECK eldönti, hogy az aktuálisan megvizsgált sorozat gráfos-e vagy sem.

A CHECK eljárásban az EGLJ feltételét [125, (26) egyenlőség] használjuk fel.

CHECK(n, b, H, c, C, W)

1. **for** $i = 1$ **to** c // 1–4. sor: ellenőrzőpontok tesztelése
2. $y = \max(W_{C_i}, i)$ // 2. sor: aktuális vágópont kiszámítása
3. **if** $H_i > i(y-1) + H_n - H_y$ // 3–4. sor: EG tesztelés
4. **return** 0
5. **return** 1 // 6. sor: b gráfos

NEW3(n, b, H, c, C, W)

1. $b_n = b_n - 2$ // 1–10. sor: új sorozat előállítása, ha $b_n = 3$
2. $H_n = H_n - 2$
3. **if** $b_n = b_{n-1} - 2$
4. $c = c + 1$
5. $C_c = n - 1$
6. $W_{b_n} = W_{b_n} - 1$
7. **if** $b_n \leq b_{n-1}$
8. $W_{b_n+1} = n + 1$
9. $W_{b_n} = n + 1$
10. **return** H, c

NEW2(n, b, H)

1. **if** $b_{n-1} = 2$ // 1–53. sor: új sorozat előállítása, ha $b_n = 2$
2. $b_n = 1$ // 1–9. sor: új sorozat előállítása, ha $b_{n-1} = 2$
3. $b_{n-1} = 1$
4. $H_{n-1} = H_{n-1} - 1$
5. $H_n = H_n - 2$
6. $W_2 = n - 2$
7. **if** $b_{n-2} = 2$ // 7–9. sor: új sorozat előállítása, ha $b_{n-2} = 2$
8. $c = c + 1$
9. $C_c = n - 1$

10. **else if** $b_{n-1} = 3$ // 10–16. sor: új sorozat előállítása, ha $b_{n-1} = 3$
11. $b_{n-1} = 2$
12. $b_n = 1$
13. $H_{n-1} = H_{n-1}$
14. $H_n = H_n - 2$
15. $W_3 = n - 2$
16. $W_2 = n - 1$
17. **else** $H_{n-1} = H_{n-1} - 1$
18. **if** $b_{n-2} = b_{n-1}$ **and** b_n páratlan
19. $b_{n-1} = b_{n-1} - 1$
20. $b_n = b_{n-1}$
21. $H_n = H_n + b_{n-1} - b_n - 1$
22. $C_c = C_c - 1$
23. $W_{b_{n-2}} = n - 2$
24. **for** $i = 1$ **to** b_{n-2}
25. $W_i = n$
26. **if** $b_{n-2} = b_{n-1}$ **and** $b_n - 1$ páros
27. $b_{n-1} = b_{n-1} - 1$
28. $b_n = b_{n-1} - 1$
29. $H_n = H_n + b_{n-1} - b_n - 1$
30. $C_c = C_c - 1$
31. $c = c + 1$
32. $C_c = n - 1$
33. $W_{b_{n-2}} = n - 2$
34. $W_{b_{n-1}} = n - 1$
35. **for** $i = 1$ **to** $b_{n-2} - 2$
36. $W_i = n$
37. **if** $b_{n-2} > b_{n-1}$ **and** b_{n-1} páros
38. $b_{n-1} = b_{n-1} - 1$
39. $b_n = b_{n-1}$
40. $H_n = H_n + b_{n-1} - b_n - 1$
41. $c = c - 1$
42. $W_{b_{n-2}-1} = n - 2$
43. $W_{b_{n-2}-1} = n - 1$
44. **for** $i = 1$ **to** $b_{n-1} - 1$
45. $W_i = n$
46. **if** $b_{n-2} > b_{n-1}$ **and** $b_n - 1$ páros
47. $b_{n-1} = b_{n-1} - 1$

48. $b_n = b_{n-1} - 1$
49. $H_n = H_n + b_{n-1} - b_n - 1$
50. $W_{b_{n-1}+1} = n - 1$
51. **for** $i = 1$ **to** $b_{n-1} - 1$
52. $W_i = n$
53. **return** H, c, C, W

New1 hasonló New2-höz (bár bonyolultabb, lásd Generate-New-Sequence a következő részben), ezért azt itt nem részletezzük.

4. A feladat részekre osztása (szeletelés a szabályos sorozatok száma alapján)

A leszámlálás párhuzamos megoldásának fontos része a feladat kisebb részekre osztása. Ezt *szeletelésnek* nevezzük.

Felosztási módszereink alapja a következő *szabályos* (l, u, m)*-lemma.*

Legyenek l, u és m egész számok, továbbá m legyen pozitív. Ekkor az

$$l \le a_1 \le \cdots \le a_m \le u$$

feltételnek eleget tevő $\mathbf{a} = a_1, \dots a_m$ sorozatokat *nemcsökkenő* (l, u, m)*-szabályos*, míg az

$$l \le a_m \le \cdots \le a_1 \le u$$

feltételnek eleget tevő sorozatokat *nemnövekvő* (l, u, m)*-szabályos* sorozatoknak nevezzük.

4.1. Lemma. (Iványi, Lucz, Móri, Sótér [95, (21) és (22)]) *Legyenek l, u és m egész számok, továbbá m legyen pozitív. Ekkor az*

$$l \le a_1 \le \cdots \le a_m \le u$$

feltételnek eleget tevő (l, u, m)-szabályos *sorozatok száma*

$$R(l, u, m) = \binom{u - l + m}{m}. \tag{5}$$

Bizonyítás. Jelöljük $\mathbf{b}$-vel a $b_i = a_i + i - 1$ előírással képzett $b_1, \dots, b_m$ sorozatot. A $\mathbf{b}$ sorozatok száma megegyezik az $\mathbf{a} = a_1, \dots, a_n$ sorozatok számával. Másrészt a $\mathbf{b}$ sorozatok száma annyi, ahány módon $2m-1$ különböző elem közül kiválaszthatunk m elemet, azaz az (5) egyenlőségben szereplő binomiális együttható valóban megadja a keresett számosságot. □

1	1	1	1
1	2	3	4
1	3	6	10
1	4	10	20
1	5	15	35

7. táblázat. A felosztáshoz használt mátrix $n = 5$ esetén.

2	2	2	2	2
3	2	2	2	2
3	3	3	3	3
4	2	2	2	2
4	3	3	3	3
4	4	3	3	3
4	4	4	4	4

8. táblázat. $n = 5$-ös felosztás határsorozatai.

Az (5) képlet segítségével kiszámítható, hány olyan sorozat van, amelynek hossza, valamint első és utolsó eleme adott. Ezt felhasználva megadható egy mátrix, amely megadja, hány olyan sorozat van, amely az f értékkel kezdődik és a g értékkel végződik. Ezt a táblázatot felhasználva a következő módon adható meg a felosztás:

1. válasszuk meg a maximális szeletméretet (ezt a konkrét számítások során úgy választottuk meg, hogy esetünkben ez akkora szeleteket jelentett, amelyek egy éjszaka alatt kiszámíthatóak);

2. induljunk el a mátrix alsó sorának első elemétől, és kezdjük a mátrix további sorait kiolvasni és összegezni mindaddig, amíg az aktuális kapacitás vagy a kiolvasandó értékek el nem fogytak;

3. ha egy érték túl nagy az adott maximális mérethez viszonyítva, akkor lépjünk egy sorral feljebb, és kezdjük el azt a sort kiolvasni addig az oszlopig, ahonnan felléptünk;

4. folytassuk ezt az algoritmust, amíg az utolsó sor végére nem értünk.

A korábban ismertetett algoritmus szerint $n = 5$-re a 7. táblázatból kiolvasható a 8. táblázatban látható felosztás. A táblázatban látható sorozatok a következő

módon értendőek: az első szelet az 1; 1; 1; 1; 1 sorozattól a 2; 2; 2; 1; 1 sorozatig tart, a második pedig a 2; 2; 2; 2; 2 sorozattól a 3; 2; 2; 1; 1 sorozatig és így tovább. Ez a módszer ugyan nem garantálja, hogy a részfeladatok hossza pontosan ugyanakkora legyen, azonban megszünteti a korábbiakban tapasztalt „mamut" hatást.

Ezt a módszert használtuk fel $G_z(29)$ kiszámításakor. A teljes számítást összesen több mint 15 000 részfeladatra felosztva végeztük. A teljes felosztás generálásának első lépése a 3. táblázathoz hasonló mátrix előállítása a megfelelő n értékhez. A mátrix feltöltését végzi el az alábbi GENERATE-MATRIX program.

4.1. Algoritmus. GENERATE-MATRIX $(n, MaxSize)$

Bemenet. n: a csúcsok száma ($n \geq 4$);
MaxSize: a maximális megengedett szeletméret
(nullmentes szabályos sorozatok maximális száma a szeletben).
Kimenet. A képernyőre írjuk a szeleteket határoló sorozatokat.

1. **for** $j = n - 1$ **downto** 1 // 1–2. sor: a mátrix első sorának kitöltése
2. $M_{1j} = 1$
3. **for** $i = n$ **downto** 2 // 3–5. sor: a mátrix feltöltése
4. **for** $j = 1$ n
5. $M_{i,j} = \binom{i+j-2}{i-1}$
6. GENSEQUENCES$(M, n, n, 1, n-1, MaxSize, 0)$ // 6. sor: szelet generálása

Miután megvan a mátrixunk, már csak ki kell olvasnunk belőle a szeletek határsorozatait (a 8. táblázathoz hasonló alakban) az alábbi GENERATE-SEQUENCES algoritmus segítségével.

4.2. Algoritmus. GENERATE-SEQUENCES$(M, n, i, j, j_m, MaxSize, J)$

Bemenet. M: a felosztáshoz tartozó mátrix, amit a GENERATE-MATRIX algoritmussal kaptunk;
n: a csúcsok száma ($n \geq 1$);
i, j, J: segédparaméterek;
MaxSize: a maximális megengedett szeletméret (nullmentes szabályos sorozatok maximális száma a szeletben).
Kimenet. A szeleteket határoló sorozatok (rekurzívan).

1. $C = 0$ // 1. sor: a szelet kezdő méretének beállítása
2. **while** $j < jm + 1$
3. **if** $C + M_{ij} \leq MaxSize$ // 3. sor: ha bővíthető a szelet
4. $C = C + M_{i,j}$ // 4. sor: bővítés
5. **if** $j \leq j_m$ // 5–6. sor: tovább lépünk a mátrixban

```
6.              j = j + 1
7.      else                                   // 7. sor: nem bővíthető a szelet
8.          if C ≠ 0                           // 8. sor: a szelet nem üres
9.              for k = 2 to size(J, 2)        // 9–14. sor: kiírás
10.                 print J_k
11.             for k = 2 to size(J, 2)
12.                 print j − 1
13.                 print newline              // 13. sor: sortörés
14.                 C = 0
15.         if M_{i,j} > MaxSize és j ≤ j_m // 15. sor: felbonthatóság ellenőrzése
16.             Generate-Sequences(M, n, i − 1, 1, j, MaxSize, [J, j])
17.             j = j + 1
18. if C ≠ 0                                   // 18. sor: utolsó szelet nem üres
19.     for k = 2 to size(J, 2)                // 19–23. sor: utolsó szelet kiírása
20.         print(J_k)
21.     for k = 1 to n − size(J, 2) − 1
22.         print(J, size(J, 2))
23.     print newline
```

5. A feladat részekre osztása (szeletelés a páros sorozatok száma alapján)

Az $n \geq 32$ esetben a következő tételen – melyet *páros* (l, u, m)*-tételnek* (röviden: páros tételnek) nevezünk – alapuló felosztási módszert alkalmaztunk.

Legyenek l, u és m egész számok, melyekre $u \geq l$ és $m \geq 1$. Legyen $\mathbf{a} = a_1, \ldots, a_m$ egész számok olyan sorozata, amelyre teljesül

$$u \geq a_m \geq \cdots \geq a_1 \geq l \tag{6}$$

és

$$a_1 + \cdots + a_m \quad \text{páros.} \tag{7}$$

Jelöljük $E(l, u, m)$-mel a (6) és (7) feltételeknek eleget tevő sorozatok számát.

Ascher tételének [5] következő általánosítása megadja $E(l, u, m)$-et.

5.1. Tétel. *Ha l, u és m olyan egész számok, melyekre $u \geq l$ és $m \geq 1$, továbbá $p = 1$, ha $l(u - l + 1)$ páratlan, és $p = 0$ egyébként, akkor*

$$E(l, u, m) = \sum_{i=0}^{\lfloor m/2 \rfloor + p} \binom{(s + 2i}{2i} \cdot \binom{s + m - 2i}{m - 2i}, \tag{8}$$

ahol $(s = \lfloor u - l + 1)/2 \rfloor + p$.

Bizonyítás. a) Először tegyük fel, hogy $u-l+1$ páros. Ekkor $p=0$ és az $[u,l]$ intervallum $(u-l+1)/2$ páratlan és $(u-l+1)/2$ páros számot tartalmaz. A páratlan számok közül páros sokat – azaz $0,2,\ldots,(u-l+1)/2$ elemet kell kiválasztanunk. A még hiányzó elemeket pedig a páros elemek közül kell kiválasztanunk.

Mivel n elem k-ad osztályú ismétléses variációinak száma $\binom{n+k-1}{k}$, ezért ebben az esetben az

$$E(l,u,m)=\sum_{i=0}^{(\lfloor m/2\rfloor}\binom{(u-l+1)/2+2i}{2i}\cdot\binom{(u-l+1)/2+m-2i}{m-2i}$$

eredményt kapjuk, ami a $p=0$ esetben megfelel (8)-nak.

b) Most tegyük fel, hogy mind l, mind $u-l+1$ páratlan. Ekkor $p=1$. Ekkor az $[l,u]$ intervallumban páratlan számú páratlan elem van, amelyek közül most is páros számút kell kiválasztani, ezért ebben az esetben az

$$E(l,u,m)=\sum_{i=0}^{\lfloor m/2\rfloor+1}\binom{(u-l+1)/2+p+2i)}{2i}\cdot\binom{(u-l+1)/2+p+m-2i}{m-2i}$$

eredményt kapjuk, ami a $p=1$ esetben megfelel (8)-nak. □

A 5.1. tételnek mind a páros sorozatok $E(n)$ számát megadó páros lemma [90, Lemma 6], mind pedig a nullamentes páros sorozatok $E_z(n)$ számát megadó nullamentes páros lemma [90, Lemma 3] speciális esete. Ehhez érdemes hozzátenni, hogy a következményekben szereplő képletek egyszerűbbek, mint a korábban publikált lemmákban lévőek: négy képlet helyett csak egyet kell használni. Ez annak köszönhető, hogy most közvetlenül a foksorozatokat vizsgáltuk a monotonitást biztosító transzformáció nélkül.

5.1. Következmény. *Ha $n\geq 1$, akkor*

$$E(n)=\sum_{i=0}^{\lfloor n/2\rfloor}\binom{\lfloor n/2\rfloor-1+2i}{2i}\cdot\binom{\lceil n/2\rceil-1+n-2i}{n-2i}. \tag{9}$$

Bizonyítás. A 5.1. tételben végezzük el az $l=0$, $u=n-1$, $p=0$ és $m=n$ helyettesítést. □

5.2. Következmény. *Ha $n\geq 1$, akkor*

$$E_z(n)=\sum_{i=0}^{\lfloor n/2\rfloor}\binom{\lfloor (n-1)/2\rfloor-1+2i}{2i}\cdot\binom{\lceil (n-1)/2\rceil-1+n-2i}{n-2i}. \tag{10}$$

Bizonyítás. A 5.1. tételben végezzük el az $l=1$, $u=n-1$ és $m=n$ helyettesítést. □

Tekintsünk néhány példát.

1. Először számítsuk ki $E(4)$-et. (9) szerint

$$E(4) = \binom{1}{0} \cdot \binom{5}{4} + \binom{3}{2} \cdot \binom{3}{2} + \binom{5}{4} \cdot \binom{1}{0} = 1 \cdot 5 + 3 \cdot 3 + 5 \cdot 1 = 19.$$

2. Számítsuk ki $E_z(4)$-et. (10) szerint

$$E_z(4) = \binom{4}{0} \cdot \binom{4}{4} + \binom{3}{2} \cdot \binom{2}{2} + \binom{5}{4} \cdot \binom{0}{0} = 1 \cdot 1 + 3 \cdot 1 + 5 \cdot 1 = 9.$$

3. Számítsuk ki $E(5)$-öt (9) szerint

$$E(5) = \binom{1}{0} \cdot \binom{7}{2} + \binom{3}{2} \cdot \binom{5}{2} + \binom{5}{4} \cdot \binom{3}{1} = 1 \cdot 21 + 3 \cdot 10 + 5 \cdot 3 = 66.$$

4. Számítsuk ki $E(6)$-ot.

$$E(6) = \binom{2}{0} \cdot \binom{8}{2} + \binom{4}{2} \cdot \binom{6}{2} + \binom{6}{2} \cdot \binom{4}{2} + \binom{8}{2} \cdot \binom{2}{0} =$$

$$= 28 + 90 + 90 + 28 = 236 = 1 \cdot 21 + 3 \cdot 10 + 5 \cdot 3 = 66.$$

A 9. táblázat tartalmazza az $R(n)/R(n+1)$, $R_z(n)/R_z(n+1)$, $E(n)/R(n)$, $E(n)/E(n+1)$, $E_z(n)/E_z(n+1)$ és $E_z(n)/R_z(n)$ hányadosokat $n = 1, \ldots, 32$ csúcs esetén.

$R(n)$ értékét a (2), az $R_z(n)$ értékét pedig a (3) egyenlőségek, $E(n)$ értékét az 5.1., míg $E_z(n)$ értékét az 5.2. következmény alapján számítottuk.

Érdemes megjegyezni, hogy $R(101)/R(102)$ és $R_z(101)/R_z(102)$ első kilenc decimális számjegye megegyezik.

A 9. táblázatban lévő adatok azt mutatják, hogy ha $1 \leq n \leq 32$, és n páratlan, akkor $E_z(n)/R_z(n) = 0,5$. Ez a tulajdonság n nagyobb értékei esetén is jellemző a hányadosra.

5.1. Lemma. *Ha* $1 \leq k \leq 600$, *akkor*

$$\frac{E_z(2k-1)}{R_z(2k-1)} = 0,5.$$

Bizonyítás. Lásd $R_z(n)$ és $E_z(n)$ pontos értékeit [128]. □

A 10. táblázat az $E_z(n)/G_z(n)$ hányadosokat tartalmazza $n = 1, \ldots, 31$ csúcs esetén, míg a 12. táblázat a $G_z(n)/T(n)$ hányadosokat $n = 30$ és $n = 31$ csúcs esetén.

n	$\frac{R(n)}{R(n+1)}$	$\frac{R_z(n)}{R_z(n+1)}$	$\frac{E(n)}{R(n)}$	$\frac{E(n)}{E(n+1)}$	$\frac{E_z(n)}{E_z(n+1)}$	$\frac{E_z(n)}{R_z(n)}$
1	0.333333	0.000000	1.000000	0.000000	0.000000	– – –
2	0.300000	0.250000	0.666667	0.500000	0.500000	1.000000
3	0.287714	0.266667	0.600000	0.222220	0.222222	0.500000
4	0.277778	0.257857	0.487179	0.321427	0.321429	0.600000
5	0.270562	0.266667	0.523810	0.254545	0.254545	0.500000
6	0.269231	0.265151	0.510823	0.277778	0.277778	0.523810
7	0.266667	0.263736	0.505828	0.260698	0.260698	0.500000
8	0.264706	0.262500	0.502720	0.265559	0.265559	0.505828
9	0.263158	0.261437	0.501440	0.260687	0.260687	0.500000
10	0.261905	0.260526	0.500682	0.261276	0.261276	0.501440
11	0.260870	0.259740	0.500357	0.259555	0.259555	0.500000
12	0.260000	0.259058	0.500171	0.259243	0.259243	0.500357
13	0.259259	0.258461	0.500089	0.258415	0.258416	0.500000
14	0.258621	0.257937	0.500043	0.257982	0.257982	0.500089
15	0.258065	0.257471	0.500022	0.257460	0.257460	0.500000
16	0.257578	0.257056	0.500011	0.257068	0.257068	0.500022
17	0.257143	0.256684	0.500005	0.256682	0.256682	0.500000
18	0.256757	0.256349	0.500003	0.256352	0.256352	0.500006
19	0.256410	0.256046	0.500001	0.256045	0.256045	0.500000
20	0.256098	0.255769	0.500001	0.255770	0.255770	0.500001
21	0.255814	0.255517	0.50000034	0.255517	0.255517	0.500000
22	0.255556	0.255285	0.50000016	0.255286	0.255286	0.50000034
23	0.255319	0.255072	0.50000009	0.255072	0.255072	0.50000000
24	0.255102	0.254876	0.50000004	0.254876	0.254876	0.50000000
25	0.254902	0.254694	0.50000002	0.254694	0.254694	0.50000009
26	0.254717	0.254525	0.50000001	0.254525	0.254525	0.50000000
27	0.254545	0.254368	0.50000001	0.254368	0.254368	0.50000000
28	0.254386	0.254221	0.50000000	0.254221	0.254221	0.50000000
29	0.254237	0.254083	0.50000000	0.254083	0.254083	0.50000000
30	0.254098	0.253854	0.50000000	0.253955	0.253955	0.50000000
31	0.253968	0.253834	0.50000000	0.253834	0.253834	0.50000000
32	0.253846	0.253720	0.50000000	0.253720	0.253720	0.50000000

9. táblázat. Az $R(n)/R(n+1)$, $R_z(n)/R_z(n+1)$, $E(n)/R(n)$, $E(n)/E(n+1)$, $E_z(n)/E_z(n+1)$ és $E_z(n)/R_z(n)$ hányadosok $n = 1, \ldots, 32$ csúcs esetén.

A 10. táblázat adatai azt mutatják, hogy a $G_z(n)/E_z(n)$ sorozat csökkenő. Feltesszük, hogy a sorozat monoton csökkenve nullához tart, ha n tart a végtelenhez (hasonlóan ahhoz, ahogy a $G(n)/E(n)$ sorozat tart nullához [95, page 260, Corollary 23]).

A 11. táblázat tartalmazza $G_z(n)$ és $G_n(n)$ értékét $n = 1, \ldots, 30$ esetén, valamint $G_z(n+1)/G_z(n)$ és $G(n+1)/G(n)$ értékét $n = 1, \ldots, 31$ esetén.

A 12. táblázat tartalmazza a $G_z(n)$, $T(n)$ és $G_z(n)/T(n)$ értékeit $n = 30$ és $n = 31$ csúcs esetén: a gráfos és tesztelt sorozatok számának aránya lényegesen

n	$G_z(n)$	$E_z(n)$	$G_z(n)/E_z(n)$
01	0	0	– – –
02	1	1	1.000000
03	2	2	1.000000
04	7	9	0.777778
05	20	28	0.714286
06	71	110	0.645455
07	240	396	0.606061
08	871	1 519	0.573404
09	3 148	5 720	0.550350
10	11 655	21 942	0.531173
11	43 332	83 980	0.515980
12	162 769	323 554	0.503149
13	614 198	1 248 072	0.492117
14	2 330 537	4 829 708	0.482542
15	8 875 768	18 721 080	0.474106
16	33 924 859	72 714 555	0.466548
17	130 038 230	282 861 360	0.459724
18	499 753 855	1 101 992 870	0.453500
19	1 924 912 894	4 298 748 300	0.447784
20	7 429 160 296	16 789 046 494	0.442500
21	28 723 877 732	65 641 204 200	0.437589
22	111 236 423 288	256 895 980 068	0.433002
23	431 403 470 222	1 006 308 200 040	0.428699
24	1 675 316 535 350	3 945 186 233 014	0.424648
25	6 513 837, 679 610	15 478 849 767 888	0.420821
26	25 354 842 100 894	60 774 332 618 300	0.417197
27	98 794 053 269 694	238 775 589 937 976	0.413752
28	385 312 558 571 890	938 702 947 395 204	0.410473
29	1 504 105 116 253 904	3 692 471 324 505 040	0.407344
30	5 876 236 938 019 300	14 532 512 180 224 216	0.404351
31	22 974 847 474 172 100	57 224 797 531 384 560	0.401484
32	– – –	225 441 858 758 287, 187	– – –
33	– – –	888 545 038 032 771 168	– – –
34	– – –	3 503 546 152 385 412 870	– – –
35	– – –	13 820 048 716 545 422 988	– – –
36	– – –	54 534 996 163 146 555 910	– – –

10. táblázat. $G_z(n)$, $E_z(n)$ és $G_z(n)/E_z(n)$ $n = 1, \ldots, 31$ csúcs esetén.

magasabb, mint kisebb n-ek esetén, és ez a sorozat növekvő. A drasztikus változás annak köszönhető, hogy EGE2 sok nemgráfos sorozat tesztelését átugorja.

A 13. táblázat negyedik oszlopa alátámasztja Royle Gordon következő sejtését.

5.1. Sejtés. (Royle, 2012 [159]). *Ha n tart a végtelenbe, akkor a $G_z(n+1)/G_z(n+1)$ sorozat tart a 4-hez.*

n	$G_z(n)$	$G(n)$	$\frac{G_z(n+1)}{G_z(n)}$	$\frac{G(n+1)}{G(n)}$
1	0	1	− − −	0.500000
2	1	2	2.000000	2.000000
3	2	4	3.500000	3.750000
4	7	11	2.857143	2.818182
5	20	31	3.550000	3.290323
6	71	102	3.380282	3.352941
7	240	342	3.629167	3.546784
8	871	1 213	3.614237	3.595218
9	3 148	4 361	3.702351	3.672552
10	11 655	16 016	3.717889	3.705544
11	43 332	59 348	3.756323	3.742620
12	162 769	222 117	3.773434	3.786674
13	614 198	836 315	3.794439	3.802710
14	2 330 537	3 166 852	3.808465	3.817067
15	8 875 768	12 042 620	3.822189	3.828918
16	33 924 859	45 967 479	3.833125	3.839418
17	130 038 230	176 005 709	3.843130	3.848517
18	499 753 855	675 759 564	3.851172	3.856630
19	1 924 912 894	2 600 672 458	3.859479	3.863844
20	7 429 160 296	10 029 832 754	3.866369	3.870343
21	28 723 877 732	38 753 710 486	3.872612	3.876212
22	111 236 423 288	149 990 133 774	3.878257	3.881553
23	431 403 470 222	581 393 603 996	3.883410	3.886431
24	1 675 316 535 350	2 256 710 139 346	3.888124	3.890907
25	6 513 837, 679 610	8 770 547 818 956	3.894458	3.895031
26	25 354 842 100 894	34 125 389 919 850	3.895503	3.897978
27	98 794 053 269 694	132 919 443 189 544	3.900159	3.898843
28	385 312 558 571 890	518 232 001 761 434	3.903597	3.902238
29	1 504 105 116 253 904	2 022 337 118 015 338	3.906814	3.905666
30	5 876 236 938 019 300	7 898 574 056 034 638	3.909789	3.908734
31	22 974 847 474 172 100	30 873 429 530 206 738	− − −	− − −

11. táblázat. $G_z(n)$, $G(n)$, $G_z(n+1)/G_z(n)$ és $G(n+1)/G(n)$ értéke $n = 1, \ldots, 31$ csúcs esetén.

n	$G_z(n)$	$T(n)$	$G_z(n)/T(n)$
31	5 876 236 938 019 300	6 790 865 476 867 340	86, 531487
32	22 974 847 471 172 100	26 507 499 250 791 700	86, 673010

12. táblázat. $G_z(n)$, $T(n)$ és $G_z(n)/T(n)$ $n = 30$ és $n = 31$ csúcs esetén.

Azt gondoljuk, hogy a következő sejtés [90] is igaz.

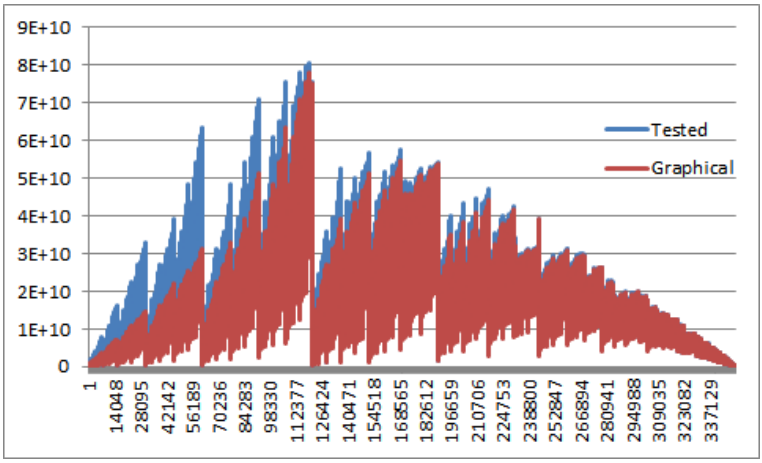
5.2. Sejtés. *Ha n tart a végtelenbe, akkor $G(n+1)/G(n)$ tart a 4-hez.*

A 12. táblázat szerint az $n = 30$ esetben a tesztelt potenciális gráfos sorozatok 85.4 százaléka, míg az $n = 31$ esetben a sorozatok 86.67 százaléka gráfos. Ezért a futási idő tovább csökkenthető, ha a lineáris időt igényelő tesztelés nélkül, konstans idő alatt meg tudjuk állapítani, hogy a vizsgált sorozat gráfos.

n	$G_z(n)$	$G(n)$	$\frac{G_z(n+1)}{G_z(n)}$	$\frac{G(n+1)}{G(n)}$
1	0	1	0.000000	0.500000
2	1	2	2.000000	2.000000
3	2	4	3.500000	3.750000
4	7	11	2.857143	2.818182
5	20	31	3.550000	3.290323
6	71	102	3.380282	3.352941
7	240	342	3.629167	3.546784
8	871	1 213	3.614237	3.595218
9	3 148	4 361	3.702351	3.672552
10	11 655	16 016	3.717889	3.705544
11	43 332	59 348	3.756323	3.742620
12	162 769	222 117	3.773434	3.786674
13	614 198	836 315	3.794439	3.802710
14	2 330 537	3 166 852	3.808465	3.817067
15	8 875 768	12 042 620	3.822189	3.828918
16	33 924 859	45 967 479	3.833125	3.839418
17	130 038 230	176 005 709	3.843130	3.848517
18	499 753 855	675 759 564	3.851172	3.856630
19	1 924 912 894	2 600 672 458	3.859479	3.863844
20	7 429 160 296	10 029 832 754	3.866369	3.870343
21	28 723 877 732	38 753 710 486	3.872612	3.876212
22	111 236 423 288	149 990 133 774	3.878257	3.881553
23	431 403 470 222	581 393 603 996	3.883410	3.886431
24	1 675 316 535 350	2 256 710 139 346	3.888124	3.890907
25	6 513 837, 679 610	8 770 547 818 956	3.894458	3.895031
26	25 354 842 100 894	34 125 389 919 850	3.895503	3.897978
27	98 794 053 269 694	132 919 443 189 544	3.900159	3.898843
28	385 312 558 571 890	518 232 001 761 434	3.903597	3.902238
29	1 504 105 116 253 904	2 022 337 118 015 338	3.906814	3.905666
30	5 876 236 938 019 300	7 898 574 056 034 638	3.909789	3.908734
31	22 974 847 474 172 100	30 873 429 530 206 738	– – –	– – –

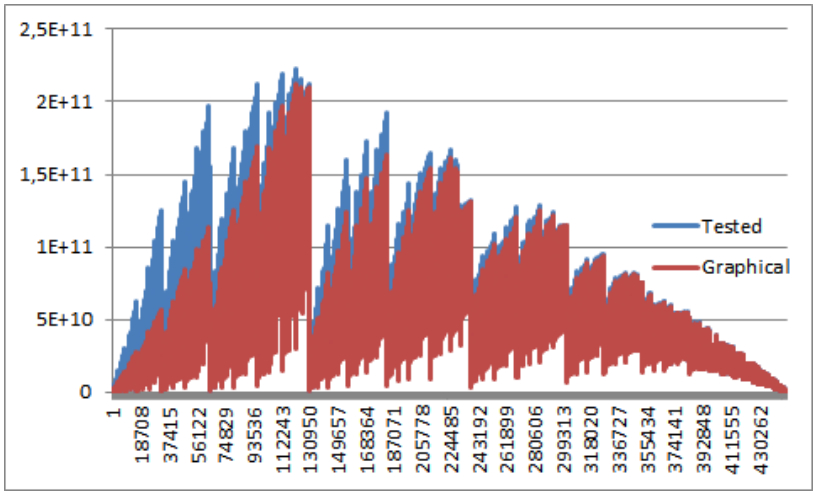
13. táblázat. Nullamentes gráfos sorozatok $G_z(n)$ száma és gráfos sorozatok $G(n)$ száma, továbbá a $G_z(n)/G_z(n+1)$ és $G(n)/G(n+1)$ hányadok $n = 1, \ldots, 30$ csúcs esetén.

A 4. ábra a tesztelt és a gráfos sorozatok számát mutatja a szeletek sorszámának függvényében $n = 30$ csúcs esetén.



4. ábra. A tesztelt és a gráfos sorozatok száma a szeletek sorszámának függvényében $n = 30$ csúcs esetén.

Az 5. ábra a hasonló adatokat mutatja $n = 31$ csúcs alapján.



5. ábra. A tesztelt és a gráfos sorozatok száma a szeletek sorszámának függvényében $n = 31$ csúcs esetén.

Megjegyezzük, hogy a folyóirat honlapján a 4. és 5. ábrák színesek (a grafikus sorozatokat piros, míg a tesztelt sorozatokat kék szín jelzi).

6. Párhuzamos Erdős–Gallai-algoritmus (EGP)

A $G(n)$ értékek kiszámítása hosszú ideig tart, ha szekvenciális programot használunk, ezért az EGE gyorsított párhuzamos változatát használtuk fel. A felhasznált processzorok és a $G_z(n)$ kiszámításához szükséges idő megközelítőleg fordítottan arányos, így minél több processzort használunk fel, annál kevesebb időre lesz szükségünk.

A lineáris algoritmusunk használhatósága érdekében szükség van egy algoritmusra, amely képes az ellenőrzendő sorozatok egy részén dolgozni.

Az Erdős–Gallai-Parallel algoritmus felhasználásával kiszámítottuk ezeket a számokat $n = 31$-ig. Az értékek [95] 2. táblázatában, valamint az OEIS-ben [96] találhatóak.

Alkalmazásunk két részre bontható: egy szerverre és egy kliensre. A szerver tárolja a kliensek közti feladatok felosztásához szükséges információkat, valamint összegyűjti az eredményeket. A kliens tárolja a szerver IP címét és portját az új feladatok kéréséhez.

A párhuzamos algoritmus egyik legkritikusabb része a probléma közel azonos méretű részfeladatokra osztása. A következő egyenlet megadja, hogy közelítőleg hány azonos fejjel kezdődő sorozatok létezik. Ezen számok ismeretében korlátozott méretű feladatok generálására vagyunk képesek, más szóval egyik feladat sem lesz nagyobb egy megadott maximumnál.

Könnyen belátható [95, (22) egyenlőség], hogy az (l, u, m)-szabályos sorozatok $Q(l, u, m)$ száma

$$R(l, u, m) = \binom{u - l + m}{m}. \tag{11}$$

(11) alapján készítettük el a feladatokat generáló algoritmust.

6.1. Algoritmus. Generate-Matrix(n, ms, M)

Bemenet. n: a sorozat hossza;
ms: egy feladat maximális mérete.
Kimenet. M: a feladatok paramétereit tartalmazó mátrix.
Munkaváltozók. i, j ciklusváltozók.

1. **for** $i = n$ **downto** 2 // 1–3. sor: a mátrix kitöltése
2. **for** $j = 1$ **to** $n - 1$
3. $M_{i,j} = \binom{i+j-2}{i-1}$
4. **for** $j = n - 1$ **downto** 1 // 4–5. sor: a mátrix első sorának kitöltése

```
5.      M_{1,j} = 1
6.  GENERATE-NEW-SEQUENCES(M, n, n, 1, n − 1, ms, 0)
                                          // 6. sor: új feladat előállítása
7.  return M                              // 7. sor: eredmény visszaadása
```

Az algoritmus egy, az egyenlőség felhasználásával kapott értékekkel feltöltött mátrixot ad vissza. Ezek után képesek leszünk a sorozatok generálására. Induljunk el a mátrix alsó sorának első elemétől, és kezdjük el az értékeket kiolvasni és összegezni. Abban az esetben, amennyiben az érték túl nagy a maximális mérethez viszonyítva, lépjünk feljebb egy sorral, és kezdjük el kiolvasni és összegezni az elemeket az első sortól kezdve addig az oszlopig, ahonnan felléptünk. Addig folytassuk az algoritmust, amíg el nem érjük a szükséges méretet. A következő (rekurzív) algoritmus a fent említett eljárás szerint működik.

6.2. Algoritmus. GENERATE-NEW-SEQUENCE(M, n, i, j, jm, ms, J)

Bemenet. n: a sorozat hossza;
ms: egy feladat maximális mérete.
Kimenet. M: a feladatok paramétereit tartalmazó mátrix.
Munkaváltozók. i, j: ciklusváltozók.

```
1.  S = 0                             // 1. sor: aktuális szelet méretének beállítása
2.  while j < j_m + 1
3.      if S + M_{i,j} ≤ ms           // 3. sor: ha további sorozatot tudunk hozzáadni
4.          S = S + M_{i,j}           // 4. sor: további sorozatot adunk a szelethez
5.          if j ≤ j_m                // 5–6. sor: átlépés a mátrix következő oszlopához
6.              j = j + 1
7.      else if S ≠ 0                 // 7. sor: szelet nem üres
8.              for k = 2 to size(J, 2)   // 8–13. sor: nyomtatás
9.                  print(J_k)
10.             for k = 1 to n − size(J, 2) + 1
11.                 print(j − 1)
12.             print  newline        // 13. sor: új sor
13.             S = 0
14.         if M_{i,j} > ms and j ≤ j_m   // 14. sor: ha felbonthatatlan
15.             GENERATE-NEW-SEQUENCE(M, n, i − 1, 1, j, ms, [J, j])
16.             j = j + 1
17. if S ≠ 0                          // 17. sor: utolsó szelet nem üres
18.     for k = 2 to size(J, 2)       // 18–22. utolsó szelet nyomtatása
19.         print (J_k)
20.     for k = 1 to n − size(J, 2) + 1
21.         print (J(size(J, 2)))
22.     print  newline
```

Most már rendelkezünk a probléma kisebb méretű részfeladataival. Ezt követően elkezdhető a szerver program segítségével a különböző számítógépek közötti szétosztásuk. A DISTRIBUTING-JOBS algoritmus megmutatja, hogy a szerver miképp küldi el a feladatokat a klienseknek. Az algoritmusban csak a feladatok szétosztására koncentráltunk, így nem tartalmazza a hálózati kommunikációval foglalkozó kódot, kivéve a nagyon fontos hálózati primitíveket (számítógépes hálózatokról bővebben olvashatunk [178]-ban).

6.3. Algoritmus. DISTRIBUTING-JOBS(n, N, M, G_z)

Bemenet. n: a sorozat hossza;
N: a feladatok becsült száma;
M: a feladatok paramétereit tartalmazó mátrix.
Kimenet. G_z: az n-szabályos nullamentes gráfos sorozatok száma.
Munkaváltozók. $S = S_0, \ldots, S_n$: a feladatok állapotát tároló vektor;
fj: befejezett feladatok száma;
aj: a kliensnek elküldött utolsó feladat száma;
ji: a bejövő eredmény feladatának indexe;
cl: kliens azonosító (hálózati kommunikációhoz szükséges);
msg: klienstől jövő üzenet (csak hálózati kommunikációhoz szükséges);
S: aktuális feladat mérete;
$time$: aktuális feladat futási ideje másodpercben;
al: alsó korlát;
bu: felső korlát.

```
1.  S_0 = True                          // 1–4. sor: szelet állapot kezdeti beállítása
2.  S_{N+1} = True
3.  for j = 1 to N + 1
4.      S_j = False
5.  G_z = 0                             // 5. sor: G_z kezdeti értékének beállítása
6.  while fj < N                        // 6. sor: amíg van befejezetlen szelet
7.      accept(cl)                      // 7. sor: klienssel való kapcsolat elfogadása
8.      recv(cl, msg)                   // 8. sor: kliens üzenetének fogadása
9.      if msg = 0                      // 9. sor: kliens új szeletet kér
10.         aj = aj + 1  // 10. sor: utoljára küldött szelet indexének növelése
11.         for i = M_{aj-1,0} to n     // 11–12. sor: kezdő sorozat frissítése
12.             b_i = n + M_{aj-1,1}
13.         while S_{aj} = True or aj > N   // 13–22. sor: befejezetlen szelet?
14.             aj = aj + 1
15.             if aj > N               // 15. sor: átléptük a maximális indexet
16.                 aj = 1              // 16. sor: index beállítása 1-re
17.             for i = M_{aj-1,0} to n // 17–18. kezdő sorozat frissítése
```

18. $b_i = n + M_{aj-1,1}$
19. **if** $aj < N$ // 19–30. sor: utolsó szeletet jellemző sorozatok beállítása
20. $al = M_{aj,0}$
21. $b = n + M_{aj,1}$
22. **else** $al = 1$
23. $bu = 1$
24. **send**(c, b, al, bu) // 24. sor: szelet küldése a kliensnek
25. **else recv**$(c, ji, F_{init}, F_{last}, Z_{n,m}, time)$ // 25. sor: eredmény fogadása
26. **if** $S_{ji} =$ FALSE // 26. sor: kapott eredmény új
27. $S_j =$ TRUE // 27. sor: befejezett szelet állapotának beállítása
28. $fj = fj + 1$ // 28. sor: befejezett szeletek számának növelése
29. $G_z = G_z + Z_{n,m}$ // 29. sor: G_z frissítése
30. **close**(cl) // 30. sor: hálózati kapcsolat zárása
31. **return** G_z // 31. sor: eredmény

A kliens készítésekor fontos szempont volt az egyszerűség. Egy olyan programot szerettünk volna elkészíteni, amelynek nincs szüksége semmilyen felhasználói interakcióra. Elegendő, ha a felhasználó csak elindítja, és attól a pillanattól kezdve a program önállóan képes futni a háttérben. Ez azért fontos, mert a programot annyi helyen akartuk kis részekre osztva futtatni, amennyi helyet fel tudtunk használni számítógépes laboratóriumunkban, mivel nem volt elég időnk és emberünk, akik részt vettek volna a program futtatásában.

Egy másik fontos ötlet az volt, hogy nem akartuk újraindítani a programot, amikor a számítás $G_z(n)$-ről átvált $G_z(n+1)$-re. Miután a kliens végzett feladatával, és a szerver nem tudott neki újat adni, a háttérben várakozott – mindaddig amíg nem kapott új feladatot – jelentős erőforrás-felhasználás nélkül.

A kliens program szálként működik. Ennek egyszerű oka van: a programot feltöltöttük a publikus honlapunkra, így bárki csatlakozhatott a számításokhoz. Ezzel az volt a célunk, hogy egyetlen felhasználót se veszítsünk el amiatt, hogy a programunk lefoglalja az erőforrásait.

Harmadik szempontként egy igazán gyors programot szerettünk volna készíteni, mivel a futási idő óriási mértékben megnövekedhet az n értékétől függően. Ezen okból az ANSI C nyelvet használtuk programunk elkészítésekor. Kísérleteink során kiderült, hogy programunk ANSI C-ben írt változata százszor gyorsabb volt, mint a MATLAB-ban készült változat. A hálózati kommunikációhoz Berkeley-socketeket használtunk.

A kliens a következőképp működik:

– Miután létrehoztuk a csatlakozáshoz szükséges csatlakozót (socket), megpróbálunk kapcsolódni a szerverhez. Ha ez nem lehetséges, várunk egy ideig, és megkétszerezzük ezt a várakozási időt. Ezt az eljárást addig ismételjük, amíg nem tudunk csatlakozni a szerverhez, vagy el nem érünk egy megadott időkorlátot. Ezután már nem növeljük a várakozási időt. Könnyen belátható, hogy a várakozási idő exponenciálisan nő.

– A szerverre való csatlakozás után elkérünk egy részfeladatot, majd miután megkaptuk, lebontjuk a hálózati kapcsolatot.

– Kiszámítjuk a $G_z(n)$ részleges eredményét, majd visszaküldjük a szervernek az első lépésben bemutatott csatlakozási eljárás szerint.

A klienseken futó Parallel Erdős–Gallai-algoritmus két részre osztható: Check és Enumerating részre. Az első csak ellenőrzi a sorozatokat, mást nem csinál. A második generálja a sorozatokat, a H értékeket és az ellenőrző pontokat.

A Check-ben a lineáris Erdős–Gallai-algoritmus módosított változatát használjuk.

6.4. Algoritmus. Check(b, H, c, C)

Bemenet. b: bemenő sorozat;
$H = H_1, \ldots, H_n$: a b sorozat elemeinek összege;
c: ellenőrző pontok száma;
$C = C_1, \ldots, C_{n-1}$: ellenőrző pontok.

Kimenet. L: logikai érték. Ha a megvizsgált sorozat gráfos, akkor $L = 1$, különben $L = 0$.

Munkaváltozó. p: aktuális ellenőrző pont.

```
1.  i = 1                                          // 1. sor: i kezdeti értékének beállítása
2.  while i ≤ c and H_{C_i} > C_i(C_i − 1)          // 2–10. sor: sorozatok tesztelése
3.      p = C_i                                    // 3. sor: p kezdeti értékének beállítása
4.      while J_p < n and b_{J_{p+1}} > p          // 4–7. sor: J_p frissítése
5.          J_p = J_p + 1
6.      while J_p > p and b_{J_p} ≤ p
7.          J_p = J_{p−1}
8.      if H_p > H_n − H_{J_p} + p(J_p − 1)        // 8. sor: tesztelés
9.          L = 0                                  // 9–10. sor: b nem gráfos
10.         return L
11.     i = i + 1
12. L = 1                                          // 12–13. sor: b gráfos
13. return L
```

6.5. Algoritmus. ENUMERATING$(n, b, last_index, last_value)$

Bemenet. n: a sorozat hossza;
b: első sorozat;
$last\_index$: az utolsó ellenőrizendő sorozat elérésekori elem indexe;
$last\_value$: az utolsó ellenőrizendő sorozat elérésekori elem értéke.
Kimenet. G_z^p: Az első és utolsó ellenőrizendő sorozat közötti n-szabályos nulla mentes gráfos sorozatok száma.

```
1.  H_1 = b_1                                          // 1. sor: H_1 beállítása
2.  for i = 2 to n                                     // 2–3. sor: H elemeinek számítása
3.      H_i = H_{i-1} + b_i
4.  if b_n ≠ n − 1                                     // 4. sor: ha a gráf nem teljes
5.      if H_n páratlan                                // 5–10. sor: sorozat frissítése
6.          b_n = b_n − 3
7.          H_n = H_n − 3
8.      else b_n = b_n − 2
9.          H_n = H_n − 2
10. for i = 1 to n          // 10–11. sor: súlypontok kezdeti értékeinek beállítása
11.     J_i = n − 1
12. for i = 1 to n − 2                  // 12–15. sor: ellenőrző pontok számítása
13.     if b_i ≠ b_{i+1} and b_i ≠ b_n
14.         c = c + 1
15.         C_c = i
16. L = CHECK(b, H, c, C)                              // 16. sor: első sorozat tesztelése
17. G_z^p = G_z^p + L
18. while b_{last_index} > last_value                  // 18. sor: szelet utolsó sorozatáig
19.     k = n                // 19. sor: munkaváltozó kezdeti értékének beállítása
20.     if b_k = 1                                     // 20. sor: ha a sorozat utolsó eleme 1
21.         j = n − 1
22.         while b_j ≤ 1
23.             j = j − 1
24.         if b_j = 2                                 // 24. sor: ha az utolsó elem 2
25.             b_{j-1} = b_{j-1} − 1                  // 25. sor: sorozat frissítése
26.             H_{j-1} = H_{j-1} − 1                  // 26. sor: H elemeinek frissítése
27.             if j > 2                // 27–36. sor: ellenőrző pontok frissítése
28.                 if (c ≤ 2 or (c > 2 and C_{c-2} ≠ j − 2)) and
                                (c > 1 and C_{c-1} ≠ j − 2)
29.                     if c > 1 and C_{c-1} > j − 2
```

30. $C_{c+1} = C_c$
31. $C_c = C_{c-1}$
32. $C_{c-1} = j - 2$
33. $c = c + 1$
34. **else** $C_{c+1} = C_c$
35. $C_c = j - 2$
36. $c = c + 1$
37. **for** $k = j$ **to** n
38. $b_k = b_{j-1}$ // 39. sor: b utolsó részének frissítése
39. $H_k = H_{k-1} + b_k$ // 40. sor: H frissítése
40. **while** $c > 1$ **and** $C_c > j - 1$ // 42–43. sor: ellenőrző pontok frissítése
41. $c = c - 1$
42. **if** H_n páratlan // 42. sor: ha a paritás páratlan
43. $b_n = b_{n-1} - 1$ // 43. sor: b frissítése
44. $H_n = H_{n-1} + b_n$ // 44. sor: H frissítése
45. $c = c + 1$ // 45–46. sor: ellenőrző pontok frissítése
46. $C_c = n - 1$
47. **else** $b_j = b_j - 1$ // 47. sor: b frissítése
48. $H_j = H_j - 1$ // 48. sor: H frissítése
49. **if** $j > 1$ // 49–50. sor: ellenőrző pontok frissítése
50. **if** ($c = 1$ **and** $C_c \neq j - 1$) **or** ($c > 1$ **and** $C_{c-1} \neq j - 1$)
51. **if** $c > 0$ **and** $C_c > j - 1$
52. $C_{c+1} = C_c$
53. $C_c = j - 1$
54. $c = c + 1$
55. **for** $k = j + 1$ **to** n
56. $b_k = b_j$ // 56. sor: b frissítése
57. $H_k = H_{k-1} + b_k$ // 57. sor: H frissítése
58. **while** $c > 1$ **and** $C_c > j - 1$ // 58–59. sor: ellenőrző pontok frissítése
59. $c = c - 1$
60. **if** H_n páratlan // 60. sor: paritás ellenőrzése
61. $b_n = b_n - 1$ // 61. sor: b frissítése
62. $H_n = H_n - 1$ // 62. sor:H frissítése
63. $c = c + 1$ // 63. sor: ellenőrző pontok frissítése
64. $C_c = n - 1$ // 64. sor: új ellenőrző pont listához fűzése
65. **else if** $b_k = 2$

66. $b_{k-1} = b_{k-1} - 1$ // 66. sor: b frissítése
67. $H_{k-1} = H_{k-1} - 1$ // 67. sor: H elemeinek frissítése
68. **if** ($c = 1$ **and** $C_c \neq n-2$) **or** ($c > 1$ **and** $C_{c-1} \neq n-2$ **and** $C_c \neq n-2$) // 68–73. sor: ellenőrző pontok frissítése
69. **if** $c > 0$ **and** $C_c > n-2$
70. $C_{c+1} = C_c$
71. $C_c = n-2$
72. **else** $c = c+1$
73. $C_c = n-2$
74. **if** b_{k-1} odd // 74. sor: paritás tesztelése
75. $b_k = b_{k-1}$ // 75. sor: b frissítése
76. **if** $c > 0$ **and** $C_c = n-1$
77. $c = c-1$ // 77. sor: ellenőrzőpont frissítése
78. **else** $b_k = b_{k-1} - 1$ // 78. sor: b frissítése
79. $H_k = H_{k-1} + b_k$ // 79. sor: H elemének kiszámítása
80. **else** $b_k = b_k - 2$ // 80. sor: b frissítése
81. $H_k = H_k - 2$ // 81. sor: H elemének kiszámítása
82. **if** $c < 1$ **or** $C_c \neq n-1$ // 82–84. sor: ellenőrző pontok frissítése
83. $c = c+1$
84. $C_c = n-1$
85. $G_z^p = G_z^p +$ CHECK(b, H, c, C) // 85. sor: G_z^p frissítése

A *The On-Line Encyclopedia of Integer Sequences* [170] tartalmazza az n csúcsú egyszerű gráfok foksorozatainak számát. Az $n = 24, \ldots, 29$ értékeket november 16-án töltöttük fel. A [95] cikk 2. táblázatában megtalálható az összes $G(n)$ érték, amelyet az OEIS adatbázis jelenleg tartalmaz.

Számításaink során több mint 200 számítógépet használtunk fel. Számítási teljesítményüket összegezve elmondható, hogy számítási kapacitásunk elméleti maximuma – beleértve a magánszemélyek teljesítményét is – elérte a 6 TFLOPS-ot.

A gráfos sorozatok kiszámításának futási idejét a 14. táblázat tartalmazza. Észrevehető, hogy a futási idő növekedése nem ugyanolyan arányú különböző n értékek esetén. Ez a felhasznált processzorok típusával magyarázható. A korai számítások idején (például $n = 25$ esetén) néhány erős géppel rendelkeztünk, azonban a növekvő n-ek során fellépő nagyobb komplexitás miatt inkább sok kevéssé erős gépet használtunk. A számítások teljes ideje kevesebb, ha néhány erős gépet használunk, de a valós futási idő nőni fog. Több mint 200 gép felhasználásával a G_{29} valós futási ideje több, mint két hét volt.

n	Futási idő (nap)	Szeletek száma
25	26	435
26	70	435
27	316	435
28	1130	2 001
29	6733	15 119

14. táblázat. A mért futási idők összege, valamint a feldolgozott szeletek száma.

7. A program további gyorsításának lehetőségei

Leszámláló programunk futási idejét két fő módszerrel csökkenthetjük: az egyik módszer a tesztelendő sorozatok számát, a másik pedig az egyes sorozatok ellenőrzésének idejét csökkenti.

Ebben a részben az előbbi módszer három érdekes elemét, a rossz és jó sorozatok egy részének átugrását elemezzük.

7.1. A nemgrafikus sorozatok egy részének átugrása

Ezt a módszert az $n = 30$-as projektben vezettük be.

Az $n = 23, \ldots, 29$ esetben a foksorozatokat a páros sorozatok között kerestük. Ekkor a 7 516 816 644 943 560 páros sorozat között a 2 022 337 118 015 338 grafikus sorozat 26, 90 százalékot képviselt. Ismert (lásd a [95] cikkben a 13. lemmát), hogy $E(n) = \Theta\left(4^n/\sqrt{n}\right)$, míg Burns tétele (lásd az előbbi cikkben a 22. tételt) $G(n)$ szerint van olyan pozitív C, amelyre $G(n) = O\left(4^n/\sqrt{n}/(\log n)^C\right)$, ahonnan következik, hogy a grafikus sorozatok a páros sorozatok között aszimptotikusan nullmértékű jelentős részét részhalmazt képviselnek.

Az $n = 30$ és $n = 31$ esetben a rossz sorozatok jelentős részét átugrottuk. Ennek köszönhetően az $n = 30$ esetben a tesztelt sorozatok 85, 40; míg az $n = 31$ esetben 86, 67 százaléka volt grafikus. Mivel a futási idők még nagyszámú processzor felhasználása mellett is jelentősek, kulcsfontosságú mind a nemgrafikus, mind pedig grafikus sorozatok minél nagyobb hányadának átugrása.

A nemgrafikus sorozatok jelentős részének tényleges ellenőrzését a következő lemma alapján ugrottuk át az $n = 30$ és $n = 31$ esetben.

Ha $b = (b_1, \ldots, b_n)$ $(0, n-1, n)$-szabályos sorozat, akkor a $c = (c_1, c_2, \ldots, c_n)$ sorozatot b-nél lexikografikusan (i, j)-kisebbnek nevezzük, ha vannak olyan $1 \le i < j \le n$ indexek, melyekre

$$\text{ha} \quad k = 1, \ldots, i, \quad \text{akkor} \quad c_k = b_k,$$

és

$$\sum_{k=i+1}^{n} c_k \le \sum_{k=i+1}^{n} b_k$$

és

$$\sum_{k=i+1}^{j} c_k < \sum_{k=i+1}^{j} b_k.$$

7.1. Lemma. *Ha $b = b_1, \ldots, b_n$ nemgrafikus sorozat, és vannak olyan i és j indexek és egy olyan $c = c_1, c_2, \ldots, c_n$ sorozat, amely lexikografikusan (i, j)-kisebb, mint b, akkor c nemgrafikus.*

Bizonyítás. Lásd [88]. □

A lemma alkalmazásával mintegy 30 százalékkal sikerült az EGE1 program futási idejét csökkenteni.

7.2. A kis grafikus sorozatok átugrása

Az előző definícióhoz és lemmához hasonló eszközökkel kezeljük a grafikus sorozatok egy részének átugrását.

Ha $b = (b_1, \ldots, b_n)$ szabályos sorozat, akkor a $c = (c_1, \ldots, c_n)$ sorozatot b-nél lexikografikusan (i, j)-nagyobbnak nevezzük, ha vannak olyan $1 \leq i < j < n$ indexek, melyekre

$$\text{ha} \quad k = 1, \ldots, i, \quad \text{akkor} \quad c_k = b_k$$

és

$$\sum_{k=i+1}^{j} c_k > \sum_{k=i+1}^{j} b_k,$$

továbbá

$$\sum_{k=i+1}^{n} c_k \geq \sum_{k=i+1}^{n} b_k.$$

7.2. Lemma. *Ha $b = b_1, b_2, \ldots, b_n$ nemgrafikus sorozat, és vannak olyan i és j indexek és egy olyan $c = c_1, c_2, \ldots, c_n$ sorozat, amely lexikografikusan (i, j)-nagyobb, mint b, akkor c nemgrafikus.*

Bizonyítás. Lásd [88]. □

7.3. A gráfos sorozatok komplementereinek átugrása

Ha n pozitív egész szám, és a $b = b_1 \leq \cdots \leq b_n \leq n - 1$ nemcsökkenő n-szabályos sorozat, akkor a $b' = n - 1 - b_n \leq \cdots \leq n - 1 - b_1$ sorozatot a b sorozat *nemcsökkenő komplementerének* nevezzük. Ehhez hasonlóan definiáljuk a b nemnövekvő n-szabályos sorozatok *nemnövekvő komplementerét* is.

A gráfos sorozatok átugrásának alapja lehet az az egyszerű észrevétel is, hogy egy gráfos sorozat komplementere is gráfos sorozat.

7.3. Lemma. *Ha egy nemnövekvő szabályos $b = b_1, b_2, \ldots, b_n$ sorozat gráfos, akkor a $b' = n-1-b_n, n-1-b_{n-1}, \ldots, n-1-b_1$ sorozat is gráfos.*

Bizonyítás. Ha b gráfos sorozat, akkor létezik olyan G egyszerű gráf, amelynek nemnövekvően rendezett foksorozata $b = b_1, b_2, \ldots, b_n$. Ennek a gráfnak van komplementere, és abban a fokszámok nemnövekvően rendezett sorozata $b' = n-1-b_1, n-1-b_2, \ldots, n-1-b_n$. Tehát a b' sorozat szintén gráfos. □

Ha egy nemnövekvő (vagy nemcsökkenő) n-szabályos sorozat nemnövekvően (nemcsökkenően) rendezett komplementer sorozata ugyancsak b, akkor akkor a b sorozatot *félpalindrom sorozatnak* nevezzük. A névválasztás oka, hogy a palindrom kifejezés már foglalt [2, 101, 102] az olyan sorozatokra, amelyek megegyeznek a saját megfordításukkal.

A $b = b_1, \ldots, b_n$ sorozat *szimmetrikus elempárjainak* a b_1—b_n, b_2—b_{n-1}, …, $b_{\lfloor (n+1)/2 \rfloor}$—$b_{\lceil (n+1)/2 \rceil}$ elempárokat nevezzük. Megjegyezzük, hogy ez a definíció azt is jelenti, hogy páratlan n esetén az utolsó szimmetrikus párban kétszer szerepel a sorozat középső eleme.

7.4. Lemma. *A $b = b_1, \ldots, b_n$ n-szabályos sorozat akkor és csak akkor félpalindrom sorozat, ha elempárjainak összege $n-1$.*

Bizonyítás. Az állítás a félpalindrom sorozatok definíciójából következik. □

Ha egy n-szabályos sorozat nem félpalindrom sorozat, akkor szimmetrikus elempárjainak elemei közül a lexikografikusan kisebbet (az adott rendezés szerint előbb sorra kerülőt) *erősnek*, a másik elemet pedig *gyengének* nevezzük.

7.3.1. Komplementerek átugrása szabályos sorozatok tesztelése során

Ha a potenciális gráfos sorozatok tesztelése során egy b sorozatról kiderül, hogy gráfos, akkor a b' sorozat is gráfos. Ha egy gráfos b sorozat b' komplementere nem azonos b-vel, akkor b'-t nem kell tesztelni. Ezért például majd az $n = 33$ csúcsú gráfok foksorozatainak leszámlálása során elegendő a legalább 16-tal kezdődő potenciális sorozatokat vizsgálni, mert ha egy b nemcsökkenő gráfos sorozatban $b_1 \leq 16$, akkor a nemcsökkenő b' sorozatban $b'_1 \geq 16$, azaz ennek a sorozatnak a komplementerét már korábban megvizsgáltuk.

Egy legalább 16-tal kezdődő b gráfos sorozat egyúttal félpalindrom sorozat, akkor *egy* gráfos sorozatot képvisel, egyébként azonban kettőt. Ezért a megtalált gráfos sorozatokat ellenőrizni kell, hogy félpalindrom sorozatok-e. A Félpalindrom algoritmus elvégzi ezt az ellenőrzést.

7.1. Algoritmus. Félpalindrom(n, b)

Bemenet. n: csúcsok száma ($n \geq 1$);
$b = b_1, \ldots, b_n$: a megvizsgálandó n-szabályos sorozat.
Kimenet. 0 vagy 1 (1, ha a vizsgált sorozat félpalindrom, egyébként 0).

Munkaváltozó. i: ciklusváltozó.

```
1. if n páratlan and (n+1)/2 ≠ (n+1)/2          // 1–6. sor: tesztelés
2.     return 0                                  // 1–5. sor: b nem félpalindrom sorozat
3. for i = 1 to ⌈n/2⌉
4.     if b_i + b_{n−i+1} ≠ n − 1
5.     return 0
6. return 1                                      // 6. sor: b félpalindrom sorozat
```

FÉLPALINDROM futási ideje legrosszabb esetben (például ha b félpalindrom sorozat) $\Theta(n)$, azonban az átlagos futási ideje ennél lényegesen kisebb, mivel a gráfos sorozatok között kevés félpalindrom sorozat van.

Egy nemnövekvő 1-szabályos sorozat van: 0 – ez félpalindrom sorozat. Három nemnövekvő 2-szabályos sorozat van: 1, 1; 0, 1 és 0, 0. Közülük az 1, 1 erős, a 0, 0 gyenge, a 0, 1 pedig félpalindrom sorozat. Tíz 3-szabályos sorozat van: 2, 2, 2; 2, 2, 1; 2, 2, 0; 2, 1, 1; 2,1, 0; 2, 0, 0; 1, 1; 1, 1, 0; 1, 0, 0 és 0, 0, 0. Ezek közül az 1, 1, 1 és a 0, 1, 2 félpalindrom, a többiek közül 4 erős és 4 gyenge. Harmincöt 4-szabályos sorozat van:
(3, 3, 3, 3); (3, 3, 3, 2); (3, 3, 3, 1); (3, 3, 3, 0); (3, 3, 2, 2); (3, 3, 2, 1); (3, 3, 2, 0); (3, 3, 1, 1); (3, 3, 1, 0); (3, 3, 0, 0); (3, 2, 2, 2); (3, 2, 2, 1); (3, 2, 2, 0); (3, 2, 1, 1); (3, 2, 1, 0); (3, 2, 0, 0); (3, 1, 1, 1); (3, 1, 1, 0); (3, 1, 0, 0); (3, 0, 0, 0); (2, 2, 2, 2); (2, 2, 2, 1); (2, 2, 2, 0); (2, 2, 1, 1); (2, 2, 1, 0); (2 ,2, 0, 0); (2, 1, 1, 1); (2, 1, 1, 0); (2, 1, 0, 0); (2, 0, 0, 0); (1, 1, 1, 1); (1, 1, 1, 0); (1, 1, 0, 0); (1, 0, 0, 0); (0, 0, 0, 0).

A százhuszonhat 5-szabályos sorozat közül
(4, 4, 2, 0, 0); (4, 3, 2, 1, 0); (4, 2, 2, 2, 0); (3, 3, 2, 1, 1); (3, 2, 2, 2, 1) és (2, 2, 2, 2, 2)
félpalindrom, a többieknek pedig fele erős, fele pedig gyenge. A 6-szabályos sorozatok között 10, a 7-szabályos sorozatok között pedig 20 félpalindrom sorozat van.

A következő állítás megadja az n-szabályos sorozatok között lévő félpalindrom, erős és gyenge sorozatok számát.

7.5. LEMMA. *Ha $n \geq 1$, akkor a nemcsökkenő n-szabályos sorozatok között*

$$P(n) = \binom{n-1}{\lfloor n/2 \rfloor}$$

n-félpalindrom sorozat, valamint

$$S(n) = W(n) = (E(n) - P(n))/2$$

erős és gyenge sorozat van.

Bizonyítás. Ha n páros, akkor a sorozat első $\lfloor n/2 \rfloor$ elemét ismétléssel választhatjuk ki a $[0, \rfloor(n-1)/2\rfloor]$ intervallum $\rfloor n/2 \rfloor$ eleme közül. Ha pedig n páratlan, akkor a

n	$R(n)$	$P(n)$	$S(n) = W(n)$	$W(n)/R(n)$
1	1	1	0	0.0000000000
2	3	1	1	0.3333333333
3	10	2	4	0.4000000000
4	35	3	16	0.4571428571
5	126	6	60	0.4761904762
6	462	10	226	0.4891774891
7	1 716	20	848	0.4941724942
8	6 435	35	3 200	0.4972804973
9	24 310	70	12 120	0.4985602633
10	92 378	126	46 126	0.4993180194
11	352 716	252	176 232	0.4996427721
12	1 352 078	462	675 808	0.4998291519
13	5 200 300	924	2 599 688	0.4999111590
14	20 058 300	1 716	10 028 292	0.4999572247
15	77 558 760	3 432	38 777 664	0.4999977875
16	300 540 195	6 435	150 266 880	0.4999989295
17	1 166 803 110	12 870	583 395 120	0.4999944849
18	4 537 567 650	24 310	2 268 771 670	0.4999973213
19	17 672 631 900	48 620	8 836 291 640	0.4999986244
20	68 923 264 410	92 378	34 461 586 016	0.4999993298
21	269 128 937 220	184 756	134 564 376 232	0.4999996568
22	1 052 049 481 860	352 716	526 024 564 572	0.4999998324
23	4 116 715 363 800	705 432	2 058 357 329 184	0.4999999143
24	16 123 801 841 550	1 352 078	8 061 900 244 736	0.4999999581
25	63 205 303 218 876	2 704 156	31 602 650 257 360	0.4999999786
26	247 959 266 474 052	5 200 300	123 979 630 636 876	0.4999999891
27	973 469 712 824 056	10 400 600	486 734 851 211 728	0.4999999947

15. táblázat. Csúcsok n, nemnövekvő szabályos sorozatok $R(n)$, félpalindrom sorozatok $P(n)$, erős sorozatok $S(n)$ és gyenge sorozatok $W(n)$ száma, valamint az átugrott és a szabályos sorozatok számának $W(n)/R(n)$ hányadosa $1, \ldots, 27$ csúcs esetén.

sorozat középső eleme $\lfloor n/2 \rfloor$, az $\lfloor n/2 \rfloor$ kezdő elemet pedig ismétléssel választhatjuk ki $\lceil n/2 \rceil$ elem közül.

Az erős és gyenge sorozatok kölcsönösen egyértelműen megfeleltethetőek egymásnak, ezért az első egyenlőség igaz. A szabályos sorozatok halmazában háromféle sorozat van: félpalindrom, erős és gyenge. Ezért a második egyenlőség is igaz. □

A 15. táblázatban összefoglaljuk a nemnövekvő szabályos sorozatok bizonyos adatait $n = 1, 2, \ldots, 27$ csúcs esetén.

A táblázat alapján úgy tűnik, hogy a hányadosok sorozata monoton növekedve 0.5-höz tart, azaz aszimptotikusan a sorozatok felét nem kell tesztelnük. A 7.1. tétel bizonyítja, hogy ez igaz.

7.1. Tétel. *Ha n tart a végtelenbe, akkor*

$$\lim_{n\to\infty} \frac{S(n)+P(n)}{R(n)} = \frac{1}{2}\,. \tag{12}$$

Bizonyítás. Két eset van. Ha n páros, akkor a 7.5. lemma szerint

$$\frac{S(n)+P(n)}{R(n)} = \frac{R(n)/2 - P(n)/2 + P(n)}{R(n)} = \frac{R(n)+P(n)}{2R(n)} = \frac{1}{2} + \frac{P(n)}{2R(n)} =$$

$$\frac{1}{2} + \frac{(n-1)!n!(n-1)!}{2(n/2)!(n/2)!(2n-1)!}\,.$$

A Stirling-formula

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + O\left(\frac{1}{n}\right)\right)$$

alakját felhasználva

$$\frac{n^{3n-2}e^{3n-1}}{n^{3n-1}e^{3n-2}} \frac{\sqrt{2\pi(n-1)^2 2\pi n}}{\sqrt{2(n\pi)^3}} \frac{(1+O(1/(n-1))^2 O(1+1/n)}{(1+O(2/n))^2(1+O(1/(2n-1)))}\,,$$

ahonnan

$$\frac{S(n)+P(n)}{R(n)} = \frac{1}{2} + \frac{e}{n}\Theta(1),$$

amiből következik (12).

Ha n páratlan, akkor a gondolatmenet és az eredmény hasonló. □

Ha a potenciális gráfos sorozatokat lexikografikusan nemcsökkenő sorrendben teszteljük, a félpalindrom sorozatok továbbra is félpalindrom sorozatok lesznek, míg az erős és gyenge sorozatok szerepet cserélnek. Eközben a különböző típusú sorozatok száma nem változik.

7.3.2. Komplementerek átugrása páros sorozatok tesztelése során

Most vizsgáljuk meg azt az esetet, amikor a gráfos sorozatokat a páros sorozatok között keressük.

Egy nemnövekvő páros 1-sorozat van: 0 – ez félpalindrom sorozat. Két nemnövekvő páros 2-szabályos sorozat van: 1, 1 és 0, 0. Közülük az 1, 1 erős, a 0, 0 pedig gyenge. Hat 3-páros sorozat van: 2, 2, 2; 2, 2, 0; 2, 1, 1; 2, 0, 0; 1, 1, 0 és 0, 0, 0. Ezek között nincs félpalindrom, viszont 3 erős és 3 gyenge sorozat van. Tizenkilenc 4-páros sorozat van:
(0, 0, 0, 0); (0, 0, 0, 2); (0, 0, 1, 1); (0, 0, 1, 3); (0, 0, 2, 2); (0, 0, 3,,3); (0, 1, 1, 2);
(0, 1, 2, 3); (0, 2, 2, 2); (0, 2, 3, 3); (1, 1, 1, 1); (1, 1, 1, 3); (1, 1, 2, 2); (1, 1, 3, 3);
(1, 2, 2, 3); (1, 3, 3, 3); (2, 2, 2, 2); (2, 2, 3, 3) és (3, 3, 3, 3).

n	$E(n)$	$P_e(n)$	$S_e(n) = W_e(n)$	$W_e(n)/E(n)$
1	1	1	0	0.000000000000000
2	2	0	1	0.500000000000000
3	6	0	3	0.500000000000000
4	19	3	8	0.421052631578947
5	66	6	30	0.454545454545455
6	236	10	113	0.478813559322034
7	868	0	434	0.500000000000000
8	3235	35	1600	0.494590417310665
9	12190	70	6060	0.497128794093519
10	46252	126	23063	0.498637896739600
11	176484	0	88242	0.500000000000000
12	676270	462	337904	0.499658420453369
13	2600612	924	1299844	0.499822349508500
14	10030008	1716	5014146	0.499914456698340
15	38781096	0	19390548	0.500000000000000
16	150273315	6435	75133440	0.499978589012959
17	583407990	12870	291697560	0.499988969983082
18	2268795980	24310	1134385835	0.499994642532820
19	8836340260	0	4418170130	0.500000000000000
20	34461678394	92378	17230793008	0.499998659699639
21	134965161188	184756	67482488216	0.499999315541884
22	526024917288	352716	263012282286	0.499999664734513
23	2058358034616	0	1029179017308	0.500000000000000
24	8061901596814	1352078	4030950122368	0.499999916143978
25	31602652961516	2704156	15801325128680	0.499999957216313
26	123979635837176	5200300	61989815318438	0.499999979027604
27	486734861612328	0	243367430806164	0.500000000000000

16. táblázat. Csúcsok n, nemnövekvő páros sorozatok $E(n)$, páros félpalindrom sorozatok $P_e(n)$, páros erős sorozatok $S_e(n)$ és páros gyenge sorozatok $W_e(n)$ száma, valamint az átugrott sorozatok és a páros sorozatok számának $W_e(n)/E(n)$ hányadosa $1, \ldots, 27$ csúcs esetén.

Ezek közül a (3, 3, 0, 0); (3, 2, 1, 0) és (2, 2, 1, 1) félpalindrom, a többiek között pedig 16 erős és 16 gyenge sorozat van. A hatvanhat 5-szabályos sorozat közül a (4, 4, 2, 0, 0); (4, 3, 2, 1, 0); (4, 2, 2, 2, 0); (3, 3, 2, 1, 1); (3, 2, 2, 2, 1) és a (2, 2, 2, 2, 2) félpalindrom, a többieknek pedig fele erős, fele pedig gyenge.

7.6. Lemma. *Ha $n \geq 1$, akkor a nemnövekvő n-páros sorozatok között páros n és $4k+1$ alakú páratlan n esetén*

$$P(n) = \binom{n-1}{\lfloor n/2 \rfloor},$$

$4k+3$ alakú páratlan n esetén pedig

$$P(n) = 0$$

n-félpalindrom sorozat és

$$W_e(n) = S_e(n) = \frac{E_(n) - P_z(n)}{2}$$

erős és gyenge sorozat van.

Bizonyítás. Lásd [88]. □

7.3.3. Komplementerek átugrása nullamentes páros sorozatok tesztelése során

Most vizsgáljuk meg azt az esetet, amikor a gráfos sorozatokat a nullamentes páros sorozatok között keressük. Ezek kezelése bonyolultabb, mint a szabályos és páros sorozatoké volt.

Egy nullamentes gráfos sorozat izolált csúcsot tartalmazó gráfhoz tartozik. Egy nullamentes sorozatban vagy van $n-1$, vagy nincs. Az $n-1$-et tartalmazó sorozatok komplementerében van nulla, ezért az ilyen sorozatok komplementerét nem generáljuk, és ha gráfosak, csak egyszeres súllyal vesszük őket figyelembe. Az ilyen sorozatokat *félerősnek* nevezzük és számukat $M'_z(n)$-nel jelöljük az $(n-1)$-et nem tartalmazó esetben, és $M''_z(n)$-nel a másikban.

Először foglalkozzunk az $(n-1)$-et nem tartalmazó sorozatokkal. A nullamentes 1-páros és a 2-páros sorozatok között nincs megfelelő sorozat. A tizenkilenc 4-páros sorozat közül most csak 3 felel meg: 2, 2, 2, 2; 2, 2, 1, 1 és 1, 1, 1, 1. Közülük a 2, 2, 1, 1 félpalindrom, a 2, 2, 2, 2 erős és az 1, 1, 1, 1 a gyenge párja, azaz $P'_z(4) = 1$ és $S'_z(4) = W'_z(4) = 1$. A hatvanhat 5-páros sorozat közül most csak a következő 9 felel meg:
(3, 3, 3, 3, 2); (3, 3, 3, 2, 1); (3, 3, 2, 2, 2); (3, 3, 2, 1, 1); (3, 2, 2, 2, 1); (3, 2, 1, 1, 1);
(2, 2, 2, 2, 2); (2, 2, 2, 1, 1); (2, 1, 1, 1, 1).

Ezek közül a 3, 3, 2, 1, 1; 3, 2, 2, 2, 1 és 2, 2, 2, 2, 2 a félpalindrom, és van további 3 erős és 3 gyenge.

A 236 páros 6-sorozat közül a következő ötven 0-mentes és ugyanakkor 5-mentes is:
(4, 4, 4, 4, 4, 4); (4, 4, 4, 4, 4, 2); (4, 4, 4, 4, 3, 3); (4, 4, 4, 4, 3, 1); (4, 4, 4, 4, 2, 2);
(4, 4, 4, 4, 1, 1); (4, 4, 4, 3, 3, 2); (4, 4, 4, 3, 2, 1); (4, 4, 4, 2, 2, 2); (4, 4, 4, 2, 1, 1);
(4, 4, 3, 3, 3, 3); (4, 4, 3, 3, 3, 1); (4, 4, 3, 3, 2, 2); (4, 4, 3, 3, 1, 1); (4, 4, 3, 3, 2, 1);
(4, 4, 3, 2, 2, 1); (4, 4, 3, 1, 1, 1); (4, 4, 2, 2, 2, 2); (4, 4, 2, 2, 1, 1); (4, 4, 1, 1, 1, 1);
(4, 3, 3, 3, 3, 2); (4, 3, 3, 3, 3, 1); (4, 3, 3, 3, 2, 2); (4, 3, 3, 3, 2, 1); (4, 3, 3, 2, 2, 2);
(4, 3, 3, 2, 1, 1); (4, 3, 2, 2, 2, 1); (4, 3, 2, 1, 1, 1); (4, 2, 2, 2, 2, 2); (4, 2, 2, 2, 1, 1);
(4, 2, 1, 1, 1, 1); (3, 3, 3, 3, 3, 3); (3, 3, 3, 3, 3, 1); (3, 3, 3, 3, 2, 2); (3, 3, 3, 3, 1, 1);
(3, 3, 3, 3, 2, 2); (3, 3, 3, 3, 2, 2); (3, 3, 3, 3, 1, 1); (3, 3, 3, 2, 2, 1); (3, 3, 3, 1, 1, 1);
(3, 3, 2, 2, 2, 2); (3, 3, 2, 2, 1, 1); (3, 3, 1, 1, 1, 1); (3, 2, 2, 2, 2, 1); (3, 2, 2, 1, 1, 1);
(3, 1, 1, 1, 1, 1); (2, 2, 2, 2, 2, 2); (2, 2, 2, 2, 1, 1); (2, 2, 1, 1, 1, 1); (1, 1, 1, 1, 1, 1).

Ezek között nincs félpalindrom, és 25 erős, valamint 25 gyenge van.

A 17. táblázatban összefoglaljuk a nemnövekvő nullamentes és $(n-1)$-mentes páros sorozatok bizonyos adatait.

n	$E_z(n)$	$E'_z(n)$	$P'_z(n)$	$S'_z(n) = W'_z(n)$	$\frac{W'_z(n)}{E_z(n)}$
1	0	0	0	0	————
2	1	0	0	0	0.00000
3	2	0	0	0	0.00000
4	9	3	1	1	0.11111
5	28	9	3	3	0.10714
6	110	44	0	22	0.20000
7	396	160	0	80	0.20202
8	1 519	651	15	318	0.20934
9	5 720	2 485	35	1 225	0.21416
10	21 942	9 752	0	4 876	0.22222
11	83 980	37 728	0	18 864	0.22462
12	323 554	147 070	210	73 430	0.22695
13	1 248 072	571 802	462	285 670	0.22889
14	4 829 708	2 229 096	0	1 114 548	0.23077
15	18 721 080	8 691 072	0	4 345 536	0.23212
16	72 714 555	33 933 459	3 003	16 965 228	0.23331
17	282 861 360	132 588 045	6 435	66 290 805	0.23436
18	1 101 992 870	518 584 880	0	259 292 440	0.23529
19	4 298 748 300	2 029 952 320	0	1 014 976 160	0.23611
20	16 789 046 494	7 952 706 234	43 758	3 976 331 238	0.23684
21	65 641 204 200	31 179 525 806	92 378	15 589 716 714	0.23750
22	256 895 980 068	122 331 419 080	0	61 165 709 540	0.23810
23	1 006 308 200 040	480 283 282 752	0	240 141 641 376	0.23864
24	3 945 186 233 014	1 886 828 198 398	646 646	943 413 775 876	0.23913
25	15 478 849 767 888	7 416 948 171 074	1 352 078	3 708 473 409 498	0.23958
26	60 774 332 618 300	29 171 679 656 784	0	14 585 839 828 392	0.24000
27	238 775 589 937 976	114 795 954 100 800	0	57 397 977 050 400	0.24038
28	938 702 947 395 204	451 968 085 782 876	9 657 700	225 984 038 062 588	0.24074

17. táblázat. Csúcsok n, nemnövekvő nullamentes páros sorozatok $E_z(n)$, nullamentes, páros, $(n-1)$-et nem tartalmazó sorozatok $E'_z(n)$, félpalindrom sorozatok $P'_z(n)$, erős sorozatok $S'_z(n)$ és gyenge sorozatok $W'_z(n)$ száma, valamint a tesztelés nélkül átugrott sorozatok és a nullamentes páros sorozatok számának $W'_z(n)/E'_z(n)$ hányadosa $1, \ldots, 28$ csúcs esetén.

A táblázatban lévő adatokat a következő képletekkel számítottuk.

A nullamentes páros sorozatok $E_z(n)$ számát megadja az (5.2) képlet. A következő 7.7. lemma alapján számítottuk $E'_z(n)$ értékét.

7.7. Lemma. *Ha $n \geq 1$, akkor*

$$E'_z(n) = \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{\lceil (n-2)/2 \rceil - 1 + 2i}{2i} \cdot \binom{\lfloor (n-2)/2 \rfloor - 1 + n - 2i}{n-2i}.$$

Bizonyítás. Mivel páros sorozatokat vizsgálunk, a bennük lévő páratlan elemek száma $0, 2, \ldots, \lfloor n/2 \rfloor$ lehet. A sorozatok lehetséges elemei most $1, 2, \ldots, n-2$, ezek között $\lceil (n-2)/2 \rceil$ páratlan és $\lfloor (n-2)/2 \rfloor$ páros elem van, amelyek közül $0, 2, \ldots, \lfloor n/2 \rfloor$ páratlan elemet kell kiválasztanunk, a maradék helyeket pedig páros elemekkel kell feltöltenünk. □

A félpalindrom, erős, gyenge és félerős sorozatok számát ebben az esetben a 7.8. lemma adja meg.

7.8. Lemma. *a) Ha $k \geq 0$ és $n = 4k+2$, akkor*

$$P'_z(n) = 0.$$

b) Ha $k \geq 0$ és $n = 4k$, akkor

$$P'_z(n) = \binom{(n-2)/2 - 1 + n/2}{n/2}.$$

c) Ha $k = 4k+3$, akkor

$$P'_z(n) = 0.$$

d) Ha $n = 4k+1$, akkor

$$P'_z(n) = \binom{(n-1)/2 - 1 + (n-1)/2}{(n-1)/2} = \binom{n-2}{(n-1)/2}.$$

e) Ha $n \geq 1$, akkor

$$S'_z(n) = W'_z(n) = \frac{E'_z(n) - P'_z(n)}{2}.$$

f) Ha $n \geq 1$, akkor

$$S''_z(n) = W''_z(n) = \frac{E'_z(n)}{2}.$$

Bizonyítás. a) Mivel $n = 4k+2$, egy félpalindrom sorozatban $n/2 = 2k+1$ szimmetrikus elempár lenne, melyek elemei összeadva $(n-1 = 4k+1)$-et adnak, így a sorozat elemeinek összege $(2k+1)(4k+1)$ lenne, ami páratlan szám.

b) Ha $n = 4k$, akkor $n/2$ szimmetrikus pár van, és ha mindegyikben az előírt $4k-1$ az elemek összege, a sorozat páros. Az első $n/2$ elem az $[1, \lfloor (n-1)/2 \rfloor] = [1, (n-2)/2]$ intervallumba tartozik és innen kell ismétléssel $n/2$ elemet kiválasztani.

c) Ha $n = 4k + 3$, akkor a kételemű szimmetrikus párok elemeinek összege $4k + 2$, viszont a középső elemnek a $2k + 1$ értéket kellene felvennie, ami páratlan szám, és így a sorozat elemeinek összege is páratlan lenne.

d) A középső elemnek a páros $2k = (n - 1)/2$ értéket kell felvennie, ezért a szóba jövő sorozatok elemeinek összege páros. Az első $(n - 1)/2$ elemet az $[1, (n - 1)/2]$ intervallum elemei közül kell ismétléssel kiválasztanunk és ezek az elemek egyértelműen meghatározzák az utolsó $(n - 1)/2$ elemet.

e) A palindrom sorozatokon kívül csak erős és gyenge sorozatok vannak, amelyek kölcsönösen egyértelműen megfeleltethetőek egymásnak.

f) Ebben az esetben minden erős sorozatnak van gyenge párja, palindrom és félerős sorozat nincs. □

Most nézzük az $(n - 1)$-et tartalmazó sorozatokat. $E''_z(n) = E_z(n) - E'_z(n)$ ilyen sorozat van. Mivel $n - 1$ eleme a vizsgált sorozatoknak, viszont a nulla nem, ezért itt félpalindrom sorozat nincs, továbbá erős, valamint gyenge sorozat sincs – minden sorozat félerős, ezért mindegyiket tesztelni kell.

Korábbi szimulációs eredményeink [90] szerint aszimptotikusan a páros sorozatok fele nullamentes. A 17. táblázat adatai pedig alátámasztják azt a sejtést, hogy aszimptotikusan a nullamentes páros sorozatok negyede gyenge, azaz átugorható, ezért ettől az algoritmustól a futási idő közel 25 százalékos csökkenését várjuk.

8. Összefoglalás

A naplófájlok és programok forráskódja megtalálható a
http://people.inf.elte.hu/lulsaai/Holzhacker és
http://people.inf.elte.hu/tomintt/DegreeSeq
címeken [120, 128].

Köszönetnyilvánítás.

A szerzők köszönik Lucz Lorándnak, Matuszka Tamásnak és Szabados Kristófnak a számítógépes szimulációhoz nyújtott segítséget.

Hivatkozások

[1] Acosta, P., Bassa, A., Chaikin, A., Riehl, A., Tingstad, A. Zhao, L., Kleitman, D. J.: *On a conjecture of Brualdi and Shen on block transitive tournaments*, Journal of Graph Theory, **44(3)**, (2003) 215–230.

[2] Anisiu, M.-C., Anisiu, V., Kása, Z.: *Total palindrome complexity of finite words*, Discrete Mathematics, **310**, (2010) 109–114.

[3] Arikati, S. R., Maheshwari, A.: *Realizing degree sequences in parallel*, SIAM Journal on Discrete Mathematics, **9(2)**, (1996) 317–338.

[4] Ao, S., Hanson, D.: *Score vectors and tournaments with cyclic chromatic number 1 or 2*, Ars Combinatoria, **49**, (1998) 185–191.

[5] Ascher, M.: *Mu torere: An analysis of a Maori game*, Mathematical Magazine, **60(2)**, (1987,) 90–100.

[6] Avery, P.: *Score sequences of oriented graphs*, Journal of Graph Theory, **15**, (1971) 251–257.

[7] Avery, P.: *Condition for a tournament score sequence to be simple*, Journal of Graph Theory, **4(2)**, (1980) 157–164.

[8] Bagga, K. S., Beineke, L. W., Harary, F.: *Two problems on coloring tournaments*, Vishwa Internat Journal of Graph Theory, **1(1)**, (1992) 83–94.

[9] Bang, C. M., Sharp, H.: *An elementary proof of Moon's theorem on generalized tournaments*, Journal of Combinatorial Theory, Series B, **5**, (1977) 299–301.

[10] Bang, C. M., Sharp, H., Jr.: *Score vectors of tournaments*, Journal of Combinatorial Theory, Series B, **26(1)**, (1979) 81–84.

[11] Bang-Jensen, J, Gutin, G.: *Generalizations of tournaments – a survey*, Journal of Graph Theory, **28**, (1998) 178–202.

[12] Bang-Jensen, J., Bessy, S.: *Arc-disjoint flows in networks*, Theoretical Computer Science, **526**, (2014), 28–40.

[13] Bang-Jensen, J., Huang, J., Prisner, E.: *In-tournament digraphs*, Journal of Combinatorial Theory, Ser. B, **59(2)**, (1993) 267–287.

[14] Barnes, T. M., Savage C. D.: *A recurrence for counting graphical partitions*, Electronic Journal of Combinatorics, **2**, (1995) #P11, 10 oldal.

[15] Barnes, T. M., Savage, C. D.: *Efficient generation of graphical partitions.* Discrete Applied Mathematics, **78(1–3)**, (1997) 17–26.

[16] Barrus, M. D.: *Hereditary unigraphs and Erdős-Gallai inequalities*, Discrete Matematics **313(21)**, (2013) 2469–2481.

[17] Barrus, M. D., Hartke, S. G., Jao, K. F., West, D. B.: *Length thresholds for graphic lists given fixed largest and smallest entries and bounded gaps*, Discrete Mathematics, **312(9)**, (2012) 1294–1501.

[18] Beasley, L. B., Brown D. E., Reid, K. B.: *Extending partial tournaments.* Mathematical Computer Modelling **50(1)**, (2009) 287–291.

[19] Bege, A., Kása, Z.: *Algoritmikus kombinatorika és számelmélet*, Presa Universittară Clujeană, 2006, 215 oldal.

[20] Behrens, S., Erbes, C., Ferrara, M., Hartke, H. S., Reiniger, B., Spinoza, H., Thomlinson, C.: New results on degree sequences of uniform hypergraphs, *Electron. J. Comb.*, **20(4)**, (2013) #P14, 18 oldal.

[21] Beineke, L. W., Reid, K. B.: *Tournaments*, in: Selected Topics in Graph Theory, Academic Press (Harcourt Brace Jovanovich Publishers), London, 1978, 169–204.

[22] Bell, J. P., Bender, E. A., Cameron, P. J., Richmond, L. B.: *Asymptotics for the probability of connectedness and the distribution of number of components*, Electronic Journal of Combinatorics, **7** (2000), #R33, 33 oldal.

[23] Bender, E. A., Canfield, E. R.: *The asymptotic number of labeled graphs with given degree sequences*. Journal of Combinatorial Theory, Series A **24(3)**, (1978) 296–307.

[24] Bender, E. A., Canfield, E. R, McKay, B. D.: *The asymptotic number of labeled connected graphs with a given number of vertices and edges*. Random Structures & Algorithms, **1(2)**, (1990) 127–170.

[25] Bender, E. A., Canfield, E. R, McKay, B. D.: *Asymptotic properties of labeled connected graphs*. Random Structures & Algorithms, **3(2)**, (1992) 183–202.

[26] Bender, E. A., Gao, Z.: *Asymptotic enumeration of labelled graphs by genus*, Electronic Journal of Combinatorics, **18(1)**, (2011) #P13, 28 oldal.

[27] Berge, C.: *Graphs and Hypergraphs*, North Holland, 1973.

[28] Berge, C.: *Graphs* (harmadik kiadás), North Holland, 2001 (első kiadás: 1989).

[29] Berger, A.: *A note on the characterization of digraph sequences*, arXiv, arXiv:1112.1215v1 [math.CO] (6 December 2011), 6 oldal.

[30] Berger, A., Müller-Hannemann, M.: *Uniform sampling of digraphs with a fixed degree sequence*, in: (szerk. D. M. Thilikos) WG2010, LNCS **6410**, (2010) 220–231.

[31] Berger, A., Müller-Hannemann, M.: *How to attack the NP-complete dag realization problems in practice*, arXiv, arXiv:1203.36v1, 2012, 20 oldal.

[32] Bödei, N.: *Gráfok foksorozatai*, Diplomamunka (témavezető Frank András), ELTE Operációkutatási Tanszék, Budapest, 2010, 43 oldal.

[33] Bollobás, B.: *The distribution of the maximum degree of a random graph*, *DiscreteMathematics*, **32(2)**, (1980) 201–203.

[34] Bollobás, B.: *A probabilistic proof of an asymptotic formula for the number of labelled regular graphs*, European Journal of Combinatorics, **1,(4)** 4 (1980) 311–316.

[35] B. Bollobás: *Degree sequences of random graphs*, *Discrete Mathematics*, **33(1)**, 1 (1981) 1–19.

[36] Brualdi, A. R., Fristcher, E.: *Tournaments associated with multigraphs*, Discrete Mathematics, 2014, 17 oldal **526(2)**, (2014) 28–40.

[37] Brualdi, A. R, Kiernan, K.: *Landau's and Rado's theorems and partial tournaments*, Electronic Journal of Combinatorics, **16**, #N2, 2009, 6 oldal.

[38] Brualdi, A. R., Li, Q.: *The interchange graph of tournaments with the same score vector*. in: Progress in Graph Theory (Waterloo, ON, 1982), Academic Press, Toronto, ON, 1984, 129–151.

[39] Brualdi, R. A., Michael, T. S.: *The class of 2-multigraphs with a prescribed degree sequence*, Linear and Multilinear Algebra, **24(2)**, (1989) 81–102.

[40] Brualdi, A. R., Shen, J: *Landau's inequalities for tournament scores and a short proof of a theorem on transitive sub-tournaments*, Journal of Graph Theory, **38**, (2001) 244–254.

[41] Brunson, J. C.: The S-metric, the Beichl-Croteaux approximation and preferential attachment, *arXiv*:1308.4067v1 [mathCO] 19 August 2013, 14 oldal.

[42] Burns, J. M.: *The number of degree sequences of graphs*, PhD disszertáció, The MIT, 2007, 62 oldal.

[43] Busch, A. H, Chen, G., Jacobson, M. S.: *Transitive partitions in realizations of tournament score sequences*, Journal of Graph Theory, **64(1)**, (2010) 52–62.

[44] Busch, A. H., Ferrara, M. J., Hartke, S. G., Jacobson, M. S., Kaul, H., Packing of graphic n-tuples, *Journal of Graph Theory*, **70**, (2012) 29–39.

[45] Choudum, S. A., *A simple proof of the Erdős-Gallai theorem on graph sequences*, Bulletin of the Australasian Mathematical Society, **33**, (1986) 67–70.

[46] Chungphaisan, V.: *Conditions for sequences to be r-graphic*, Discrete Mathematics, **7**, (1974) 31–39.

[47] Codish, M., Miller, A., Prosser, P.: *Breaking symmetries in graph representation*, in: Proceedings of Twenty Third Int. Join Conf. on Art. Int. (IJCAI), 2013, 510–516.

[48] Cohen, N.: *Number of distinct degree sequences among all n-vertex graphs with no isolated vertices: new values for* $n = 20, 21, 22,$ *and* 23, in: ed. by N. J. A. Sloane, The On-Line Encyclopedia of Integer Sequences, 2012, `http://oeis.org/A095268`.

[49] Cormen, T. H., Leiserson, Ch. E., Rivest, R. L., Stein, C.: *Introduction to Algorithms* (harmadik kiadás), The Press McGraw Hill, Cambridge/New York, 2009.

[50] Cruse, A. B.: *On linear programming duality and Landau's characterization of tournaments scores*, Manuscript, based on the talk presented on the 57th Annual Meeting of the American Mathematical Society, Souteasternn Section, held March 31–April 1, 1978, at Clemson University, Clemson, SC, 18 oldal.

[51] Cruse, A. B.: *On linear programming duality and Landau's characterization of tournaments scores*, Acta Universitatis Sapientiae, Informatica **6(1)**, (2014) 21–32.

[52] Daniels, H. E.: *Round-robin tournament scores*, Biometrika, **56** (1969) 295–299.

[53] Del Genio, C. I., Kim, H., Toroczkai, Z., Bassler, K. E: *Efficient and exact sampling of simple graphs with given arbitrary degree sequence.* PLoS ONE **5(4)**, e10012 (2010), 7 oldal.

[54] Dimitrov, D., *Efficient computation of trees with minimal atom-bound connectivity index*,. arXiv:1305.1155v2 [csDM] 4 October 2013, 13 oldal.

[55] Entringer, R. C., Tolman, L. K.: *Characterizations of graphs having orientations satisfying local degree restrictions*, Czechoslovak Mathematical Journal, **28(1)**, (1978) 108–119.

[56] Erdős, P., Gallai, T.: *Gráfok előírt fokszámú pontokkal*, Matematikai Lapok, **11**, (1960) 264–274.

[57] Erdős, P., Richmond L. B.: *On graphical partitions.* Combinatorica **13(1)**, (1993) 57–63.

[58] Erdős, P., Moon, P. J. W.: *On sets of consistent arcs in a tournament.* Canadian Mathematical Bulletin, **8**, (1965) 269–271.

[59] Erdős, L. P., Király, Z., Miklós, I.: *On the swap-distances of different realizations of a graphical degree sequence*, arXiv, arXiv:1205.2842v1 [math.CO] 13 May 2012, 20 oldal.

[60] Erdős, L. P., Kiss, S. Z., Miklós, I.: *On the swap-distances of different realizations of a graphical degree sequence*, Combinatorial Probability and Computations, **22(3)**, (2013) 366–383.

[61] Erdős, L. P.: Király, Z., Miklós, I., Soukup, L.: *Constructive sampling and counting graphical realizations of restricted degree sequences*, *arXiv* arXiv:13017523v3[math.CO], 24 oldal.

[62] Erdős, L. P.: Miklós, I., Toroczkai, Z.: *A simple Havel-Hakimi type algorithm to realize graphical degree sequences of directed graphs*, Electronic Journal of Combinatorics, **17(1)**, (2010), R66, 10 oldal.

[63] Erdős, L. P., Miklós, I., Toroczkai, Z.: *A decomposition based proof for fast mixing of a Markov chain over balanced realizations of a joint degree matrix*, *arXiv*, arXiv:1307.5295v1 [math.CO], 2013, 18 oldal.

[64] Ford, L. R., Selmer, S. M.: *A tournament problem*, The American Mathematical Monthly, **66(5)**, (1959) 387–389.

[65] Frank, A.: *Connections in Combinatorial Optimization,* Oxford University Press, Oxford, 2011.

[66] Garg, A., Goel, A., Tripathi, A.: *Constructive extensions of two results on graph sequences.* Discrete Applied Mathematics, **159(17)**, (2011) 2170–2174.

[67] Gervacio, S. V.: *Score sequences: lexicographic enumeration and tournament construction*, Discrete Mathematics, **72(1–3)**, (1988) 151–155.

[68] Gervacio, S. V.: *Construction of tournaments with a given score sequence*, Southeast Asian Bulletin of Mathematics, **17(2)**, (1993) 151–155.

[69] Graham, R. L., Spencer, J. H.: *A constructive solution to a tournament problem*, Canadian Mathematical Bulletin, **14** (1971) 45–48.

[70] Griggs, J. R., Reid, K. B.: *Landau's theorem revisited*, Australasian Journal of Combinatorics, **20**, (1999) 19–24.

[71] Gross, J. L., Yellen, J., Zhang, P.: *Handbook of Graph Theory*, CRC Press, Boca Raton, FL, 2013, XIX + 1610 oldal.

[72] Guiduli, B., Gyárfás, A., Thomassé, S., Weidl, P.: *2-partition-transitive tournaments*, Journal of Combinatorial Theory, Series B, **72(2)**, (1998), 181–196.

[73] Gyárfás, A.: *Transitive tournaments and self-complementary graphs*, Journal of Combinatorial Theory, Series B, **38(2)**, (1998) 111–112.

[74] Hakimi, S. L.: *On the realizability of a set of integers as degrees of the vertices of a simple graph.* Journal of SIAM Applied Mathematics, **10**, (1962) 496–506.

[75] Hakimi, S. L.: *On the degrees of the vertices of a graph*, Journal of the Franklin Institute, **279(4)**, (1965) 290–308.

[76] Harary, F., Moser, L.: *The theory of round robin tournaments*, The American Mathematical Monthly, **73**, (1966) 231–246.

[77] Harary, F., Palmer, E. M.: *Graphical Enumeration*, Academic Press, New York and London, 1973, 271 oldal.

[78] Havel, V.: A remark on the existence of finite graphs (cseh), *Časopis Pěstovany Matematica* **80**, (1955) 477–480.

[79] Hell, P., Kirkpatrick, D.: *Linear-time certifying algorithms for near-graphical sequences*, Discrete Mathematics **309(18)**, (2009) 5703–5713.

[80] Hemasinha, R.: *An algorithm to generate tournament score sequences*, Mathematical and Computer Modelling, **37(1–3)**, (2003), 377–382.

[81] Holshouser, A., Moon, J. W., Reiter, H.: *Win-loss sequences for generalized roundrobin tournaments*, *Missouri Journal of Mathematical Science*, **23(2)**, (2011), 103–207.

[82] G. Isaak, D. B. West, The edge-count criterion for graphic lists, *Electronic Journal of Combinatorics*, **17** (2010) N#36, 5 oldal.

[83] Iványi, A.: *Versenyek pontsorozatai*, in: 25. Magyar Operációkutatási Konferencia (Debrecen, 2001. október 17–20), 52–52.

[84] Iványi, A.: *Reconstruction of complete interval tournaments*, Acta Universitatis Sapientiae, Informatica, **1(1)**, (2009) 71–88.

[85] Iványi, A., *Reconstruction of complete interval tournaments II*, *Acta Univ. Sapientiae, Mathematica*, **2(1)**, (2010) 47–71.

[86] Iványi, A.: *Deciding the validity of the score sequence of a soccer tournament*, in (ed. A. Frank): Open problems of the Egerváry Research Group, Budapest, 2012. `http://lemon.cs.elte.hu/egres/open/`.

[87] Iványi, A., *Degree sequences of multigraphs*, Annales Universitatis Scientiarum de Rolando Eötvös Nominatae, Section Computatorica **37**, (2012) 195–214.

[88] Iványi, A., Elek J.: *Score sets in multitournaments II. Simulation results*, **LIX(1)**, (2014) 150–164.

[89] Iványi, A., Lucz, L., *Multigráfok foksorozatai*, Alkalmazott Matematikai Lapok, **29**, (2012) 1–54.

[90] Iványi, A., Lucz, L., Gombos, G., Matuszka, T.: *Parallel enumeration of degree sequences of simple graphs II*, Acta Universitatis Sapientiae, Informatica **5(2)**, (2013) 245–270.

[91] Iványi, A., Lucz, L., Matuszka, Pirzada, S.: *On the Erdős-Gallai and Havel-Hakimi algorithms*, Acta Universitatis Sapientiae, Informatica **3(2)**, (2011) 230–268.

[92] Iványi, A., Lucz, L., Gombos, G., Matuszka, T.: $C(2n+1, n+1)$*: number of ways to put* $n+1$ *indistinguishable balls into* $n+1$ *distinguishable boxes = number of* $(n+1)$*-st degree monomials in* $n+1$ *variables = number of monotone maps from* $1\ .\ .\ n+1$ *to* $1\ .\ .\ n+1$, in (ed. N. J. A. Sloane): The On-Line Encyclopedia of the Integer Sequences, 2013. `http://oeis.org/A001700`.

[93] Iványi, A., Lucz, L., Gombos, G., Matuszka, T.: *The number of degree-vectors for simple graph.* in (ed. N. J. A. Sloane): The On-Line Encyclopedia of the Integer Sequences, 2013. `http://oeis.org/A004251`.

[94] Iványi, A., Lucz, L., Matuszka, T., Pirzada, S.: *Parallel enumeration of degree sequences of simple graphs*, Acta Universitatis Sapientiae, Informatica, **4(2)**, (2012) 260–288.

[95] Iványi, A., Lucz, L., Móri, T. F., Sótér, P., *On the Erdős-Gallai and Havel-Hakimi algorithms*, Acta Universitatis Sapientiae, Informatica **3(2)**, (2011) 230–268.

[96] Iványi, A., Lucz, L., Móri F. T., Sótér, P.: *The number of degree-vectors for simple graphs*, in: ed. by N. J. A. Sloane, The On-Line Encyclopedia of Integer Sequences, 2011. `http://oeis.org/A004251`.

[97] Iványi, A., Pirzada, S.: *Comparison based ranking*, in: Algorithms of Informatics, Vol. 3 (ed. A. Iványi), AnTonCom, Budapest 2011, 1209–1258.

[98] Iványi, A., Pirzada, Shah, N. A.: *Imbalances of bipartite multitournaments*, *Annales Univ. Sci. Budapest., Sect. Comput.* **37** (2012), 215–228.

[99] Iványi, A., Schoenfield, J. E.: *Deciding football sequences.* Acta Universitatis Sapientiae, Informatica **4(1)**, (2012) 130–183.

[100] Johnson, R. H.: *Properties of unique realizations—a survey*, Discrete Mathematics, **3(1)**, (1980) 185–192.

[101] Kása, Z., Anisiu, M.-C.: *Szavak bonyolultsága*, in: Informatikai algoritmusok, 3. kötet, Mondat Kft., Vác, 2013, 1697–1739.

[102] Kása, Z., Anisiu, M.-C, Anisiu, V.: *Properties of palindromes in finite words* Pure Mathematics and Applications, **17(3-4)**, (2006) 183–195.

[103] Kemnitz, A., Dulff, S.: *Score sequences of multitournaments*, Congressus Numerantium, **127**, (1997) 85–95.

[104] Khan, M. A.: Equal sum sequences and imbalance sets of tournaments, Arxiv, arxiv:14022.2456v1 math.CO 11 February 2014, 18 oldal.

[105] Kim J. H., Pittel, B.: *Confirming the Kleitman-Winston conjecture on the largest coefficient in a q-Catalan number*, Journal Combinatorial Theory, Series A, **99(2)**, (2000) 197–206.

[106] Kim, J. H., Toroczkai, Z., Miklós, I., Erdős, P. L., Székely, L. A.: *Degree-based graph construction.* Journal of Physics: Mathematical Theory A **42(39)**, (2009) 392–401.

[107] Király, Z.: *Recognizing graphic degree sequences and generating all realizations*, Egres Technical Reports, TR-2011-11, April 23, 2012, 12 oldal.

[108] Király, Z., Szigeti, Z.: *Simultaneous well-balanced orientations of graphs*, J. Combin. Theory Ser. B 96 (2006) 684–692.

[109] Kleitman, J. *The number of score sequences for a large number of players*, in: Combinatorial Structures and Their Applications (ed. R. K. Guy et al.), Gordon and Breach, New York (1970), 209–213.

[110] Kleitman, D. J., Wang, D. L.: *Algorithms for constructing graphs and digraphs with given valencies and factors.* Discrete Mathematics **6**, (1973) 79–88.

[111] Kleitman, D. J., Winston K. J.: *Forests and score vectors*, Combinatorica **1(1)**, (1981) 49–54.

[112] Knuth, D. E.: *The Art of Computer Programming. Volume 4A, Combinatorial Algorithms*, Addison-Wesley, Upper Saddle River, 2011, XVI + 883 oldal.

[113] Kohnert, A.: *Dominance order and graphical partitions.* Electronic Journal of Combinatorics, **11(1)**, (2004) No. 4. 17 oldal.

[114] Koren, M.: *Sequences with a unique realization by simple graphs*, Journal of Combinatorial Theory, Series B, **21(3)**, (1976) 235–244.

[115] Kovács, G. Zs., Pataki, N.: *Foksorozatok elemzése*, TDK dolgozat, ELTE, Informatikai Kar, Budapest 2002. 38 oldal.

[116] LaMar, M. D.: *Algorithms for realizing degree sequences of directed graphs*. arXiv, 2010. `http://arxiv.org/abs/0906.0343`.

[117] Landau, L. G., *On dominance relations and the structure of animal societies. III. The condition for a score sequence*, Bulletin of Mathematics of Biophysycs **15**, (1953) 143–148.

[118] Librandi, V.: *Number of bracelets (turn over necklaces) with n red,* 1 *pink and* $n-1$ *blue beads for* $n = 1, \ldots, 1000$, in (szerk. N. J. A. Sloane): The On-Line Encyclopedia of the Integer Sequences. 2012. `http://oeis.org/A005654/b005654.txt`.

[119] Lu, X, Bressan, S.: Generating random graph sequences, in (ed. J. X. Yu, M. H. Kim, R. Unland): *DASFAA2011, Part I*, LNCS **6587**, Springer-Verlag, Berlin/Heidelberg, 2011, 570–579.

[120] Lucz, L.: *Párhuzamos Erdős-Gallai algoritmus*. TDK dolgozat, ELTE IK, Budapest (2011). Elérhető: http://people.inf.elte.hu/lulsaai/Holzhacker/TDK/.

[121] Lucz, L.: *Gráfok foksorozatainak elemzése*, Programtervező informatikus diplomamunka, ELTE IK, Budapest, 2012. Elérhető: `http://people.inf.elte.hu/lulsaai/diploma`.

[122] Lucz, L.: *Párhuzamos Erdős-Gallai algoritmus*, TDK dolgozat, ELTE Informatikai Kar, Budapest, 2013. 31 oldal.

[123] Lucz, L.: *Football league numbers: the possible point series for a league of n teams playing each other twice*. OEIS, A064422 számú sorozat. Elérhető: http://oeis.org/A064422.

[124] Lucz, L.: *Football league numbers with distinct point totals*. OEISA209467 számú sorozat, Elérhető: http://oeis.org/A209467.

[125] Lucz, L., Sótér, P.: *Foksorozatokat ellenőrző algoritmusok.* TDK dolgozat. ELTE IK, Budapest, 2011. Elérhető: `http://people.inf.elte.hu/lulsaai/Holzhacker/TDK/`.

[126] Mahmoodian, E. S., *A critical case method of proof in combinatorial mathematics*, Bulletin of the Iranian Mathematical Society, **8**, (1978) 1L–26L.

[127] Matuszka, T.: *Foksorozatokkal kapcsolatos programok és eredmények*, 2012. `http://people.inf.elte.hu/tomintt/DegreeSeq`.

[128] Matuszka: Programs and Results Connected with Degree Sequences. ELTE IK, Budapest, 2013. Elérhető: http://people.inf.elte.hu/tomintt/DegreeSeq.

[129] McKay, B. D., Wang, X.: *Asymptotic enumeration of tournaments with a given score sequence*, Journal of Combinatorial Theory, Series A, **73(1)**, (1996) 77–90.

[130] Meierling, D., Volkmann, L.: *A remark on degree sequences of multigraphs.* Mathematical Methods of Operations Research **69(2)**, (2009) 369–374.

[131] Metropolis, N., Stein, P. R.: *The enumeration of graphical partitions.* European Journal of Combinatorics, **1(2)**, (1980) 139–153.

[132] Miklós, I., Erdős, P. L., Soukup, L., *Towards random uniform sampling of bipartite graphs with given degree sequence*, Electronic Journal of Combinatorics **20(1)**, (2013), Article 16, 51 oldal.

[133] Miklós, I., Podani, J.: *Randomization of presence-absence matrices: comments and new algorithms*, Ecology, **85(1)**, (2004) 86–92.

[134] Miller, J. W. *Reduced criteria for degree sequences*, Discrete Mathematics, **313(4)** (2013) 550–562.

[135] Miller, A., Prosser, P.: *Diamond free degree sequences*, Acta Universitatis Sapientiae, Informatica, **4(2)**, (2012) 189–200.

[136] Milo, R., Kashtan, N., Itzkovitz, S., Newman, M. E. J., Alon, U.: *On the uniform generation of random graphs with prescribed degree sequences*, *arXiv,* arXiv:cond-mat/0312028v2 [cond-mat.stat-mech], 2004, 4 oldal.

[137] Moon, J. W.: *On the score sequence of an n-partite tournament*, Canadian Mathematical Bulletin, **5**, (1962) 51–58.

[138] Moon, J. W.: *An extension of Landau's theorem on tournaments*, Pacific Journal of Mathematics, **13**, (1963) 1343–1345.

[139] Moon, J. W.: *Topics on Tournaments*, Holt, Rinehart and Wilson, New York, 1968, 104 oldal.

[140] Moon, J. W., Moser, L.: *Almost all tournaments are irreducible*, Canadian Mathematical Bulletin, **5**, (1962) 61–65.

[141] Narayana, T. V., Bent, D. H.: *Computation of the number of score sequences in round-robin tournaments*, Canadian Mathematical Bulletin **7(1)**, (1964) 133–136.

[142] Narayana, T. V., Matsen, R. M., Sarangi, J.: *An algorithm for generating partitions and its applications*, Journal of Combinatorial Theory, **11**, (1971) 54–61.

[143] Noe, T. D.: *Table of $a(n)$ for $n = 1, \ldots, 100$*, in (ed. N. J. A. Sloane): The On-Line Encyclopedia of the Integer Sequences. 2010. `http://oeis.org/A001700`.

[144] Noe, T. D.: *Table of binomial coefficients $C(2n-1, n)$*, in (ed. N. J. A. Sloane): The On-Line Encyclopedia of the Integer Sequences. 2012, `http://oeis.org/A001791`.

[145] Özkan, S.: *Generalization of the Erdős-Gallai inequality*, Ars Combinatoria **98**, (2011) 295–302.

[146] Pálvölgyi, D.: *Deciding soccer scores and partial orientations of graphs*, Acta Universitatis Sapientiae, Mathematica, **1(1)**, (2009) 35–42.

[147] Parker, E. T., Reid, K. B.: *Disproof of a conjecture of Erdős and Moser on tournaments*, Journal of Combinatorial Theory, **9**, (1970) 225–238.

[148] Patrinos, A. N., Hakimi, S. L.: *Relations between graphs and integer-pair sequences.* Discrete Mathematics, **15(4)**, (1976) 347–358

[149] Pécsy, G., Szűcs, L.: *Parallel verification and enumeration of tournaments*, *Studia Universitatis* Babeş-Bolyai, Informatica, **45(2)**, 2 (2000) 11–26.

[150] Pehlivan, H., Nabiyev, V. N.: *Score calculation from final tournament tables*, Computers & Operations Research, **36(3)**, (2009) 936–950.

[151] Pirzada, S.: *An Introduction to Graph Theory*. Universities Press, Hyderabad, 2012, VIII + 396 oldal.

[152] Pirzada, S., Iványi, A.: *Minimal digraphs with given imbalance sequence.* Acta Universitatis Sapientiae, Mathematica, **4(1)**, (2012), 86–101.

[153] Pirzada, S., Naikoo, T. A., Shah, N. A., *Score sequences in oriented graphs*, Journal of Applied Mathematics & Computing, **23(1–2)**, (2007) 257–268.

[154] Pirzada, S., Naikoo, T. A., Samee, U., Iványi, A.: *Imbalances in multidigraphs*, *Acta Universitatis Sapientiae, Mathematica* **2(2)**, (2010) 133–145.

[155] Reid, K. B.: *Tournaments: scores, kings, generalizations and special topics*, Congressus Numerantium, **115**, (1996) 171–211.

[156] Reid, K. B.: *Tournaments*, in Handbook of Graph Theory (harmadik kiadás, szerk. Gross, J. L., Yellen, J., Zhang, P.), CRC Press, Boca Raton, 2013, 196–225.

[157] Reid, K. B., Zhang, C. Q.: *Score sequences of semicomplete digraphs*, Bulletin of Institute of Combinatorial Applications, **24**, (1998) 27–32.

[158] Rødseth, Ø. J., Sellers, J. A., Tverberg, H.: *Enumeration of the degree sequences of non-separable graphs and connected graphs.* European Journal of Combinatorics **30(5)**, 1309–1319.

[159] Royle, G.: *Is it true that* $a(n+1)/a(n)$ *tends to* 4*?*, In (szerk. N. J. A. Sloane): The On-Line Encyclopedia of the Integer Sequences. 2006. `http://oeis.org/A095268`.

[160] Rubinstein, A.: *Ranking the participants in a tournament*, SIAM Journal on Applied Mathematics, **38(1)**, (1980) 108–111.

[161] Ruskey, F., Cohen, F. R., Eades, P, Scott, A.: *Alley CATs in search of good homes.* Congressus Numerantium, **102**, (1994) 97–110.

[162] Ryser, H. J.: *Combinatorial properties of matrices of zeros and ones*, Canadian Journal of Mathematics, **9**, (1957), 371–377.

[163] Ryser, H. J.: *Matrices of zeros and ones in combinatorial mathematics*, in: Recent Advances in Matrix Theory (Proceedings of an Advanced Seminar, Mathematics Research Center, U.S. Army, University of Wisconsin, Madison, WI, 1963), University of Wisconsin Press, Madison, WI, 1964, 103–124.

[164] Sah, P., Singh, L. O., Clauset, A., Bansal, S.: *Exploring community structure in biological networks with random graphs*, bioRxiv, 2013, 19 oldal.

[165] Schoenfield, J. E., The number of football score sequences, in: ed. by N. J. A. Sloane, *The On-Line Encyclopedia of Integer Sequences,* 2008. http://oeis.org/A064626.

[166] Shuo-Yen, R. L. *Graphic sequences with unique realization*, Journal of Combinatorial Theory, Series B, **19**, (1975) 42–68.

[167] Sierksma, G., Hoogeveen, H.: *Seven criteria for integer sequences being graphic*, Journal of Graph Theory, **15(2)**, (1991) 223–231.

[168] Siklósi, B.: *Sportproblémákat megoldó soros és párhuzamos algoritmusok összehasonlítása.* Programtervező matematikusi diplomamunka. ELTE, Informatikai Kar, Budapest, 2001.

[169] Sloane, N. J. A.: *The number of ways to put $n+1$ indistinguishable balls into $n+1$ distinguishable boxes*, in (szerk. N. J. A. Sloane): The On-line Encyclopedia of the Integer Sequences. (ed. by N. J. A. Sloane). http://oeis.org/A004251.

[170] Sloane, N. J. A.: *The number of degree-vectors for simple graphs.* In: The On-Line Encyclopedia of the Integer Sequences (szerk. N. J. A. Sloane). http://oeis.org/A004251.

[171] Sloane: *The number of bracelets with n red, 1 pink and $n-1$ blue beads*, in (szerk. N. J. A. Sloane): The On-Line Encyclopedia of the Integer Sequences. (szerk. N. J. A. Sloane). http://oeis.org/AA005654.

[172] Sloane, N. J. A., Plouffe, S.: *The Encyclopedia of Integer Sequences,* Academic Press, 1995.

[173] Soroker, S.: *Fast Parallel Algorithms for Graphs and networks*, PhD értekezés, University of California, Berkeley, CA, 1987, 104 oldal.

[174] Soroker, S.: *Optimal parallel construction of prescribed tournaments,* Discrete Applied Mathematics, **29(1)**, (1990) 113–125.

[175] Spencer, J.: Optimal ranking of tournaments, Networks, **1**, (1971/1972) 135–138.

[176] Stockmeyer, P. K.: *The falsity of the reconstruction conjecture of tournaments*, Journal of Graphs, **1(1)**, (1977) 19–25.

[177] Takahashi, M., *Optimization Methods for Graphical Degree Sequence Problems and their Extensions*, PhD thesis, Waseda University, Tokyo, 2007. http://hdl.handle.net/2065/28387.

[178] Tanenbaum A. S., Wetherall, D. J.: *Computer Networks* (5. kiadás), Prentice Hall, 2010.

[179] Tetali, P. *Unique tournaments*, Journal of Combinatorial Theory, Series B, **72(1)**, (1998), 157–159.

[180] Thomassen, C.: *Landau's characterization of the tournament score sequences*, in: The Theory and Applications of Graphs (Kalamazoo, MI, 1980, szerk. G. Chartrand et al.) Wiley, New York, 1981, 589–591.

[181] Tripathi, A., Tyagy, H.: *A simple criterion on degree sequences of graphs.* Discrete Applied Mathematics **156(18)**, 18 (2008) 3513–3517.

[182] Tripathi, A., Vijay, S.: *A note on a theorem of Erdős & Gallai.* Discrete Mathematics **265(1–3)**, (2003) 417–420.

[183] Tripathi, A., Venugopalan, S., West, D. B.: *A short constructive proof of the Erdős-Gallai characterization of graphic lists*. Discrete Mathematics **310(4)**, (2010) 833–834.

[184] Tyshkevich, R. I., Melnikov O. I., Kotov, V. M.: *On graphs and degree sequences: a canonical decomposition* (orosz), *Kibernetika* **6**, (1981) 5–8.

[185] Volkman, L.: *Strong subtournaments of multipartite tournaments*, Australoasian Journal of Combinatorics, **20**, (1999) 189–196.

[186] Volkman, L.: *Multipartite tournaments: A Survey*, Discrete Mathematics, **307**, (2007) 3097–3129.

[187] Weisstein, E. W.: *Degree Sequence,* From MathWorld—Wolfram Web Resource, 2011. MathWorld—Wolfram Web Resource, 2014.

[188] Weisstein, E. W.: *Graphic Sequence,* From MathWorld—Wolfram Web Resource, 2014.

[189] Winston, K. J., Kleitman, D. J.: *On the asymptotic number of tournament score sequences*, Journal of Combinatorial Theory, Series A. **35**, (1983) 208–230.

[190] Yin, J.-H, Li, J.-S.: *Two sufficient conditions for a graphic sequence to have a realization with prescribed clique size*, Discrete Mathematics, **301(2–3)**, (2005), 218–227.

[191] Zverovich, I. E., Zverovich, V. E.: Contributions to the theory of graphic sequences, *Discrete Mathematics*, **105(1–3)**, (1992) 293–303.

(Beérkezett: 2014. március 8.)

IVÁNYI ANTAL
tony@inf.elte.hu
Eötvös Loránd Tudományegyetem
Informatikai Kar
1117 Budapest, Pázmány Péter sétány 1/C

KÁSA ZOLTÁN
kasa@ms.sapientia.ro
Sapientia Erdélyi Magyar Tudományegyetem
Matematikai és Informatikai Tanszék
Marosvásárhely/Koronka, Segesvári út 1/C

PARALLEL ENUMERATION OF DEGREE SEQUENCES

Antal Iványi, Zoltán Kása

The problem of testing, reconstruction and enumeration of the degree sequences of simple graphs has reach bibliography. In this paper we report on the parallel enumeration of the degree sequences of simple graphs resulting the number of sequences for $n = 24, \ldots, 31$ vertices.

Alkalmazott Matematikai Lapok **31** (2014), 99-108.

VÖDRÖK OPTIMÁLIS PAKOLÁSA RAKLAPOKRA

CSENDES TIBOR ÉS KOZMA ATTILA[1]

Egy nyomdaipari szállítási problémából kiindulva a raklapra való körpakolás feladatára olyan eljárást mutatunk be, amely képes reális számítási idő felhasználásával közel optimális megoldásokat adni. Megadjuk a talált pakolásokat a 80×120 arányú téglalapba $n = 1 - 50$ körre, és áttekintjük a kapott megoldások tulajdonságait. Tárgyaljuk az eredmények lehetséges alkalmazásai feltételeit és eredményességét.

A feladat

A négyzetbe való körpakolási feladatok megoldásában elért eredményeink [3, 4, 5, 7, 8, 9] alapján megkeresett bennünket egy nyomdai festékeket forgalmazó belga cég a következő problémával. A festékeiket vödrökben árulják, azokat raklapokra (lásd az 1. ábrát), nyilván átlapolás nélkül és a kilógást kerülve rakodták. Mégis azt tapasztalták, hogy a minimális kilógás a kamionra való pakolás során azt eredményezte, hogy a vödrök teteje felnyílt, és a festék egyrészt kárba veszett, másrészt a tisztítás miatt extra költségek álltak elő. Tőlünk azt kérdezték, hogy adott méretű vödrökből mennyi helyezhető el a 80×120 centiméteres Euro-raklapra, illetve hogy adott darabszámú vödröt előírva raklaponként mi a maximális átmérője azoknak a vödröknek, amivel azok még megfelelően elhelyezhetők.

A probléma tehát az, hogy egy 80×120 oldalarányú téglalap alakú területen hogyan helyezzünk el azonos méretű, adott számú köröket úgy, hogy azok ne fedjék át egymást, és ne is lógjanak túl a téglalap oldalain – és a körök által lefedett terület maximális legyen.

Az algoritmus

A megoldáshoz Eckard Specht négyzetbe való körpakolásra kifejlesztett programját [6, 10] igazítottuk át. Megjegyezzük, hogy a feltett kérdések megválaszolása még nem elegendő az optimális rakodáshoz, mert a kapott optimális elhelyezés a legtöbb esetben nagyon szabálytalan. Emiatt azok rutinszerű kirakása nem

[1]Köszönetnyilvánítás: A kutatást támogatta a Telemedicína fókuszú kutatások Orvosi, Matematikai és Informatikai tudományterületeken (TOMI) című pályázat: TÁMOP-4.2.2.A-11/1/KONV-2012-0073.

várható el a rakodóktól. Az optimális elhelyezések megadása már segíthet, de a valódi megoldást valószínűleg az ún. rácspakolások jelentik, amelyek szabályosságuk miatt könnyen alkalmazhatók. Az eredményeink persze bármilyen kör alapú áru esetén használhatók.

Algoritmus. PDS-algoritmus (Pulsating Disk Shaking)

0. lépés: Állítsunk elő egy lehetséges megoldást random módon. Legyen az aktuális sugár r, epszilon pozitív szám, egy kis konstans érték, továbbá $M := 0$ az átfedés és a túlnyúlás mértéke. $k := 0$.

1. lépés: $k := k + 1$, $r := r + \epsilon$. Ha $k >$ MAXIT, vagy a pulzálás amplitúdója megfelelően kicsi, akkor STOP.

2. lépés: Rázzuk össze az aktuális konfigurációt. Ha az M átfedés nem nőtt az előző iterációhoz képest, akkor folytassuk az 1. lépéssel.

3. lépés: $r := r - \epsilon$. Folytassuk az 1. lépéssel.

1. ábra. Az EUR raklap.

Az algoritmus előnye, hogy a futási idő nagy részében lehetséges megoldások között válogat, így a feltételeknek való megfelelés megteremtése viszonylag kevés időt visz el - szemben más eljárásokkal. Maga a program sztochasztikus jellegű, tehát közelítő megoldás szolgáltatására képes. Emiatt bizonyítottan optimális elhelyezést pusztán erre támaszkodva nem kaphatunk. Másrészt számos nehéz problémára kaptuk már meg ezzel az optimális megoldást rövid idő alatt, amit aztán egy megbízható számítógépes eljárással könnyebb volt verifikálni, mint az utóbbival azt elő is állítani (vö. [9]).

Az egyszerű algoritmusunk már sikeresnek bizonyult számos körpakolási feladat megoldására. Megvizsgáltuk a feladatunkat Jozef Kallrath új, GAMS és Baron alapú eljárásával is [1, 2], de az azzal kapott eredmények egyrészt nem voltak megbízhatók, másrészt szinte mindig gyengébb közelítéseket kaptunk csak.

Eredmények

Az eljárást $N = 1 - 50$ darab körre futtattuk le. A kapott eredmények csak jó lehetséges megoldások, de ezek optimalitása már nem garantált. Persze az optimumtól való kis eltérés a gyakorlati alkalmazásban általában elfogadható. Másrészt tényleges rakodáshoz csak az olyan konfigurációk használhatók, melyek valamilyen könnyen érvényesíthető szabályszerűséget tartalmaznak. A kapott eredmények egy részét a 2–4. ábrák mutatják.

Nem szimmetrikus konfigurációk tartoznak ezekből az $n = 7$, 10, 14, 17, 19, 21, 22, 23, 26, 27, 29 értékekhez (30-ig, lásd az 1. táblázatot). Érdekes, hogy viszonylag milyen sok szimmetrikus elhelyezést találtunk, bár a nagyobb körszámok esetén egyre kevesebb van. Ez összhangban van a síkon való optimális körpakolásra vonatkozó elméleti eredménnyel.

N	r	szimmetria	rács	N	r	szimmetria	rács
1	40.000	✓		16	11.7294	✓	
2	30.718	✓	✓	17	11.4808		
3	24.0408	✓	✓	18	11.3291	✓	✓
4	22.005	✓	✓	19	11.0779		
5	20.3992	✓	✓	20	10.9677	✓	✓
6	20.000	✓	✓	21	10.5439		
7	17.1786			22	10.2746		
8	16.3175	✓	✓	23	10.1396		
9	15.2646	✓	✓	24	10.000	✓	✓
10	14.8085			25	9.77135	✓	
11	14.721	✓	✓	26	9.45412		
12	13.8538	✓	✓	27	9.29373		
13	13.3348	✓	✓	28	9.2196	✓	
14	12.806			29	9.06494		
15	12.3107	✓	✓	30	9.02455	✓	

1. táblázat. Az N darab kör elhelyezésének adatai: a talált legnagyobb sugár, és annak megjelölése, hogy az adott pakolás szimmetrikus-e, illetve rácspakolás-e.

A táblázatban megadott kapott sugárértékek jól tükrözik azokat a helyzeteket, amikor várható megoldás született (mint például az $n = 1, 6$ és 24 kör esetén). Megjegyzendő az is, hogy a 6 és 24 kör optimális pakolása egyben az ipari standarddal is egyezik, ugyanis eszerint csomagolják a különféle italokat.

Az ábrákon az adott pontossággal egymással érintkező köröket szaggatott vonalak jelzik. Az egyes körök színe a kör tulajdonságát tükrözi: a szabad körök (amelyek mozgathatók kicsit, és a pakolás sűrűsége emiatt nem változik, az első példa a 7 kör pakolását mutató rajzon látható a 2. ábrán) a legsötétebbek. Az egyes pakolások feliratát a program maga generálta, ezért azok angolul vannak. Az ábrafeliratok megadják a körök sugarát (angolul radius), a pontpakolási feladat alakra vonatkozóan a pontok távolságainak minimumát (distance), a körpakolás sűrűségét (density), valamint az említett értelemben vett érintkezések, kapcsolatok számát (contacts). A 45 kör esetén láthatólag még elérhető sűrűbb pakolás is.

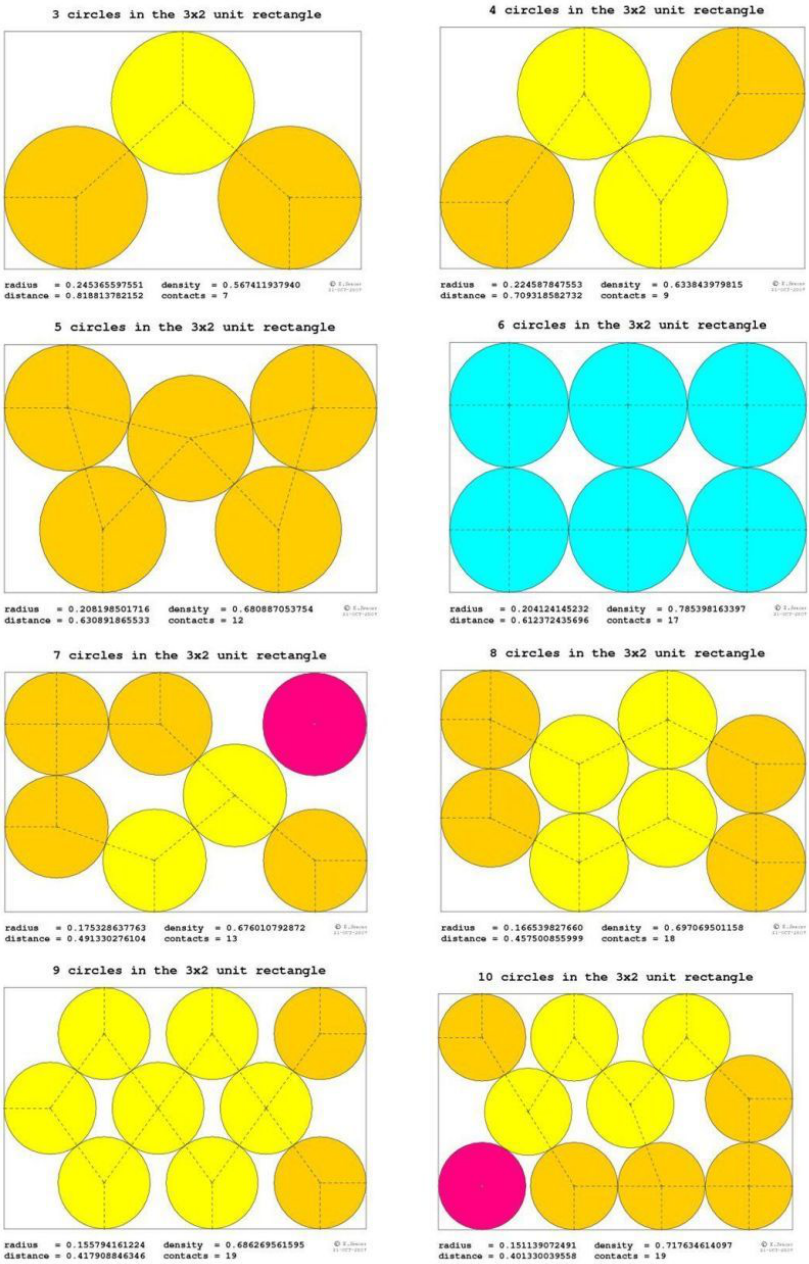
A teljes síkon optimális körpakolási minta, a méhsejt szerkezetű, hexagonális pakolás először a 31 körnek a 80×120 arányú téglalapba való elhelyezése során jelentkezik. Ez összhangban van azzal az észrevétellel, amit a négyzetbe való pakolás során tettünk ([9]): a körszám növekedésével a hexagonális táblákból álló részstruktúrák uralják el fokozatosan a véges tartományon vett pakolásokat, és ezeket a táblákat egészítik ki ezekhez illeszkedő körök.

A kapott eredmények optimalitása nem igazolt, tehát ezt intervallum aritmetikára épülő eljárásokkal kell verifikálni, mint amilyen eljárásokat a négyzetbe való pakolások egyes egyszerűbb eseteiben meg tettünk (lásd [4, 5, 7, 8]). Ebből a szempontból tanulságos, hogy a különben elfogadott GAMS–Baron módszertan (amilyent az [1, 2] közlemények alkalmaztak) milyen gyakran adott nem optimális, és az optimalitási rést illetően félrevezető eredményt. A sugár értékek szigorúan monoton módon csökkennek a növekvő körszámmal. Ez a tulajdonság általában természetesnek mondható, bár van ezen szabály alól kivétel, például a körbe körpakolás $n = 6$ és 7 esetei azonos sugárral adják az optimális elhelyezést. Ezzel együtt ez utóbbi jelenség kivételesnek számít.

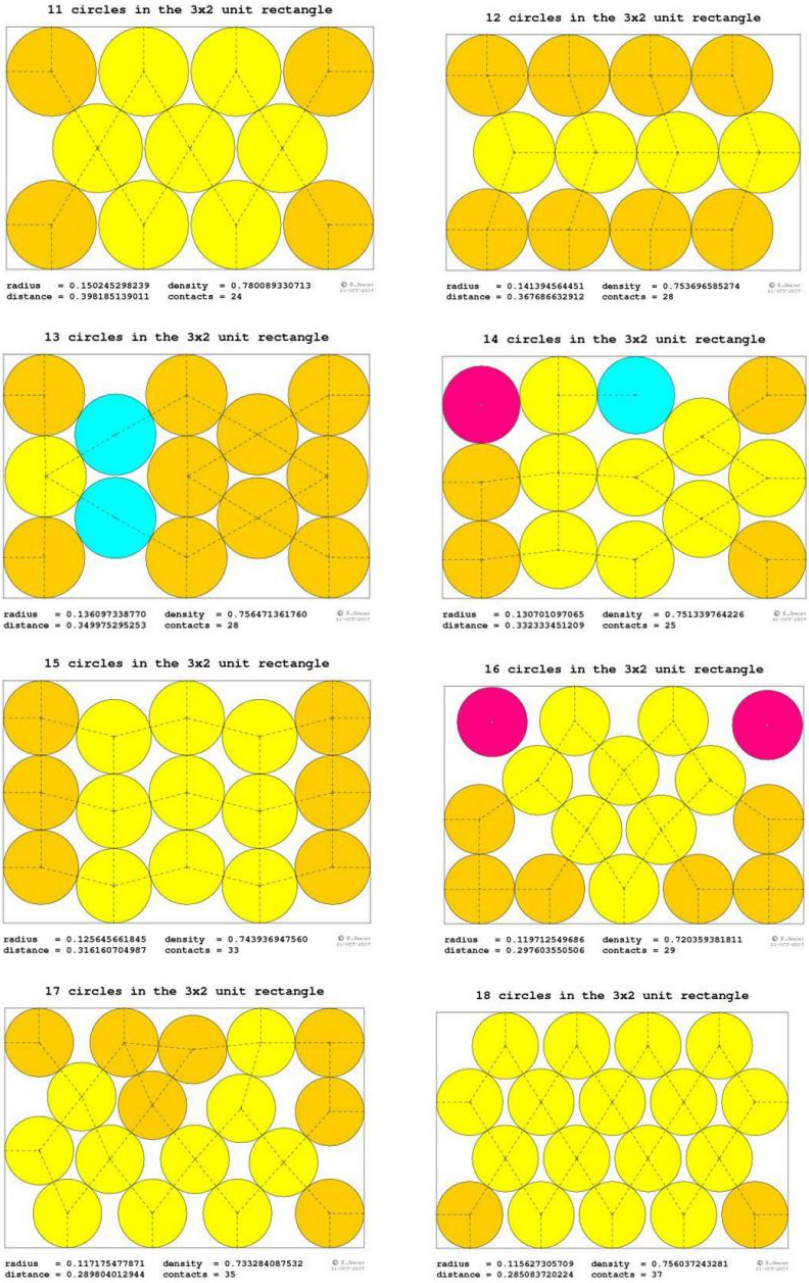
Alkalmazás

1. Megjegyzendő, hogy a gyakorlati alkalmazást további tényezők is befolyásolják. Így a hasonló rakodás során használt zsugorfólia vagy pánt a helyesen lerakott elhelyezést elronthatja. Ezek a hatások az adott szerkezetet nyilvánvalóan a körbe írt pakolások irányában torzítják. Emiatt a helyesen elhelyezett vödrök vagy hordók a raklapról lelóghatnak, és épp a megbízó által kifogásolt jelenség jelentkezik, az áru sérül, és aránytalanul nagy költségnövekedést okozhat. Az alább mutatott optimális elhelyezések rögzítését úgy kell megoldani, hogy az a fentieknek megfelelően ne torzuljon olyanná, ami megsérti a befoglaló téglalap határát.

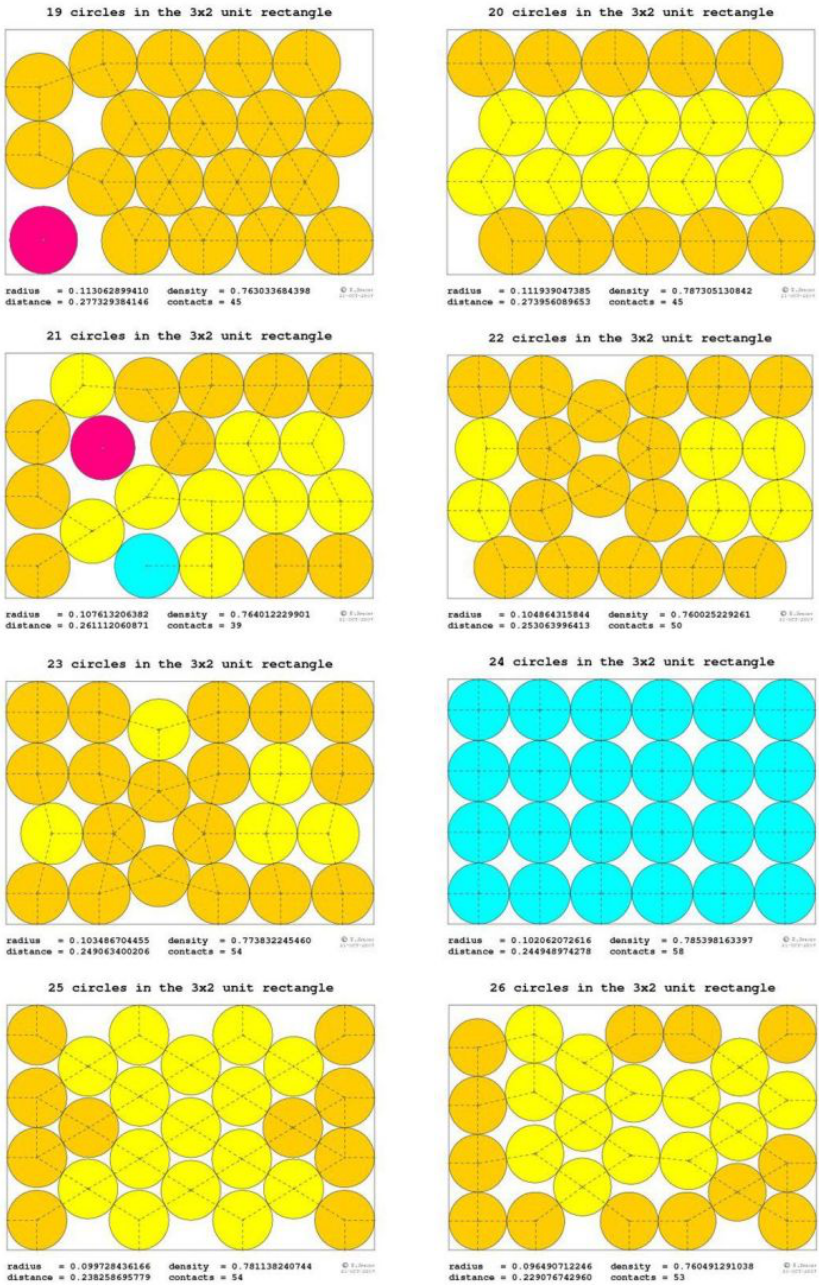
A talált előnyös pakolási minták sokszor nehezen szerkeszthetők, emiatt a pakolás kialakítása nehézséget jelenthet. Az általában nem racionális számmal adott



2. ábra. A raklap pakolásra kapott közel optimális megoldások az $n = 3 - 10$ esetekre. A részábrákon szerepel a sugár, a sűrűség, a pont pakolási távolság és az érintések száma.



3. ábra. A raklap pakolásra kapott közel optimális megoldások az $n = 11 - 18$ esetekre.



4. ábra. A raklap pakolásra kapott közel optimális megoldások az $n = 19 - 26$ esetekre.

koordináták, valamint a tényleges elhelyezés technikai gondjai miatt a pakolások megvalósítására külön gondot kell fordítani. E célból előre nyomott papírt vagy vetítést lehet alkalmazni.

2. A gyakorlati hasznosítás szempontjából kitűntetett jelentőségűek az úgy nevezett rácspakolások [9], amelyekben a körök középpontjai egy a téglalapra fektetett rács egyes szabályosan elhelyezkedő rácspontjai. A jelen közleményben megadottak nem mind ilyenek (jellemző ilyen példa az itt általunk megadott pakolás a 11, illetve a 12 körre). Amennyiben minden körszámra ilyeneket akarunk megadni, akkor az ennek megfelelő további feltételeket be kell építeni a matematikai modellbe, és az optimalizálást ennek megfelelően végrehajtani. Ezt a most alkalmazott eljárás nem támogatja, de bizonyos szempontból egyszerűbb az optimális rácspakolások meghatározása, mint az általános alakúaké.

A rácspakolások lerakása egyszerű: a vödröket vagy hordókat soronként lehet elhelyezni, és az egymást követő sorokat szabad az érintkezésig elcsúsztatni. Ez az eljárás a pakolás szerkezete miatt nem okozhatja aztán a tárolóedények túllógását a raklap határain. Ennek ellenére az 1. pontban említett technikai segédeszközök itt is segíthetnek az optimális pakolásnak megfelelő elhelyezés megtalálásában.

3. A bemutatott körpakolásokkal elérhető megtakarítás önmagában nem feltétlen jelentős, mindössze pár százalékos lehet. Ennek ellenére egyes esetekben ez azt eredményezheti, hogy kevesebb szállítóeszközzel lehet megoldani az adott áru megrendelőhöz való továbbítását. Másrészt ezek használata nyilván ésszerű, és külön költséget nem jelent, míg közben megtakarítást érhetünk el velük. Említésre méltó itt az a megjegyzés, ami hasonló logisztikai feladatok megoldásában általánosan elfogadott szakmai tervezési elv, miszerint sokszor érdemes a szállítóeszközök minimalizálására törekedni az egyéb ésszerű szempontok helyett (a lehető legkisebb személyi kiadások, minimális költség, legrövidebb teljes rakodási időtartam stb.).

4. Felmerülhet, hogy az általunk is tárgyalt kör darabszámok túl nagyok, ennyi hengeres dobozt, vödröt vagy hordót nem szokás ebben a formában csomagolni. Ez jogos, különösen nagy körszámra és abban az esetben, ha nincsenek olyan eszközeink, amivel biztosítani tudjuk, hogy a raklap határain ne csússzanak kívül a tárolóedények. Természetesen a téglalapba való körpakolás más alkalmazásai esetén, amit a következő pontban tárgyalunk röviden, ezek az érvek kevésbé hatnak.

Az a gyártási eljárás, amely a gépsorokról lejövő kör alapú csomagolású termékeket kis darabszámban a raklapnál lényegesen kisebb egységekbe csomagolja, majd ezeket rendezi a raklapra, közvetve szintén támaszkodik a téglalapba való körpakolásra. A raklapnál kisebb méretű csomagok kialakítása mellett szól az, hogy ezek kézzel jobban rakodhatók. Természetesen ezek a kisebb méretű pakolási feladatok is megoldhatók az ismertetett módszerrel, sőt adott esetben a téglalap oldalai hossza arányának egyezése esetén az eredményeink közvetlenül is használhatók erre az esetre is.

5. Más alkalmazások is vannak a téglalapba való körpakolás eredményeinek hasznosítására. A pakolási feladatoknak megfeleltethetők a szabási feladatok, ame-

lyek adott nyersanyagból a lehető legnagyobb méretű végtermékek kivágását, vagy ennek megfelelő módon adott téglalap alakú nyersanyagból a legtöbb adott méretű végtermék leszabását célozzák. Értékes nyersanyag esetén az így elérhető megtakarítás tetemes lehet. A bemutatott módszertan az általunk tárgyalttól eltérő esetekben is alkalmazható.

Bolyai Farkas is tárgyalta a körpakolási feladatok olyan alkalmazását, amelyek adott kétdimenziós alakzatba (négyzetbe, téglalapba vagy háromszögbe) egybevágó körök olyan nem átlapoló elhelyezését kereste, hogy a körök sugara maximális legyen. Ő ezt egy tervezett erdészi állás pályázatához vizsgálta. A körök szerepét az indokolja, hogy ezek adják meg az egyes fák életterét. Az adott terület legjobb kihasználását pedig épp az optimális körpakolási megoldás tudja biztosítani.

A pakolási feladatok mellett a lefedési problémák is ide kapcsolódnak. Ezek bizonyos értelemben duálisai a pakolási feladatoknak. Előbbi esetén az a célkitűzés, hogy adott korlátos tartományt (esetleg átlapolással) teljes mértékben lefedjünk minimális darabszámú egybevágó adott alakzattal. Ebben az értelemben egy adott terület, például egy ország minimális számú telekommunikációs adótoronnyal való lefedése (feltételezve, hogy adott megkövetelt térerő esetén az adótorony hatóköre kör alakú), körökkel való lefedési feladatra vezet. Másrészt interferencia és egyéb technikai jelenségek miatt az elhelyezendő objektumok egymástól egy minimális távolságot kell hogy tartsanak. Ez viszont ismét a körpakolási feladatot jelenti.

Kicsit távolabbról, de ide kötődik a Latin hiperkocka tervezés [11], amely adott térrész koordinátánként különböző koordinátájú, minél távolabbi pontok kijelölésére törekszik. Ez számos alkalmazással rendelkezik, a kódtervezéstől a mérési mintavételezés ilyen szempontból optimális meghatározásáig.

További kutatásokat kell ezt követően végezni a matematikai értelemben vett optimalitás igazolására, és az optimális rácspakolások numerikus meghatározására.

Hivatkozások

[1] J. Kallrath, S. Rebennack, and P. Pardalos: *Column Enumeration based Decomposition Techniques for a Class of Non-Convex MINLP Problems.* J. of Global Optimization **43** (2009), 277–297

[2] J. Kallrath, S. Rebennack, and P. Pardalos: *Cutting Circles and Polygons from Area-Minimizing Rectangles.* J. of Global Optimization **43** (2009), 299–328

[3] M. CS. Markót: *An Interval Method to Validate Optimal Solutions of the „Packing Circles in a Unit Square" Problems*, Central European Journal of Operational Research **8** (2000) 63–78

[4] M. CS. Markót and T. Csendes: *A new verified optimization technique for the "packing circles in a unit square" problems.* SIAM J. on Optimization **16** (2005), 193–219

[5] M. CS. Markót and T. Csendes: *A reliable area reduction technique for solving circle packing problems.* Computing **77** (2006), 147–162

[6] E. Specht: *Circles In ReCtangle (crc.c) algoritmus.*

[7] P. G. Szabó: *Some New Structures for the „Equal Circles Packing in a Square" Problem.* Central European Journal of Operations Research **8** (2000), 79–91

[8] P. G. Szabó: *Optimal substructures in optimal and approximate circle packings.* Beiträge zur Algebra und Geometrie **46** (2005), 103–118

[9] P. G. Szabó, M. CS. Markót, T. Csendes, E. Specht, L. G. Casado, and I. García: *New Approaches to Circle Packing in a Square – With Program Codes.* Springer, Berlin, 2007

[10] Packomania vendégoldal: http://www.packomania.com

[11] E. R. Van Dam, B. Husslage, D. Den Hertog, and Hand Melissen: *Maximin Latin Hypercube Designs in Two Dimensions.* Operations Research **55** (2007), 158–169

(Beérkezett: 2014. június 5.)

CSENDES TIBOR
SZTE Számítógépes Optimalizálás Tanszék
6720 Szeged, Árpád tér 2.
csendes@inf.u-szeged.hu

KOZMA ATTILA
StreamNovation Kft.
1083 Budapest, Práter utca 50/a.
Attila.Kozma@esat.kuleuven.be

OPTIMAL PACKING OF BUCKETS ON EUR-PALETTE

Tibor Csendes and Attila Kozma

A Belgian firm selling printing color asked us to help their delivery services with providing optimal number of buckets or barrels that can be positioned on a standard size eur-palette (80×120 cm). The reason why they were interested was that the earlier ad hoc packing resulted in a not exact fit, and during the loading of the lorries the lids of the bins containing the printing colors opened and the material was wasted causing substantial losses both expressed in money and time of the delivery.

We used the Pulsating Disk Shaking algorithm of Eckard Specht, and here we report the best approximative solutions obtained with some discussion on the quality of these.

Az Alkalmazott Matematikai Lapok megjelenését támogatja
a Magyar Tudományos Akadémia Könyv- és Folyóiratkiadó Bizottsága.

A kiadásért felelős a BJMT főtitkára
Szedte és tördelte Éliás Mariann

Nyomta a Synra Nyomda és Kiadó Kft., Budapest
Felelős vezető: Szűcs Ernőné

Budapest, 2014
Megjelent 18 (A/5) ív terjedelemben
250 példányban
HU ISSN 0133-3399

ÚTMUTATÁS A SZERZŐKNEK

Az Alkalmazott Matematikai Lapok csak magyar nyelvű dolgozatokat közöl. A közlésre szánt dolgozatokat e-mailen az `aml@math.elte.hu` címre kérjük elküldeni az ábrákat tartalmazó fájlokkal együtt. Előnyben részesülnek a LaTeX-ben elkészített dolgozatok.

A kéziratok szerkezeti felépítésének a következő követelményeket kell kielégíteni:

Fejléc: A fejlécnek tartalmaznia kell a dolgozat címét és a szerző teljes nevét.

Kivonat: A fejléc után egy, képletet nem tartalmazó, legfeljebb 200 szóból álló kivonatot kell minden esetben megadni.

Fejezetek: A dolgozatot címmel ellátott szakaszokra kell bontani, és az egyes szakaszokat arab sorszámozással kell ellátni. Az esetleges bevezetésnek mindig az első szakaszt kell megnevezni.

A dolgozatban előforduló képleteket a dolgozat szakaszokra bontásától független, folytatólagos arab sorszámozással kell azonosítani. Természetesen nem szükséges minden képletet számozással ellátni, csak azokat, amelyekre a szerző a dolgozatban hivatkozni kíván.

Mind az ábrákat, mind a lábjegyzeteket szintén folytatólagos arab sorszámozással kell ellátni. Az ábrák elhelyezését a dolgozat megfelelő helyén ábraazonosító sorszámokkal kell megadni. A lábjegyzetekre a dolgozaton belül az azonosító sorszám felső indexkénti használatával lehet hivatkozni.

Az esetleges definíciókat és tételeket (segédtételeket és lemmákat) szakaszonként újrakezdődő, ponttal elválasztott, kettős számozással kell ellátni. Kérjük a szerzőket, hogy ezeket, valamint a tételek bizonyítását a szövegben kellő módon emeljék ki.

Irodalomjegyzék: A dolgozatok szövegében az irodalmi hivatkozás számait szögletes zárójelben kell megadni, mint például [2] vagy [1, 7–13].

Az irodalmi hivatkozások formája a következő: Minden hivatkozást fel kell sorolni a dolgozat végén található irodalomjegyzékben, a szerzők, illetve a társszerzők esetén az első szerző neve szerint alfabetikus sorrendben úgy, hogy a cirill betűs szerzők nevét a Mathematical Reviews átírási szabályai szerint latin betűsre kell átírni. A folyóiratban megjelent cikkekre [1], a könyvekre [2] a következő minta szerint kell hivatkozni:

[1] Farkas, J.: *Über die Theorie der einfachen Ungleichungen*, Journal für die reine und angewandte Mathematik **124**, (1902) 1–27.

[2] Zoutendijk, G.: *Methods of Feasible Directions*, Elsevier Publishing Company, Amsterdam and New York (1960), 120 o.

Szerző adatai: Az irodalomjegyzék után, a kézirat befejezéseképpen fel kell tüntetni a szerző teljes nevét és a munkahelye (esetleg lakása) pontos címét, illetve e-mail címét.

Idegen nyelvű kivonat: Minden dolgozathoz csatolni kell egy angol nyelvű összefoglalót.

A szerzők a dolgozatukról 20 darab ingyenes különlenyomatot kapnak. A dolgozatok után szerzői díjat az Alkalmazott Matematikai Lapok nem fizet.

TARTALOMJEGYZÉK

INDEX