PÉCSI TUDOMÁNYEGYETEM
KÖZGAZDASÁGTUDOMÁNYI KAR

GAZDÁLKODÁSTANI DOKTORI ISKOLA

Szendrői Etelka

Egy új harmóniakereső metaheurisztika erőforrás-korlátos több megvalósítási módú projektek ütemezésére

DOKTORI ÉRTEKEZÉS

Témavezető: Dr. Csébfalvi Anikó CSc, Phd
Dr. Csébfalvi György CSc

Pécs, 2011

Tartalomjegyzék

Köszönetnyilvánítás

Szeretném megköszönni témavezetőimnek Dr. Csébfalvi Anikó egyetemi docensnek és Dr. Csébfalvi György egyetemi docens úrnak a támogatást, a biztatást, amely nagyon sok segítséget jelentett számomra. Köszönöm a konzultációs lehetőségeket, amelyek során folytatott beszélgetések nagyban hozzájárultak a felmerülő problémák leküzdéséhez.

Köszönettel tartozom a PTE Pollack Mihály Műszaki és Informatikai Karnak, hogy anyagi támogatással lehetővé tették a konferenciákon való részvételemet, amelyek fontos állomásai voltak a dolgozat elkészítésének.

1. Bevezetés

A disszertációban egy új harmóniakereső metaheurisztikát mutatunk be, amely a több-megvalósítási módú erőforrás-korlátos ütemezések halmazán a projekt időtartamát minimalizálja. A több-megvalósítási módú erőforrás-korlátos projektütemezés egyik kiemelten fontos területe a projektütemezési problémáknak, amelyek NP-nehéz feladatok. Kutatásunk érdeklődési középpontjában a heurisztikák szerepelnek, célunk egy hatékony heurisztikus eljárás létrehozása volt.

Mindennapi tevékenységünk során számos kisebb-nagyobb projektet kell megoldanunk, akár a család életének szervezése, akár a munkahelyi feladatok ellátása a célunk. Ezek a projektfeladatok a tevékenységek számában, bonyolultságában lényegesen különböznek egymástól. Azonban minden projekt alapvető jellemzője az, hogy egyszeri, megismételhetetlen, megadott határidőre be kell fejeződnie, s nem lépheti túl a költségvetési korlátot.

Az üzleti világban előforduló projektek összetettek, akár több száz tevékenységet is tartalmazhatnak, sok és sokféle erőforrást igényelnek. A projektvezetők egyik legfontosabb feladata a tevékenységek megfelelő ütemezése mellett a rendelkezésre álló erőforrások hatékony felhasználása, a tevékenységek végrehajtásához szükséges erőforrások folyamatos biztosítása. Fontos cél a tétlen, várakozó, majd újrainduló erőforrások számának minimalizálása. Számos esetben lehetőség van arra, hogy a projekt egyes tevékenységeit más-más módon végezzék el, s így erőforrást,

pénzt vagy időt takarítsanak meg. Az ezzel kapcsolatos döntések meghozatala a projektvezetők felelőssége.

A projektütemezési feladatokkal foglalkozó tudományterületen belül növekvő figyelem irányul a több-megvalósítási módú projektek felé, hiszen a tevékenységek többféle módon történő végrehajtása lehetőséget biztosít arra, hogy a projekt időtartamának minimalizálásán túl, kihasználva a több megvalósítási módban rejlő lehetőséget, az erőforráskorlátok megsértése nélkül olyan megvalósítási módban és úgy ütemezzük a tevékenységeket, hogy az más szempontokat figyelembe véve is optimális legyen.

A több-megvalósítási módú projektek ütemezése során különböző célokat valósítanak meg, melyek közül kiemelkedő gyakorlati jelentőségű a projekt időtartamának minimalizálása, a tevékenységek között fennálló elsőbbségi feltételek és erőforráskorlátok figyelembevétele mellett. Másodlagos cél lehet az erőforrás felhasználásban jelentkező ingadozások csökkentése, azaz az erőforrás felhasználási hisztogram kisimítása.

A szakirodalom tanulmányozása során megállapítható volt, hogy az egzakt módszerek alkalmazása a több-megvalósítási módú erőforrás-korlátos projektek ütemezésére csak kisméretű problémák esetén ad elfogadható időn belül optimális megoldást. A publikált eredmények beszámolnak sikeres egzakt algoritmusokról, de azt is megállapítják, hogy még a legsikeresebb egzakt algoritmusok is csődöt mondanak, nem találnak megoldást akkor, ha a tevékenységek száma meghaladja a 20 tevékenységet. Ennek tükrében az egyetlen lehetséges alternatíva csakis a heurisztikus megoldás lehet.

Vizsgálataink alapján arra a következtetésre jutottunk, hogy a heurisztikák közül a metaheurisztikus algoritmusok területén kell keresni a probléma természetéhez legmegfelelőbb és leghatékonyabb eljárást.

A metaheurisztikák hatékonyságát számos kutató bizonyította a szakirodalomban közölt futtatási eredményekkel. Ezeket a számítási eredményeket a jól ismert PSPLIB teszt halmazon végezték el. (A PSPLIB teszthalmaz részletes ismertetésére a 2. fejezetben kerül sor.) A szakirodalom áttanulmányozása megerősítette azt a meggyőződésünket, hogy a több-megvalósítási módú erőforrás-korlátos projektütemezési probléma megoldására metaheurisztikát kell alkalmaznunk, csakis a metaheurisztikus algoritmusok nyújtják azt a rugalmasságot és hatékonyságot, amelyet a feladat nehézségi foka megkövetel. Természetesen egy jó metaheurisztika, amely jól definiált operátorhalmazzal rendelkezik elképzelhető, hogy sok feladattípus megoldására alkalmazható. Ilyen például a hangyaboly metaheurisztika. Sok feladatot lehet elképzelni, amelyeknek megoldására alkalmas. Azonban egy jó metaheurisztika létezéséből még nem következik az, hogy azt jól is alkalmazzuk egy konkrét probléma megoldására. Az adaptáció minősége a fontos, vagyis a "mesét" mennyire sikeresen adaptáljuk az adott feladatra. A metaheurisztikák közül a harmóniakereső, Sounds of Silence algoritmusra (Csébfalvi, 2008a, 2008b) esett a választás, amely erőforrás-korlátos projektütemezési feladatok optimális, minimális időtartamú ütemezésére lett kifejlesztve. Ennek az algoritmusnak a továbbfejlesztett változata a disszertációban bemutatásra kerülő metaheurisztika, amely alkalmas a több-megvalósítási módú projektek optimális ütemezésére.

A továbbfejlesztés legjelentősebb eredménye, hogy egy előoptimalizálást végző eljárással bővítve az algoritmust, jelentős mértékű sebesség és minőségi növekedést értünk el.

Az algoritmus minőségének további javítását a projekt időtartamának és az egyes tevékenységek megvalósítási módjának rögzítése után a mozgatható tevékenységek halmazán végzett javító eljárások alkalmazásával értük el.

Az algoritmus továbbfejlesztésének következő szakaszában alkalmassá tettük az algoritmusunkat másodlagos cél szerinti optimalizálásra is. Másodlagos optimalizálási kritériumként az erőforrás felhasználás kiegyenlítése, az erőforrás felhasználási hisztogram kisimítása került meghatározásra.

A dolgozat felépítése a következő. A bevezetést követően, a második fejezetben a projektütemezési tématerület legfontosabb fogalmait részletezzük, valamint áttekintjük a projektütemezés lehetséges céljait a több-megvalósítási módú projektekre vonatkozóan. A második fejezet végén ismertetjük a PSPLIB tesztkönyvtár legfontosabb tulajdonságait, felhasználhatóságát. A PSPLIB tesztkönyvtár a kutatók széles köre számára nyújt lehetőséget arra, hogy a könyvtárban található projekteken próbálják ki új megoldó algoritmusaik teljesítőképességét, valamint összehasonlíthassák eredményeiket más kutatók ugyanazon projektek ütemezése során kapott eredményeivel.

A disszertáció harmadik fejezetében áttekintjük a több-megvalósítási módú projektütemezési problémák megoldási módszereit, az elsősorban a nemzetközi szakirodalomban közzé tett eljárásokat. A disszertációban kiemelten foglalkozunk a heurisztikus és metaheurisztikus eljárásokkal,

mivel a problémakör nehézségi foka a heurisztikák illetve metaheurisztikák alkalmazását teszi szükségessé.

A negyedik fejezetben részletesen ismertetjük az új harmóniakereső algoritmusunkat, amely a több-megvalósítási módú erőforrás-korlátos projektek optimális ütemezését oldja meg. Az algoritmus többcélú optimalizálásra is fel lett készítve, másodlagos célként az erőforrás felhasználási hisztogram kisimítását végzi, rögzített időtartam, és megvalósítási mód mellett a mozgatható tevékenységek halmazán. A fejezetben részletesen ismertetjük az algoritmus fő lépéseit. A negyedik fejezet végén részletes számítási eredményeket közlünk, amellyel alátámasztjuk az algoritmus képességeit.

Az ötödik fejezetben az algoritmus továbbfejlesztésével kapcsolatos elképzeléseket és eredményeket ismertetjük. A továbbfejlesztés egyik legfontosabb pontja az előoptimalizálás alkalmazása, amely jelentősen javított az algoritmus minőségén és gyorsaságán. A további minőségjavító eljárások mellett az algoritmus másik jelentős továbbfejlesztése a másodlagos kritérium szerinti optimalizálás lehetőségének megvalósítása. A részletes számítási eredmények bemutatásával zárjuk a disszertációt.

A dolgozatban használt jelöléseket az 1. táblázatban foglaltuk össze.

1. táblázat Jelölések listája

N	A valódi tevékenységek száma
M	A tevékenységek megvalósítási módjainak száma
m	Megvalósítási módok indexe, $m \in \{1,2,...M\}$
i	az i -edik tevékenység indexe, $i \in \{1,2,...N\}$
D_{im}	az i -edik tevékenység időtartama az m -edik megvalósítási módban
$i \to j$	Az i tevékenység a j tevékenység előzménye
PS	Az $i \to j$ elsőbbségi (megelőző-rákövetkező) kapcsolatok halmaza
R	A megújuló erőforrás típusok száma
C	A nem-megújuló erőforrás típusok száma
R_{imr}	Az i -edik tevékenység erőforrás szükséglete az r -edik megújuló erőforrásból az m -edik megvalósítási módban, $r \in \{1,...,R\}$
R_r	az r -edik megújuló erőforrás felső korlátja
C_c	a c -edik nem-megújuló erőforrás felső korlátja
C_{imc}	az i -edik tevékenység erőforrás szükséglete a c -edik nem-megújuló erőforrásból az m -edik megvalósítási módban
$\overline{T}$	A projekt időszükségletének felső korlátja (a tevékenységek időtartamainak összege)
T	a projekt időperiódusainak száma t∈ {1,2,…,T}
$\underline{T}$	A projekt időszükségletének alsó korlátja (Az elsőbbségi feltételeket kielégítő projekt minimális időtartama)
X_{ims}	bináris változó, $X_{ims} \in \{0,1\}$ értéke 1, ha az i -edik tevékenység az m -edik megvalósítási módban, s időpontban kezdődik el, egyébként 0.

X_i	az i -edik tevékenység kezdőidőpontja nem korlátos esetben
$\underline{X}_{im}$	az i -edik tevékenység legkorábbi kezdési időpontja az m -edik megvalósítási módban a nem korlátos, (csak elsőbbségi korlátokat kielégítő) esetben
$\overline{X}_{im}$	az i -edik tevékenység legkésőbbi kezdési időpontja az m -edik megvalósítási módban a nem korlátos, (csak elsőbbségi korlátokat kielégítő) esetben
s	A tevékenység kezdési időpontja az $\underline{X}_{im}$, $\overline{X}_{im}$ időintervallumban
M_i	Az i -edik tevékenység megvalósítási módjainak száma
W_{imst}	Bináris változó, értéke 1, ha a tevékenység aktív a t időperiódusban, egyébként 0.
$\boldsymbol{PS}^*$	A $\boldsymbol{PS}$ halmaz és a konfliktusjavító reláció halmaz nem redundáns részhalmazának uniója.
CU^+_{tr}	A t időperiódusban induló, újrainduló erőforrásegységek száma az r -dik erőforrásból
CU^-_{tr}	A t időperiódusban a leálló erőforrásegységek száma az r -dik erőforrásból.
T_i	Az i -dik tevékenységhez tartozó időablak
A_t	A t időperiódusban aktív tevékenységek halmaza

2. A projektütemezési modellek komponensei

A projektmenedzsment kiemelkedően fontos része a projektütemezés, amely a gyakorlati feladatok oldaláról tekintve a témakör egyik legnehezebb területe. A gazdasági életben megoldandó feladatok mérete egyre nő, az erőforrások pedig egyre szűkösebben állnak rendelkezésre. A mindennapi gyakorlatban előforduló projektek szerteágazóak, sok feladatból állnak, így kiemelkedő jelentősége van a projektmenedzsment számára, hogy a projektek időtartama minimális legyen, s az erőforrások felhasználása megszakításmentes legyen a projekt végrehajtása során.

A projekt egymással kapcsolatban lévő *tevékenységek* halmaza, ahol a tevékenységek közötti kapcsolatokat *elsőbbségi feltételek* írják le. Az egyes tevékenységeket végrehajtásuk *időszükségletével* valamint az *erőforrásigényével* jellemezzük. A projektütemezési modellek a projekt teljesítményének mértékében, az erőforrások felhasználási módjaiban különböznek egymástól. A projektek teljesítményének mértékét az optimalizálás *célfüggvényei* fejezik ki.

Az ütemezési feladatok típusokba sorolásával, osztályozásával számos kutató foglalkozott. Az egyik leggyakrabban alkalmazott osztályozási rendszert Bruckner és ts. (1999) dolgozták ki. Jelölésrendszerük három mezővel, $\alpha|\beta|\gamma$ jelzi az erőforrások, tevékenységek, célfüggvény jellemző tulajdonságait. A továbbiakban a projekt komponensei, a tevékenységek, kapcsolatok, erőforrások és az ütemezés jellemzőit, csoportosítási szempontjait foglaljuk össze.

2.1 Tevékenységek

A projekt tevékenységekből (activities, jobs, tasks) áll, melyek végrehajtásával történik a projekt megvalósítása. A tevékenységeket a végrehajtásukhoz szükséges *időtartamukkal* (*duration*), *erőforrásigényükkel* (*resource requirement*), más tevékenységekhez való *kapcsolatukkal* (*relations*), valamint *végrehajtási módjukkal* (*execution mode*) jellemezhetjük.

A tevékenységek közötti kapcsolatok elsőbbségi feltételekkel írhatók le. Ezek a feltételek az egymást követő tevékenységek kezdetének és végének a kapcsolatát fejezik ki. Így beszélhetünk *kezdet-kezdet, kezdet-vég, vég-vég, vég-kezdet* kapcsolatokról. A kapcsolatok közé az MPM (*Metra Potencial Method*) hálók esetében egy kivárási időt (*time lag*) is definiálhatunk, amely a fent említett kapcsolatok közé egy várakozási időt is meghatároz. Vég-kezdet típusú kapcsolat esetén ez azt jelenti, hogy a követő tevékenység csak akkor kezdődhet meg az előző tevékenység befejeződése után, ha a megadott kivárási idő már eltelt. A klasszikus erőforrás-korlátos projektütemezési feladatok (*Resource-constrained Project Scheduling* (*RCPSP*)) esetében ez a kivárási idő zérus. Abban az esetben, ha az érték negatív, akkor a tevékenységek bizonyos mértékű időbeli átfedéséről beszélhetünk.

A szakirodalomban különböző osztályozási szempontokat vezettek be a tevékenységek osztályozására. Alapvetően két nagy kategóriát különböztetünk meg, a *megszakítható* és a *nem megszakítható* tevékenységek csoportját.

2.1.1 Nem-megszakítható tevékenységek

Nem-megszakítható tevékenységnek nevezzük azt a tevékenységet, amelynek ha megkezdődött a végrehajtása, akkor azt megszakítás nélkül végre kell hajtani.

2.1.2 Megszakítható tevékenységek

Az erőforrás-korlátos projektütemezési feladatok modellezése során feltételezik, hogy a tevékenységek nem megszakíthatóak, ha már elkezdődött a végrehajtásuk. Számos kutató, köztük Brucker és Kunst (2001), Debels és Vanhoucke (2008), Demeulemeester és Herroelen (1996) munkáikban a gyakorlatban felmerülő problémákhoz alkalmazkodva, olyan eseteket vizsgáltak, ahol a tevékenységek diszkrét időpontokban, azaz a végrehajtási időtartamuk egész egysége után, megszakíthatóak.

Ballestin és ts. (2008) egy olyan változatát alkalmazza a megszakítható tevékenységeknek, amelyben egy adott tevékenységre vonatkozóan meghatározott számú megszakítást engedélyez. Munkájukban azonban csak olyan esetekre koncentrálnak, ahol a megszakítások száma mindössze egy.

Debels és Vanhoucke (2008) a megszakíthatóság fogalmát az úgynevezett „fast tracking” fogalmával bővítik, amelynek lényege, hogy egy tevékenységnek az a része, amely a megszakíthatóságból jött létre, nem szükséges, hogy szekvenciálisan kerüljön végrehajtásra, ezek a tevékenység részek időben párhuzamosan is végrehajthatók. A „fast tracking” alkalmazásával a megszakítható tevékenységek részei között eltávolítják az elsőbbségi feltételeket, melynek következtében a tevékenységek végrehajtási variációinak száma nő. Ezt az elképzelést az a

tapasztalat inspirálta, hogy a tevékenységeket gyakran bizonyos erőforrás csoportok hajtják végre, de a teljes projektmunkát több csoport végzi párhuzamosan.

2.1.3 Időben változó erőforrás-igényű tevékenységek

Alapértelmezés szerint az erőforrás-korlátos projektütemezési modellekben a tevékenységek erőforrás szükséglete rögzített érték. A gyakorlatban felmerült olyan problémák kezelése, ahol a tevékenységek erőforrásigénye időben változik. Hartmann (1999) egy valós gyakorlati orvosi kutatási projekt problémát vizsgált, amelyben a tevékenységek végrehajtásának utolsó időperiódusában volt csak szükség bizonyos erőforrásokra. Hasonló esetet szoftverfejlesztésre vonatkozóan publikált Drezet és Billaut (2008). Időperiódusonként minimális és maximális erőforrásigényeket definiáltak, valamint további erőforráskorlátot határoztak meg a napi munkaidőre vonatkozóan.

2.1.4 Előkészítési idejű (Setup time) tevékenységek

Vannak olyan gyakorlati problémák, ahol bizonyos tevékenységek végrehajtását meg kell, hogy előzze valamilyen előkészítési (ráhangolódási) folyamat az erőforrások tekintetében. Erre az előkészítésre fordított időt nevezik *setup time*-nak. Mika és ts. (2006, 2008) háromféle *setup time*-ot vezettek be. Az egyik típus a *sorrend független* (*sequence-independent*) *setup time*, ahol ez az idő csak a tevékenységtől és annak végrehajtásához szükséges erőforrástól függ. A második típus a *sorrendfüggő* (*sequence-dependent*) *setup time*, amely a tevékenységek végrehajtási sorrendjétől függően változik. Ez akkor fordul elő, amikor a megelőző tevékenység is ugyanazt az erőforrást igényli. A harmadik típus az *ütemezés-függő* (*schedule-dependent*) *setup time*, amely függ a

tevékenység erőforrásigényétől valamint az ütemezésben őt megelőző tevékenység erőforrás szükségletétől. Ez utóbbi koncepció beépült a több megvalósítási módú erőforrás-korlátos projektütemezés témakörébe is.

Drexl és ts. (2000) a több megvalósítási módú erőforrás-korlátos projektütemezés esetében alkalmazta a sorrendfüggő *setup time*-ot, amit *changeover time*-nak nevezett. Egy tevékenység *setup time (changeover time)* értéke adott megvalósítási módban függ a megelőző tevékenységtől és annak megvalósítási módjától is.

2.1.5 Több-megvalósítási módú tevékenységek

Az erőforrás-korlátos projektütemezési modell (*Resource-Constrained Project Scheduling* vagy *RCPSP* a közismert angol rövidítésnek megfelelően) feltételezi, hogy egy tevékenység csak egyféleképpen hajtható végre, amelynek meghatározott végrehajtási időtartama és erőforrás szükséglete van. A gyakorlatban azonban sokszor előfordul, hogy egy tevékenység többféleképpen is végrehajtható, s ennek következtében más-más erőforrásigénye és időtartama van. A több-megvalósítási módú esetben nemcsak megújuló, hanem nem-megújuló erőforrásokat is felhasználnak a projekt megvalósítása során.

Az első munka, amely a több-megvalósítási módú tevékenységek fogalmát bevezette Elmaghraby (1977) nevéhez fűződik. Számos publikáció született a több-megvalósítási módú tevékenységeket tartalmazó projektek modellezésével kapcsolatosan, ezek közé tartozik Alcaraz és ts. (2003), Bouleimen és Lecoq (2003), Hartmann (2001), Jarboui és ts. (2008), Józefowska és ts. (2001), valamint Özdamar (1999) munkái.

Számos kutató munkájában a több-megvalósítási mód érdekes változatai jelennek meg. Bellenguez és Neron (2005), a tevékenységek megvalósítási módját a tevékenységet végző munkaerő szakmai képességeinek szemszögéből vizsgálja. Ez egy speciális megközelítési módja a több megvalósítási módnak, ahol minden egyes mód megfelel a munkaerő egy lehetséges részhalmazának, amely halmaz elemei képesek az adott tevékenységet a megfelelő szakmai színvonalon végrehajtani.

Másik érdekes megközelítése a több-megvalósítási módnak Li és Womer (2008) munkájában jelenik meg, ahol a minőséget tekintik megvalósítási módnak. A megvalósítási módok jellemzésére egy minőségi ismérvet, mint komplex mutatót alkalmaztak. Az elképzelésüket egy ellátási lánc (supply chain) modellezésére használták fel, ahol minden csomópont rendelkezésre állását szimbolizálja a minőségi mérték.

Tiwari és ts. (2009) az előbb említett kutatókhoz hasonlóan minőségi oldalról közelítik meg a problémát. Az elképzelésük szerint egy tevékenység megkezdődik egy bizonyos megvalósítási módban, de ez a mód nem teszi lehetővé, hogy a tevékenység a kívánt minőségi szinten fejeződjék be. Az adott módban történő végrehajtást követnie kell egy újrafeldolgozási módnak, amely befejezi a tevékenységet. Ezt a megközelítést egy telekommunikációs cég számára készített projekt inspirálta, amelyben egy rendelkezésre álló kevésbé képzett munkaerő kezdi el a tevékenységet, majd egy megfelelő tudással rendelkező fejezi be. Véleményem szerint ezt a szemléletmódot szoftverfejlesztési projektek modellezésénél is lehet alkalmazni.

2.2 Erőforrások

Minden tevékenységnek, a projekt kezdetét és végét jelző úgynevezett látszat tevékenységek kivételével bizonyos mennyiségű erőforrásra van szüksége, végrehajtása során. Ezek az erőforrások lehetnek gépek, eszközök, emberek és szakmai tudásuk, nyersanyagok, félkész-termékek, természetes erőforrások, energia, információ, pénz, stb.

Az erőforrások mennyiségét *erőforrásegységekben* mérjük. Az *erőforrástípus* ugyanazt a tevékenységet ellátni képes erőforrásegységek halmaza. A szakirodalom az erőforrásokat különböző osztályokba sorolja, ezeknek az osztályoknak a száma, az utóbbi 10-15 év szakirodalmát áttekintve, a gyakorlati igényeknek megfelelően megnövekedett.

2.2.1 Alapvető kategóriák

Slowinski (1980) és Weglarz (1980,1981) két kategóriát definiált, a *megújuló* és *nem-megújuló* (*renewable* és *nonrenewable*) erőforrások kategóriáját, majd ezt bővítették a *kétszeresen korlátozott* (*doubly constrained*) erőforrások kategóriájával.

A *megújuló erőforrások* mennyisége időperiódusonként korlátozott. Egy adott időperiódusban zajló tevékenységhez hozzárendelt erőforrásegységek az adott tevékenység befejeződésekor azonnal felszabadulnak és más tevékenységekhez hozzárendelhetőkké válnak egy következő időperiódusban. Az erőforrásból rendelkezésre álló erőforrásegységek száma a felhasználás ütemének megfelelően nem csökken, az időperiódusonként rendelkezésre álló erőforrásegységek száma független az erőforrás felhasználástól, s a projekt végrehajtása során állandó.

A *nem-megújuló erőforrások* mennyisége a teljes projektre vonatkozóan korlátozott. Az erőforrás mennyisége a felhasználás ütemének megfelelően csökken. Egy tevékenység végrehajtása során felhasznált erőforrásegység nem rendelhető hozzá később más tevékenységekhez. Ilyen erőforrás például a projekt végrehajtásához rendelkezésre álló pénz, nyersanyag, energia stb.

A *kétszeresen korlátozott erőforrások* mennyisége időperiódusonként és a projekt egészére nézve is korlátozott. Talbot (1982) bebizonyította, hogy minden kétszeresen korlátozott erőforrás helyettesíthető egy megújuló és egy nem-megújuló diszkrét erőforráspárral. A bizonyítás egy transzformációs eljáráson alapult, melyet a tanulmányában részletesen ismertetett.

2.2.2 Egyéb erőforrás kategóriák

A kutatók nagyon sok új kategóriát hoztak létre a felmerült problémák kezelésére. A dolgozat a számos új kategória közül csak a téma szempontjából fontosnak ítélhetőket sorolja fel.

Részlegesen megújuló (*partially renewable*) erőforrások. Ezt a kategóriát Böttcher és ts. (1999) vezette be azzal a céllal, hogy időperiódusok részhalmazához tudjon elérhető erőforrásokat hozzárendelni. Szemléltetésképpen tekintsünk egy olyan munkaidő beosztás, amelyben egy alkalmazott a hét bizonyos napjain dolgozhat csak, például vagy hétfőtől péntekig vagy csak szombattól vasárnapig, de mind a kétféle munkarendben nem. Minden egyes nap egy időperiódus. Ennek megfelelően a hétfőtől péntekig terjedő időszak az időperiódusok egy részhalmaza, míg a hétvége, a szombat és vasárnap egy másik részhalmaz,

amelyhez biztosítani kell az erőforrásokat. A részlegesen megújuló erőforrásokat a három alapkategória általánosításának tekintik.

Zhu és ts. (2006) a részlegesen megújuló erőforrásokat beépítette a több-megvalósítási módú erőforrás-korlátos projektek problémakörébe.

Kumulatív erőforrások. Ezt az erőforrás kategóriát Neumann és Schwindt (2002) vezette be. Elgondolásuk szerint, egy termelési projekt esetében szükség lehet közbülső termékre, amelyet kivesznek a raktárból, vagy legyártanak, hogy a raktárba tegyék. Egy kumulatív erőforrást a kapacitásokkal és minimális raktárkészlet szinttel jellemezzük. Bartels és Zimmermann (2009) több-megvalósítási módú projekt esetén alkalmazza a kumulatív erőforrásokat. Egy autógyártási projekt mérnöki és tesztelési tevékenységeit modellezték. A tesztjárművet tekintették kumulatív erőforrásnak, mivel azt elő kell állítani, majd használják (felhasználják a kapacitását), később a töréstesztben összetörik (a projekt végére a teljes kapacitását felhasználják).

Dedikált erőforrások (dedicated resource). Bianco és ts.(1998) vizsgálták ezt a kategóriát. A dedikált erőforrások csak egyetlen tevékenységhez rendelhetők hozzá egy időben. A dedikált erőforrások olyan megújuló erőforrásnak tekinthetők, amelyekből csak egyetlen egy egység áll rendelkezésre egy időperiódusban.

Térbeli erőforrások (spatial resource). A térbeli erőforrás, melynek fogalmát de Boer (1998) vezette be, olyan erőforrásként definiálható, amelyet tevékenység csoportok használnak fel. A tevékenységcsoport első tevékenységének kezdetekor lefoglalásra kerül az erőforrás, és a csoport utolsó tevékenységének befejezésekor szabadul fel. Más csoportból származó olyan tevékenység végrehajtása, amely ugyanezt a lefoglalt

erőforrást igényli az adott időperiódusban tilos. Térbeli erőforrásokra lehet példa egy kikötő szárazdokkja, vagy konténerek, helyiségek, stb.

A térbeli erőforrások egy speciális változata a *szomszédos erőforrások (adjacent resources),* amely Duin és Van Der Sluis, (2006) munkájában jelent meg. Az új fogalom bevezetését az indokolja, hogy ugyanazon típusú erőforrások bizonyos egységeinek a többi egységhez képest fontos a fizikai elhelyezkedése is a tevékenységek végrehajtása során. Ilyen erőforrás lehet például egy reptéri beléptető pont (check-in desk) vagy egy több-processzoros számítógép processzorai vagy egy grid számítógépes környezet.

A *szinkronizáló erőforrások* (*synchronizing resources*) speciális változata a megújuló erőforrásoknak. Schwindt és Trautmann (2003) vezette be a megújuló erőforrások ezen új kategóriáját. A szinkronizáló erőforrás egységek lehetővé teszik egy csoporthoz tartozó tevékenységek szimultán, egyszerre történő indítását. Példaként egy írásbeli vizsgát (vagy évfolyam szintű dolgozatíratást) tekinthetünk, amelyen több hallgatót kell vizsgáztatni, mint amekkora befogadóképessége van egy tanteremnek. Ekkor a hallgatókat kisebb csoportokba szétosztva több teremben helyezzük el. Ebben a szituációban a vizsga egyedi tevékenységek csoportjának tekintendő, amely tevékenységek egy-egy hallgatói csoport vizsgáztatását jelentik. Mivel a vizsgának egyszerre kell elkezdődnie, mindegyik csoport esetén, az osztálytermeket szinkronizáló erőforrásként modellezhetjük.

A *több-funkciós erőforrás* (*multi-skill resource*) fő jellemzője (Neron, 2002) az úgynevezett erőforrás rugalmasság, ami azt jelenti, hogy egy több-funkciós erőforrás különböző erőforrásigényekhez rendelhető hozzá. Másképpen fogalmazva számos megújuló erőforrás funkcionalitásával

rendelkezik, ilyen lehet több szakmával, képzettséggel rendelkező dolgozók halmaza. Ezekhez a képzettségekhez (képességekhez) hierarchia is rendelhető.

Megszakítható erőforrások. Megszakítható erőforrásokról beszélünk, ha minden egysége megszakítható, vagyis visszahívható egy folyamatban lévő tevékenység végrehajtásából és átirányítható másik tevékenységhez és azután visszatérhet az előzőleg megszakított tevékenység végrehajtásához.

2.3 A projektek ábrázolása

A projekt gráfelméleti szempontból egy irányított, összefüggő, körutat nem tartalmazó gráf. Attól függően, hogy hogyan értelmezzük a gráf éleit és csomópontjait, két csoportot különböztethetünk meg, *a tevékenység-orientált* és az *esemény-orientált* hálókat.

A tevékenység-orientált hálóknál (*Activity on Node*, röviden *AoN* gráf) a gráf csúcsai maguk a tevékenységek és az irányított élek a tevékenységek közötti logikai (megelőző- rákövetkező) kapcsolatot fejezik ki.

Az esemény-orientált hálók (*Activity on Arc*, röviden *AoA* gráf) esetében a gráf élei az elvégzendő tevékenységeket, a csomópontok pedig az eseményeket jelképezik.

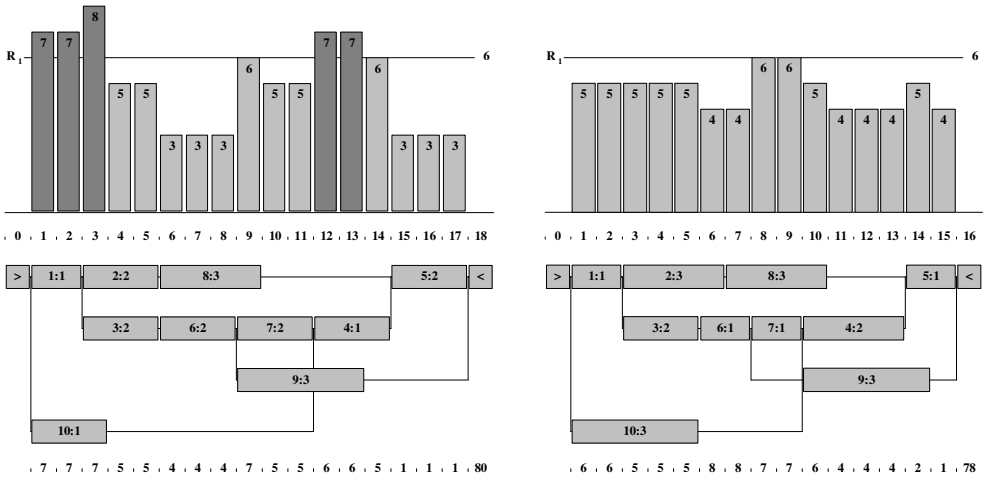
A két ábrázolási mód alkalmazását mindig a feladat természete határozza meg. Abban az esetben, ha a tevékenységek logikai sorrendje a fontos, célszerűbb a tevékenység-orientált, amennyiben az eseményeken, például pénzügyi ki- és befizetéseken van a hangsúly, célszerűbb az esemény-orientált ábrázolási módot használni. Természetesen vannak olyan problémák, ahol a kevert ábrázolási forma a legmegfelelőbb.

A mindennapi gyakorlatban más ábrázolási módszerek is elterjedtek, ilyenek a naptárnézet, erőforrás felhasználási hisztogram, időskálázott hálók, Gantt diagram. A legismertebb és legrégibb a Gantt diagram, amely az AoN gráfok speciális, időskálázott változata. A Gantt diagramban a tevékenységeket csúcspontok helyett téglalapok jelölik, a téglalapok hossza arányos a tevékenységek időtartamával. A téglalapok rendezetten helyezkednek el a vízszintes időtengely mentén, s a téglalapok bal széle a tevékenység kezdési időpontját, a jobb széle a befejezési időpontját reprezentálja. A tevékenységek közötti elsőbbségi feltételeket a téglalapokat összekötő nyilakkal jelöljük.

Az általunk modellezett több-megvalósítási módú projektek esetében a Gantt diagram módosított változatát használjuk a projektek ábrázolására. A módosítás lényege, hogy a téglalapokon feltüntetjük a tevékenység sorszámát és **:** jelet követően az ütemezésben a tevékenységhez rendelt megvalósítási mód sorszámát is. A projekt kezdetét és végét szimbolizáló látszattevékenységeket a **>** és **<** szimbólumokkal jelöljük. A látszattevékenységeknek nincs erőforrásigénye és időszükséglete. Ábrázolás-technikai okból a látszattevékenységeket egységnyi időszükséglettel ábrázoljuk, de ez a matematikai modell lényegét nem befolyásolja.

A következő ábra (1. ábra) ugyanazon több-megvalósítási módú projekt két különböző ütemezését szemlélteti, az első nem erőforrás-korlátos, a második erőforrás-korlátos ütemezés. A tevékenységeket tartalmazó Gantt diagram felett az erőforrás felhasználási hisztogramok láthatók. A projekt 10 tevékenységből áll, melyek egy megújuló erőforrást igényelnek, s minden tevékenységnek három megvalósítási módja van. Az ábra bal oldalán látható ütemezés nem erőforrás-korlátos, míg a jobb oldalon

látható ütemezés erőforrás-korlátos. Az ábráról leolvasható a tevékenységek megvalósítási módja is.



1. ábra Nem erőforrás-korlátos és erőforrás-korlátos ütemezés

2.4 Ütemezés, kritikus út

A tevékenységek időtartamának és logikai kapcsolatainak ismeretében a *kritikus út* (*Critical Path Method* (röviden *CPM*)) módszerével olyan ütemezés készíthető, amelynek *teljes időszükséglete minimális*.

A *CPM* módszer lényege, hogy megszámozzuk az egyes tevékenységeket, majd a legkorábbi kezdési időpontjaik alapján, az elsőbbségi feltételek betartása mellett előre haladva elvégezzük az ütemezést. Amikor megkaptuk a projekt befejezési idejét, akkor visszafelé haladva, a legkésőbbi kezdési időpontokat határozzuk meg.

Definíció

Az egyes tevékenységek legkorábbi (ES_i) és legkésőbbi (LS_i) kezdési idejének különbségét, $TF_i = ES_i - LS_i$ a tevékenység *tartalékidejének* nevezzük.

Definíció

Egy zérus tartalékidejű tevékenységet *kritikus tevékenységnek* hívunk. Ha egy tevékenység tartalékideje zérus, kezdésének késleltetése a projekt befejezésének ugyanolyan mértékű késleltetését eredményezi.

Definíció

Egy csupa kritikus tevékenységből álló, a kezdő tevékenységből a befejező tevékenységbe vezető utat *kritikus útnak* nevezünk.

2.5 A projektütemezés céljai

A tevékenységeket az ütemezésre vonatkozó elsőbbségi feltételek, valamint az erőforráskorlátok figyelembe vétele mellett valamilyen meghatározott cél szerint ütemezzük. A továbbiakban a számos cél közül elsősorban a disszertációhoz kötődő több-megvalósítási módú ütemezésekre vonatkozó célokat illetve a publikációkban leggyakrabban megfogalmazott célokat tekintjük át. Nem részletezzük külön az egy-megvalósítási módú projektek, a sztochasztikus, a folytonos modellek célfüggvényeit és nem foglalkozunk azokkal az esetekkel sem, ahol nem vég-kezdet kapcsolatok jellemzik a tevékenységeket. Nem tekintjük át a megszakítható tevékenységeket és megszakítható erőforrásokat tartalmazó modelleket sem.

A projektütemezés céljainak csoportosításához felhasználtuk Kolisch és Padman (2001), Demeulemeester és Herroelen (2002), Lancaster és Ozbayrak (2007), Hartman és Briskorn (2010) valamint Weglarz és ts. (2011) publikációit.

Az ütemezés céljai szerint különböző csoportok alakíthatók ki. A projektütemezés célja lehet időalapú, költség alapú, erőforrás felhasználásra vonatkozó vagy egyéb teljesítménymértékre vonatkozó mérték. A projektütemezés célfüggvényeit reguláris és irreguláris kategóriákba sorolhatjuk. A reguláris teljesítménymérték a tevékenység befejezési időpontjában monoton növekvő (minimalizáló esetben). A reguláris mértékre gyakran korai befejezés mértékként hivatkoznak.

Azok a mértékek, amelyek nem rendelkeznek a fenti tulajdonsággal, az irreguláris célfüggvények kategóriájába tartoznak. A nem reguláris mértékek esetében nem teljesül az a feltétel, hogy a célfüggvény értéke az ütemezendő tevékenységek befejezési időpontjaiban monoton növekvő. A reguláris és nem reguláris függvényekkel Neumann és ts. (2002) részletesen foglalkozott.

2.5.1 Idő alapú célfüggvények

Általánosságban kijelenthető, hogy az idő-alapú célfüggvények a reguláris célfüggvények közé sorolhatók. Egyedül a szimultán módon történő korai és késedelmes tevékenységek minimalizálása nem reguláris mérték.

Az idő alapú célfüggvények esetében a természetes teljesítménymérték a projekt teljes időszükséglete, a *makespan*.

Számos más idő alapú célfüggvényt is megfogalmaztak, ezek közül a *késési idő* illetve a *pontatlanság mérték súlyozott* vagy *átlagolt* értékének és különböző kombinációinak a minimalizálása a gyakran kutatott célok közé tartozik.

A *késési idő (lateness) mértéke* az adott tevékenység tényleges befejezési ideje (*completion time*) és az elvárt befejezési idő, vagy lejárati idő (*due date*) közötti különbség. Képletben megfogalmazva: $L_j = C_j - d_j$, ahol C_j illetve d_j a j-dik tevékenység befejezési ideje illetve elvárt befejezési ideje.

A *pontatlanság (tardiness) mértéke* hasonló mérték, de az értéke nem lehet negatív. A pontatlanság mértéke a következő kifejezéssel írható le: $T_j = \mathbf{max}\{0\,,\, C_j - d_j\}$.

A *koraiság (sietség)* (*earliness*) mértékét hasonlóan, az elvárt befejezési idő és a tényleges befejezési idő különbségével definiáljuk, $E_j = \mathbf{max}\{0\,,\, d_j - C_j\}$.

A fenti mértékek súlyozott összegének minimalizálása, vagy a maximális késések minimalizálása jelenik meg célfüggvényként Neumann és ts. (2002), Vanhoucke és ts. (2001) munkáiban.

2.5.2 Erőforrás alapú célfüggvények

Az erőforrás alapú célfüggvények az irreguláris célfüggvények családjába tartoznak. Az erőforrás-alapú célfüggvények magukban foglalják a projekt végrehajtásához szükséges erőforrások költségeit.

Az egyik legfontosabbnak ítélt példa ebben a kategóriában, az úgynevezett *erőforrás rendelkezésre állási költség* probléma (*resource availability cost problem*). A cél a megújuló erőforrás rendelkezésre állási költségének minimalizálása a projekt határidőre történő befejezése mellett. A problémakört Möhring (1984) vezette be erőforrás befektetési problémaként. Drexl és Kimms (2001) szintén erőforrás befektetési problémákat vizsgált.

Ebbe a célfüggvény csoportba sorolják az *erőforrás-kiegyenlítési* problémát, amelynek célja időkorlátos ütemezés létrehozása kiegyensúlyozott erőforrás felhasználás mellett. A megoldás minőségét az abszolút vagy négyzetes eltérési értékek összegével mérik, ahol ez az eltérés egy megadott erőforrásszinttől való eltérést fejez ki. Számtalan publikáció közül alapműnek tekinthető Burgess és Killebrew (1962) munkája.

A gyakorlatban fontos cél a teljes *erőforrás felhasználás ingadozásával* járó költség minimalizálása. Ez a probléma akkor keletkezik, ha egymást követő időperiódusokban bérelni kell munkaerőt, majd el kell bocsátani. (Brinkman és Neuman, (1996)).

Az *erőforrás beszerzésével* vagy *bérlésével* kapcsolatos optimalizálási feladatok is felmerülhetnek. Az ilyen típusú problémáknál a cél az, hogy minimalizáljuk a teljes bérleti vagy beszerzési költséget (Neumann és ts. 2002).

A több-megvalósítási módú erőforrás korlátos projektek esetében Akkan és ts. (2005) és Demeulemeester és ts. (1998) vizsgálták a diszkrét időtartam és költség összefüggés (*time/cost tradeoff*) problémák között a *nem-*

megújuló erőforrás alapú célfüggvényt, ahol a pénz volt a nem-megújuló erőforrás.

2.5.3 Költség alapú célfüggvények

Gyakorlati feladatok fordították a kutatók figyelmét a költség alapú mértékek felé. Kétféle költséget különböztetünk meg, a közvetlen a tevékenységek végrehajtásával kapcsolatos költségeket (*activity-cost*) és az indirekt költségeket, amelyek az erőforrások (*resource-cost*) felhasználása révén merülnek fel.

A tevékenységek kezdési időpontja és megvalósítási módja közvetlen költségeket határoz meg. Az ütemezés célja a projekt teljes költségének a minimalizálása, amely költség a tevékenységek költségeinek összege és az esetleg felmerülő kötbér, amennyiben a projekt nem fejeződik be megadott határidőre.

Az indirekt, erőforrásokhoz kapcsolódó költségek hangsúlyozása esetén az erőforrás kapacitásokhoz rendelt költségek minimalizálása az ütemezés célja.

2.5.4 Összefüggés (Trade-off) problémák

A több-megvalósítási módú tevékenységeket tartalmazó projektekben a tevékenység időtartamának és a tevékenységhez rendelt erőforrás felhasználásának kapcsolatát úgynevezett idő és erőforrás összefüggés (*time/resource tradeoff*) valamint idő és költség összefüggés (*time/cost tradeoff*) problémaként kezelik. Mindkettő speciális esete az időtartam versus erőforrás felhasználás modellnek. Az idő és erőforrás összefüggés

(*time/resource tradeoff*) és az idő és költség összefüggés (*time/cost tradeoff*) problémáknak létezik folytonos és diszkrét változata is.

Diszkrét idő és költség összefüggés (Time/cost trade-off) probléma (DTCTP)

Az idő és költség összefüggés (*time/cost trade-off*) probléma egyike azoknak a problémáknak, ahol a tevékenységeket több megvalósítási módban hajtjuk végre. A tevékenységek végrehajtási ideje nem növekvő függvénye a tevékenység végrehajtási költségének. A tevékenység végrehajtásának költsége aggregát összege az összes, a tevékenységhez rendelt nem-megújuló erőforrás felhasználásának. A legegyszerűbb esetben a nem-megújuló erőforrás a pénz. Megújuló erőforrások nincsenek.

Egy tevékenység végrehajtásának felgyorsítása, költségnövekedéssel jár, és fordítva a tevékenység végrehajtásának időben történő megnyújtása csökkenti a tevékenység költségét.

Összefoglaló erről a témáról Kolisch és Padmann (2001), Demeulemeester és Herroelem (2002) tanulmányában található.

Diszkrét idő és költség összefüggés (time/cost tradeoff) probléma diszkontált pénzárammal (cash-flow)

Erenguc és ts. (1993) volt az első, aki ezzel a feladat típussal foglalkozott. A cél meghatározni a tevékenységek időtartamát és egy ütemezést a tevékenységek kezdési idejére vonatkozóan úgy, hogy a nettó jelenérték (NPV) az összes pénzáramra nézve maximális legyen. A problémát vegyes egészértékű nemlineáris problémaként fogalmazta meg, megoldására az általánosított Benders dekompozíciós technikát használta fel.

Erőforrás-korlátos diszkrét idő és költség összefüggés (time/cost tradeoff) probléma diszkontált pénzárammal

Icmeli és Erenguc (1996) vezették be ezt az összefüggést. Modelljükben a tevékenységek időtartama csökkenthető a normál időtartamukhoz képest úgy, hogy több erőforrást allokálnak a tevékenységekhez. A tevékenységek időtartamának lerövidítési (crashing) költsége hozzáadódik a többi költséghez. Az ütemezés célja a projekt időtartamának és az összes pénzáram nettó jelenértékének minimalizálása az elsőbbségi és erőforrás-korlátok mellett.

Diszkrét idő és erőforrás összefüggés (time/resource trade-off) probléma

A *diszkrét idő és erőforrás összefüggés* problémát De Reyck és ts. (1998) vezették be a több-megvalósítási módú tevékenységeknél. Ebben az esetben a tevékenység végrehajtási ideje diszkrét nem növekvő függvénye a tevékenységhez rendelt megújuló erőforrás felhasználásának. Minden tevékenységnek meghatározott munkatartalma van. A tevékenységek többféle módon hajthatók végre, azonban csak olyan megvalósítási módok lehetségesek, amelyekkel a munkatartalomra vonatkozó érték teljesül. A tevékenységek végrehajtásához egyetlen erőforrás áll rendelkezésre. A tevékenység megvalósítási módjaihoz rögzített időtartam és erőforrásigény tartozik, s a két érték szorzata a munkatartalom értékét adja. Azok a megvalósítási módok a lehetséges módok, amelyeknél teljesül, hogy a szorzat értéke legalább megegyezik a munkatartalom értékével. A cél az, hogy találjunk tevékenység időtartamot (azaz módot) és egy ütemezést úgy, hogy a projekt teljes időtartama minimális legyen.

2.5.5 Több-célú projektütemezés

A legtöbb esetben a projekt időtartamának minimalizálása a cél, minden más, az ütemezésre vonatkozó kritériumot korlátozó feltételek segítségével adjuk meg. A gyakorlatban előforduló összetett problémák azonban megkövetelik, hogy egyszerre több szempontot vegyünk figyelembe az optimalizálás során.

A több szempont figyelembevételének egyik módja, hogy egyetlen célfüggvényben fogalmazzuk meg az összes optimalizálási szempontot, azok súlyozott összegeként. Ebben az esetben a fontosság szempontjából azonosnak tekintjük az optimalizálási kritériumokat. Másik mód az, amikor a célfüggvényeket egyfajta prioritási sorrend alapján adjuk meg, s először az elsődleges cél alapján végezzük az optimalizálást, majd a kapott megoldások halmazán végezzük el a másodlagos szempont szerinti optimalizálást és így tovább. Az ilyen fajta megközelítést lexikografikus vagy hierarchikus optimalizálásnak nevezzük. A disszertációban szereplő algoritmusunk is ebbe az utóbbi kategóriába tartozik, elsődleges célnak a több-megvalósítási módú projekt időtartamának minimalizálását tekintettük, másodlagos célnak az erőforrás felhasználási hisztogram kisimítását adtuk meg.

A súlyozott célfüggvények szerepelnek Nudtasomboon és Randhawa (1997) munkájában, akik a projekt időtartamát, a súlyozott pontatlanság, az erőforrás felhasználás kiegyenlítés és a nem-megújuló erőforrások felhasználására vonatkozó kritériumokat foglalták egyetlen célfüggvénybe, több-megvalósítási módú projektek ütemezésekor. Hasonlóan, Voß és Witt (2007) több-megvalósítási módú projekt ütemezésére olyan célfüggvényt fogalmazott meg, amelyben a projekt időtartama, a súlyozott pontatlanság

és az előkészítési (setup) költségek szerepeltek együttesen. Munkájukat egy acélipari termeléstervezési projekt inspirálta.

Bomsdorf és Derigs (2008) egy filmforgatási projekt ütemezésére vonatkozóan számos komponenst tartalmazó célfüggvényt fogalmazott meg. A komponensek számos különböző, a filmszakmára jellemző kritériumokat tartalmaztak, mint például a helyszínek időbeli változásának minimalizálása.

Al-Fawzan és Haouari (2005) a projekt időtartamának minimalizálását ötvözte a teljes szabad tartalékidő maximalizálásával egyetlen célfüggvénybe.

Az egyetlen célfüggvénybe tömörítés helyett egy másik módja a többkritériumos optimalizálásnak a Pareto–optimális ütemezések létrehozása. Számos kutató ezt a módszert alkalmazza. Davis és ts. (1992) a projekt időtartamát valamint minden egyes megújuló erőforrás túlfoglalkoztatásának minimalizálását fogalmazta meg az ütemezés céljának.

Viana és de Sousa (2000) az előbbi célt kiegészítette minden egyes nem-megújuló erőforrás túlfoglalkoztatásának, valamint a súlyozott pontatlanság átlagának minimalizálásával.

Slowinski és ts. (1994) több-megvalósítási módú projekt ütemezését végezte számos optimalizálási kritérium figyelembevételével, mint például a projekt időtartama, súlyozott késések átlaga, a pontatlan tevékenységek teljes száma, erőforrás felhasználási hisztogram simítása, teljes és súlyozott erőforrás felhasználás mértéke és a nettó jelenérték.

Nabrzynski és Weglarz (1999) egy tudásalapú megközelítést alkalmazott a több-megvalósítási módú projektek ütemezésére. A szerzők idő és költség kritériumok halmazát adták meg, amelyek többek között a késések súlyozott átlagát, a projekt időtartamát, az erőforrás felhasználás hisztogram simítását, teljes és súlyozott erőforrás felhasználási mértéket és nettó jelenértéket tartalmaz.

2.6 Az algoritmusok összehasonlíthatósága - a PSPLIB tesztkönyvtár

A kutatók által publikált eljárások hatékonyságának objektív összehasonlítására már az 1960-as években felmerült az igény. A kutatók úgy találták, hogy létre kell hozni tesztelési célokra olyan, tesztelésre alkalmas projektegyedeket (*benchmark instances*), amelyek nehézségi foka, bonyolultsága megfelelő mutatókkal jellemezhető, s minden kutató számára hozzáférhetővé téve, lehetőséget biztosít arra, hogy ugyanazokon a tesztadatokon ellenőrizzék saját algoritmusuk képességét.

Az első, széles körben alkalmazott, 110 darab erőforrás-korlátos projekt egyedet Patterson (1984) gyűjtötte össze és 1995-ig ez volt az egyetlen projekthalmaz, amit tesztelésre fel lehetett használni. Patterson rendszerezte az összegyűjtött adathalmazokat és megadta az optimális megoldásukat. A halmaznak azonban van egy hiányossága, mégpedig az, hogy nem lett kialakítva egy pontosan definiált kísérleti terv a paraméterek állíthatóságával kapcsolatosan.

Alvarez-Valdes és Tamarit (1989) egy új indikátort, a (*free float ratio*) (*FFR*) mértéket vezették be, amely azt fejezi ki, hogy milyen mértékben

mozgatható egy tevékenység anélkül, hogy más tevékenységek is késnének, s így a projekt időtartama megnőne, a projekt később fejeződne be. Az FFR értéke a [0,1] értéktartomány. Kimutatták, hogy ha FFR értéke kicsi egy projektre vonatkozóan, akkor az a nehéz esetekhez tartozik, míg a nagy, 1-hez közeli FFR érték a könnyű eseteket jelenti. Új, 144 teszt példánnyal növelték a tesztelhető projektek halmazát. Mindegyik egyedet előre meghatározott paraméter indikátorokkal hozták létre.

Kolisch, Sprecher és Drexl (1995) általánosították és bővítették az indikátorokat és létrehozták a PSPLIB (Kolisch és Sprecher (1996)) tesztkönyvtárat a ProGen projektegyedeket generáló program segítségével. A ProGen program számos paraméterrel rendelkezik, amelyek segítségével a paraméter beállításoknak megfelelő bonyolultságú projekteket lehet létrehozni. A projekteket csoportokba foglalták a paramétereknek megfelelően és minden csoportban 10 projektesetet hoztak létre.

Az egyes projektesetek létrehozása a következő beállítható paraméterek segítségével történik:

- $\left(J^{min}, J^{max}\right)$ – a tevékenységek minimális és maximális száma
- $\left(M_j^{min}, M_j^{max}\right)$ – a végrehajtási módok száma, ahol a j index a tevékenység sorszámára utal.
- $\left(d^{min}, d^{max}\right)$ – a j-dik tevékenység legrövidebb és leghosszabb időtartama
- $\left(R^{min}, R^{max}\right)$ – a megújuló erőforrásegységek minimális és maximális értéke

- $\left(N^{min}, N^{max}\right)$ – a nem-megújuló erőforrásegységek minimális és maximális értéke

- $\left(S_1^{min}, S_1^{max}\right)$ – a projekt kezdő tevékenységeinek minimális és maximális száma

- $\left(P_J^{min}, P_J^{max}\right)$ – a projekt befejező tevékenységeinek minimális és maximális száma

- $\left(S_j^{max}, P_j^{max}\right)$ – a j-edik tevékenységet megelőző és követő tevékenységek maximális száma

- (NC) – a hálózat bonyolultsága (a csomópontokhoz tartozó nem-redundáns élek számának átlaga)

- $\left(Q_\tau^{min}, Q_\tau^{max}\right)$ – a τ erőforrás kategóriák (megújuló, nem-megújuló) minimális és maximális száma egy-egy tevékenység-megvalósítási mód $[j,m]$ kombinációban, ahol $\tau \in \{R,N\}$

- $\left(U_\tau^{min}, U_\tau^{max}\right)$ – az r erőforrásból időegységenként felhasználható minimális és maximális mennyiségi szint és a teljes felhasználható mennyiség egy tevékenység-megvalósítási mód $[j,m]$ kombinációra, ahol $r \in \tau$, $\tau \in \{R,N\}$.

- RF_τ – a τ erőforrás kategóriák erőforrásainak erőforrásfaktora (*resource factor*), ahol $\tau \in \{R, N\}$. Ez a faktor az adott erőforrás átlagos erőforrásigényét fejezi ki. Értéke $[0,1]$ tartományba esik. Kolisch és ts. (1995) kísérletekkel igazolta, hogy RF értékének

növekedésével arányosan nő a projekt nehézségi foka is. Herroelen (2006) kimutatta, hogy RF értéke nagymértékben aszimmetrikussá válik, ha szignifikáns számban vannak a projektben erőforrást nem igénylő tevékenységek.

- RS_τ –az erőforrás erősségét (*resource strength*) fejezi ki az egyes erőforrás kategóriák erőforrásaira vonatkozóan, ahol $\tau \in \{R, N\}$. Az erőforrás erősségi mutató az igényelhető erőforrás alsó- és felsőkorlát értékek kombinációjának mértéke.

A fenti paraméterekkel létrehozott projektegyedeket a PSPLIB könyvtáron belül két nagy könyvtárban rendezték el. Az egyikbe az egy-megvalósítási módú projektesetek, a másik nagy könyvtárba a több-megvalósítási módú ProGen által létrehozott projektek kerültek. A disszertáció témája a több-megvalósítási módú projektekre korlátozódik, ezért a továbbiakban csak az utóbbi, több-megvalósítási módú projektek könyvtárával foglalkozunk.

A több-megvalósítási módú projektek generálása során 64 csoportot hoztak létre a ProGen paraméter beállításainak kombinációjára, s a lehetséges paraméter kombinációk halmazán 10 esetet generáltak. A csoportokat fix paraméter beállítással, alapparaméterek rögzítésével és bizonyos paraméterek hangolásával, illetve minden paraméter szisztematikus változtatásával hozták létre. A változtatható paraméterek közül az RF és RS értékeire A,B,C betűkkel megjelölt különböző értéktartományokat definiáltak.

A létrehozott projekthalmazok 10, 12, 14, 16, 18, 20 és 30 tevékenységet tartalmazó projektekből állnak, amelyeket a J10MM, …., J30MM névvel nevezték el. Az egyes halmazok rendre 536, 547, 551, 550, 552, 554 és

640 esetet tartalmaznak. Minden halmazban vannak könnyen megoldható projektek valamint nehéz és nagyon nehéz projektesetek is.

A J10MM és a J20MM csoportba tartozó projektekre ismertek az optimális megoldások, amelyeket egzakt algoritmusokkal állítottak elő. A J30MM halmazra nem ismert optimális megoldás, a 640 projektje közül 550 esetre találtak lehetséges megoldást. A J30MM halmaz esetében a beküldött lehetséges megoldásokhoz hasonlítva, illetve az elsőbbségi feltételeknek megfelelő alsó korlátoktól való eltérésre alapozva végezhető el az algoritmusok összehasonlítása. A *CPM* alsó korlát mindig jó összehasonlításnak bizonyul. Természetesen a beküldött megoldások halmaza dinamikusan változik.

A ProGen programmal az MRCPSP problémáknak számos variánsát is előállították. Számos kutató bővítette a teszthalmazt speciális feladatokkal. Schwindt (1995) közzétett egy tanulmányt, az új, ProGen/max projektgenerátorról, amely különböző erőforrás-korlátos projektütemezési problémákat generál minimális és maximális kivárási időkkel. A kivárási időket visszafelé haladó élekkel reprezentálja az AoN hálón. Az alkalmazás több-megvalósítási módú ütemezési problémák előállítására is képes, minimális és maximális kivárási idővel és megújuló valamint nem-megújuló és kétszeresen korlátozott erőforrásokkal. Drexl és ts. (2000) a Time/Cost tradeoff problémák projekteseteivel, a részlegesen megújuló erőforrásokat is tartalmazó MRCPSP feladatokkal bővítette a teszthalmazt és a Progen projektgenerátor kiterjesztését ProGen/πx névvel jelölték. Kiegészítették a feladatokat a megvalósítási mód-függő kivárási idővel (*mode-dependent time lags*), az átállási idő (*changeover time*), a tiltott periódusok (*forbidden periods*) és más projektjellemzőkkel.

A PSPLIB tesztkönyvtár a http://129.187.106.231/psplib/ web címen bárki számára elérhető, s ha a saját algoritmusával jobb megoldást talál valamelyik feladatra, mint amelynek az eredménye addig közlésre került, akkor az eredményeket az adott web címre feltöltheti. A tudományos publikációkban megjelent eredményeket erre a teszthalmazra vonatkozóan adják meg, összehasonlítás céljából. A disszertációban bemutatott metaheurisztikánkat a PSPLIB tesztkönyvtár halmazain teszteltük, a futási eredményeket is erre a teszthalmazra vonatkozóan adjuk meg.

3. A több-megvalósítási módú erőforrás-korlátos projektütemezési feladatok megoldási módszerei Szakirodalmi áttekintés

A több-megvalósítási módú erőforrás-korlátos projektütemezési probléma (*Multi-mode Resource-Constrained Project Scheduling Problem, röviden MRCPSP*) az RCPSP általánosítása, ahol minden tevékenység több végrehajtási móddal rendelkezik. A modellezés során a három alapvető erőforrás kategória, (Slowinski és ts. (1994)) a megújuló, a nem-megújuló és a kétszeresen korlátozott erőforrások rendelhetők hozzá a tevékenységekhez, illetve azok megvalósítási módjához. Minden erőforrás diszkrét. Hasonlóan az egy módú erőforrás-korlátos projektütemezési modellhez a tervezés időhorizontja időperiódusokra osztott, ami azt jelenti, hogy az idő diszkrét változó. Ha egy tevékenység elkezdődik valamelyik megvalósítási módban, akkor abban is kell befejeződnie. A megvalósítási mód megváltoztatása, vagy a tevékenység megszakítása nem megengedett.

A több-megvalósítási módú erőforrás-korlátos projektek ütemezésekor minden tevékenységhez egy kezdési időpontot és egy megvalósítási módot rendelünk hozzá. Az ütemezés leggyakoribb célja hasonlóan az egy-módú esetekhez, a projekt időtartamának (*makespan*) minimalizálása az elsőbbségi feltételek és az megújuló valamint a nem-megújuló erőforráskorlátok betartása mellett.

Az MRCPSP matematikai modelljét Talbot (1982) vezette be, s az általa bevezetett modell Pritsker és ts. (1969) egy-megvalósítási módú erőforrás-

korlátos projektütemezési modelljén alapszik. Az MRCPSP modellek Brucker és ts. (1999) jelölésrendszerével a $MPS \mid prec \mid C_{max}$, míg Herroelen és ts. (1999) osztályozási rendszerének jelölését alkalmazva $m,1T \mid cpm, disc, mu \mid C_{max}$ szimbólumokkal írható le.

Az MRCPSP NP-nehéz feladat, mivel általánosítása az önmagában is NP-nehéz RCPSP-nek. Sőt, több mint egy nem-megújuló erőforrás esetén lehetséges ütemezést találni már NP-teljes feladat. (Kolisch, (1995), Kolisch és Drexl (1997)).

A több-megvalósítási módú modellek az idő és költség összefüggés (*time/cost trade-off*) problémák bővítése, kiterjesztése során kerültek előtérbe. Elsőként Elmaghraty (1977) vizsgálta a különböző végrehajtási módokat és megújuló erőforrásigényeket a különböző ütemezési feladatokra, amelyekben egyetlen nem-megújuló erőforrás volt.

A több-megvalósítási módú erőforrás-korlátos projektütemezési probléma megoldási módszereivel foglalkozó irodalom rendkívül szerteágazó. A témában publikált összefoglaló tanulmányok közül a következőket emeljük ki: Hartmann és Drexl (1998), Weglarz és ts. (2011), Hartmann és Briskorn (2010), Kolisch és Padman (2001), a témával foglalkozó szakkönyvek közül Demeulemeester és Herroelen (2002), Neumann és ts. (2002), Artigues és ts. (2008).

A következő táblázatban összefoglaljuk a több-megvalósítási módú erőforrás-korlátos projektütemezési problémákkal foglalkozó tanulmányokban közölt megoldó eljárásokat és legfontosabb paramétereiket. A táblázat a Van Peteghem és Vanhoucke (2010) munkájában szereplő táblázat magyar nyelvű módosított változata.

2. Táblázat Az MRCPSP feladatok megoldó algoritmusai

Szerző	Év	Módszer	Adat-halmaz	Tevékenységek száma	M	R	NR
Slowinski	1980	LP	Saját	-	-	-	-
Talbot	1982	Leszámlálás	Saját	10, 20, 30	1-3	3	0
Patterson és ts.	1989	Leszámlálás	-	-	-	-	-
Speranza és Vercellis	1993	B&B	Saját	10-20	>=2	1-6	1
Boctor	1993	Heurisztika	Saját	50, 100	1-4	1,2,4	0
Drexl és Grünewald	1993	Heurisztika	Saját	10 / 10	2-4/ 2-4	3/3	1/3
Özdamar és Ulusoy	1994	Heurisztika	Saját	20-57	1-3	1-6	1-6
Slowinski és ts.	1994	Szimulált hűtés	Saját	30	2	3	3
Boctor	1996	Szimulált hűtés	Boctor (1993)	50,100	1-4	1,2,4	0
Boctor	1996	Heurisztika	Boctor (1993)	50,100	1-4	1,2,4	0
Sprecher és ts.	1997	B&B	PSPLIB	10	3	2	2
Mori és Tseng	1997	Genetikus algoritmus	Saját	20, 30, 40, 50, 60, 70	2-4	4	0
Kolisch és Drexl	1997	Heurisztika	PSPLIB	10, 30	3	2	2
Hartmann és Drexl	1998	B&B	PSPLIB	10, 12, 14, 16	3	2	2
Sprecher és Drexl	1998	B&B	PSPLIB / Saját	10, 12, 14, 16, 18, 20	3/1 - 5/3	2/1 - 5/2	2/1- 3/0
Özdamar	1999	Genetikus algoritmus	PSPLIB /Saját	10/90	3/2	2/2	2/2

Szerző	Év	Módszer	Adat-halmaz	Tevékenységek száma	M	R	NR
Knotts és ts.	2000	Heurisztika	Maroto Tormos (1994)	50	2	3	0
Nonobe és Ibaraki	2001	Tabulista keresés	PSPLIB	30	3	2	2
Jozefowska és ts.	2001	Szimulált hűtés	PSPLIB	10,12,14,16 18,20,30	3	2	2
Hartmann	2001	Genetikus algoritmus	PSPLIB	10,12,14,16 18,20,30	3	2	2
Bouleimen és Lecocq	2003	Szimulált hűtés	PSPLIB	10,12,14,16 18,20,30	3	2	2
Alcaraz és ts.	2003	Genetikus algoritmus	PSPLIB/ Boctor (1993)	10,12,14,16 18,20,30/ 50, 100	3/ 1-4	2/ 1,2,4	2/0
Zhang és ts.	2006	Részecske rajzás	PSPLIB	10,12,14,16,18, 20	3	2	2
Zhu és ts.	2006	B&C	PSPLIB	20,30	3	2	2
Lova és ts.	2006	Heurisztika	Boctor	50, 100	1-4	1,2,4	0
Jarboui és ts.	2008	Részecske rajzás	PSPLIB	10,12,14,16 18, 20, 30	3	2	2
Ranjbar és ts.	2008	Szórt keresés	PSPLIB	10,12,14,16 18, 20	1	2	2
Lova és ts.	2009	Genetikus algoritmus	PSPLIB/ Boctor (1993)	10,12,14,16 18,20,30/ 50, 100	3 /1-4	2/ 1,2,4	2/0

Jelmagyarázat: M= megvalósítási mód, R=megújuló erőforrások száma, NR=nem-megújuló erőforrások száma, B&C= Szétválasztás és Vágás, B&B=Szétválasztás és korlátozás. Forrás: Van Peteghem és Vanhoucke (2010) magyarra fordítva

3.1 Optimalizáló megközelítések

Az optimalizáló eljárások nagy része a *korlátozás és szétválasztás* (*Branch and Bound*) eljáráson, vagy annak valamilyen bővítésén alapszik. Az ütemezési probléma megoldásai a diszkrét keresési térben legtöbbször egy fastruktúrával leírhatók. A fa ágain végighaladva minden megoldás

megvizsgálható. A fa bejárásának egyik lehetséges módja a visszalépéses eljárás (*back-tracking*) alkalmazása. Az eljárás kiindulópontja a fa gyökere. Az algoritmus előre meghatározott módon halad végig a fa ágain. Ha eléri egy ág végét, a levelét, akkor megvizsgálja az adott megoldást, hogy lehet-e az a feladat optimális megoldása. Ha nem, akkor visszalép a legközelebbi elágazásig és egy másik ágat választva megkeresi annak legutolsó pontját. Itt is megvizsgálja a megoldást, majd visszalép az elágazási pontra. A módszer működéséből fakad, hogy nagyméretű feladatok esetén a fa bejárása nagyon sok időt igényel, ezért hatékonyabb algoritmusokat kellett kidolgozni. Ezek egyike a *korlátozás és szétválasztás* (*Branch and Bound*, röviden B&B) módszere. A módszer alkalmazhatóságának feltétele, hogy megadjunk bizonyosfajta korlátokat, megkötéseket a kereséssel kapcsolatban. A korlátozások segítségével szűkíteni tudjuk a keresési teret, vagyis kizárhatók a bejárásból azok az ágak, amelyek vizsgálata szükségtelen, nem ott található az optimum.

Talbot (1982) kétfázisú eljárást javasolt az MRCPSP megoldására. Az első fázisban a tevékenységeket, a megújuló erőforrásokat és módokat rendezi sorrendbe bizonyos szabályok alapján, hogy felgyorsítsa a leszámlálási eljárást, amire a második fázisban kerül sor. A második szakaszban egy prioritáson alapuló heurisztikát alkalmaz egy felső időtartam korlát meghatározására, melyet egy *visszalépéses korlátozás és szétválasztás* (*Branch and Bound with back tracking*) leszámlálási eljárás követ. A keresési fa minden szintjén az első, még nem ütemezett tevékenységet a rendezett listából az első működési módjában megkísérli ütemezni arra a legkorábbi befejezési időpontra, amit az elsőbbségi feltételek és az erőforráskorlátok lehetővé tesznek. Amennyiben az adott tevékenység működési módja elvetésre kerül amiatt, hogy nem teljesülnek a korlátok, a következő mód kerül kijelölésre a tevékenységhez. Ha az adott tevékenység nem ütemezhető a saját időablakában, akkor az algoritmus

visszalép a keresési fa előző szintjére, és megpróbál egy megelőző tevékenységet ütemezni, egy későbbi időpontra vagy a következő megvalósítási móddal. Az eljárás akkor fejeződik be, ha a visszalépések során visszajutunk a fa gyökeréig vagy az utolsó tevékenység (a fa levele) is ütemezésre kerül.

Patterson és ts. (1989) módosították Talbot eljárását és részletes számítási eredményeket közöltek. Az újításuk lényege az *elsőbbségi fa* (*precedence tree*) bevezetése volt. Megoldási módszerük hasonlóan két fázisból áll, az inicializálási fázisból és a leszámlálási fázisból. Az *inicializálási* fázisban a tevékenységek és végrehajtási módjaik bizonyos szabályok szerint kerülnek rendezésre, hogy egy jó kezdő ütemezést hozzanak létre. Ezen az ütemezésen alapulva, egy felső korlát kerül kiszámításra mindegyik tevékenység befejezési idejére vonatkozóan. A *leszámlálási* (*enumeration*) fázisban az elsőbbségi fa alkalmazásával biztosítják a hatékony keresést az összes elsőbbségi korlátnak megfelelő tevékenységi sorrend halmazon. A keresési fa minden szintjén csak egyetlen, az ütemezhető tevékenységek halmazából származó, tevékenység, kerül kiválasztásra egy bizonyos működési móddal. (Definíció szerint egy ütemezhető tevékenység olyan tevékenység, amelynek a megelőző tevékenységei már ütemezésre kerültek). Ezután kiszámolják a kiválasztott tevékenység erőforrás és elsőbbségi korlátokat kielégítő legkorábbi kezdési időpontját. A keresési fa bejárása hasonlóan történik, mint Talbot algoritmusában.

Ennek a megközelítésnek a teljesítménye 91 problémán került kipróbálásra, ahol a tevékenységek száma 10, 20, 30, 50, 100 és 500 volt. (Patterson és ts. 1990)

Speranza és Vercellis (1993) egy mélységi korlátozás és szétválasztás (*Depth-first Branch and Bound*) eljárást közölt, amely az aktív ütemezések

halmazán végezte a leszámlálást. A keresés hatékonyabbá tétele érdekében egy elsőbbségi feltételeken alapuló alsó korlátot (*precedence-based lower bound*) alkalmaztak a fa bizonyos ágainak lemetszésére. Hartmann és Sprecher (1996) kimutatták, hogy ez az algoritmus nem mindig képes optimális megoldást szolgáltatni olyan feladatokra, amelyben legalább két megújuló erőforrást használnak a projekt végrehajtása során.

Sprecher (1994) továbbfejlesztette Talbot és Patterson eljárását. Algoritmusában ő is az elsőbbségi kapcsolatok keresési fáját használta leszámlálási sémára. Bevezette az i-dik részütemezés megjelölést, amely egyértelműen jellemzi a leszámlálási fában az i-dik csomópontot és a hozzárendelt részütemezést. Ezenkívül, a szerző bevezetett négy dominancia és egy elfogadhatósági szabályt melyek a következők:

(i) Alap Idő Ablak szabály (*Basic Time Window Rule*)

(ii) Késleltetést tiltó szabály (*Non-delayability Rule*)

(iii) Nem-megújuló erőforrás szabály (*Nonrenewable Resource Rule*)

(iv) Lokális balra léptetési (*Local Left Shift Rule*)

(v) Egyszeri leszámlálási szabály (*Single Enumeration Rule*), amelyet tovább javított Hartmann és Drexl (1998) és Elsőbbségi fa szabálynak (*Precedence Tree Rule*) nevezett.

Sprecher és ts. (1997) egy szétválasztás és korlátozás (*Branch and Bound*) algoritmust tett közzé. Munkájukat Demeuleemester és Herroelen (1992) egy megvalósítási módú erőforrás-korlátos ütemezés (RCPSP)

megoldásánál bevezetett mód alternatíva koncepciója inspirálta, s a leszámlálási sémát mód és késési alternatívák bevezetésével oldották meg. A keresési fa minden csomópontjához rögzítenek egy időpontot (döntési pont), amelyben az egyes tevékenységek megkezdődhetnek. A keresési fa minden szintjén először a döntési pont értéke kerül kiszámításra, az éppen folyamatban lévő tevékenységek legkorábbi befejezési idejéből. Egy tevékenység akkor választható ki ütemezésre, ha minden megelőző tevékenységének befejezési ideje kisebb, mint ennek a döntési pontnak az értéke és a tevékenység folyamatban van.

A fő különbség ezen megközelítés és az elsőbbségi fa között az, hogy minden szinten több mint egy tevékenység ütemezhető, valamint az ütemezés ideiglenessége biztosítja, hogy az aktuális szinten visszavonhatók az előző szinten meghozott döntések.

Sprecher és ts. (1997) ugyanabban a tanulmányban bemutattak egy másik leszámlálási sémát, amit mód és bővítési alternatíváknak neveztek el. Ez nagyon hasonló a mód és késleltetés alternatívákhoz, azzal a különbséggel, hogy a bővítési alternatíva nem engedi meg, hogy visszavonjunk egy korábbi szinten létrehozott ütemezést.

Sprecher és Drexl (1998) algoritmusában az elsőbbségi fa leszámlálási sémán túl számos korlátozó szabályt alkalmaz, amelyet Sprecher (1994) vezetett be. Az algoritmust egy új korlátozó szabállyal, a Vágás halmaz (*CutSet*) szabállyal bővítették.

Hartmann és Drexl (1998) további két szabállyal egészítette ki Sprecher és Drexl (1998) megközelítését. A két új korlátozó szabály, a Sorrend Csere Szabály (*Order Swap Rule*) és a Közvetlen Kiválasztási Szabály

(*Immediate Selection Rule for Precedence Tree*), mely az elsőbbségi fára került bevezetésre.

Egy új egzakt megközelítés Zhu és ts. (2006) munkájában került ismertetésre. Bár az eljárásuk elsődlegesen a több-megvalósítási módú erőforrás korlátos részlegesen megújuló erőforrásokat tartalmazó ütemezési problémákra lett kifejlesztve, sikeresen lehet alkalmazni a klasszikus MRCPSP feladatokra is.

A többi algoritmusban széleskörűen alkalmazott korlátozás és szétválasztás módszer helyett, amely explicit leszámolja az összes (részleges) ütemezést, ők egy *szétválasztás és vágás (metszés)* (*Branch and Cut*, röviden:B&C) megközelítést alkalmaztak. Algoritmusukban az egészértékű lineáris programozási modell relaxációját alkalmazták arra, hogy alsó korlátot állítsanak elő a projekt időtartamára vonatkozóan a keresőfa minden pontján. Ha a keresőfa csomópontját nem lehet kibontani és a célfüggvény értéke valós (tört érték) az adott csomópontban, akkor az algoritmus megpróbál vágást találni, azaz olyan érvényes egyenlőtlenséget, amelyet a valós megoldás nem elégít ki, de minden, a keresőfa csomópontjához tartozó lehetséges egészértékű megoldás igen. Ha nem találnak vágást az adott csomóponthoz, akkor a szétválasztás módszerével új csomópontot hoznak létre a keresőfában.

A szétválasztás és vágás (B&C) algoritmusban alkalmazott szabályok négy csoportba sorolhatók:

- a változó számának csökkentésére vonatkozó szabályok
- szétválasztás és korlátozás szorosabbá tételére vonatkozó szabályok

- vágásokra vonatkozó szabályok

- magasabb szintű keresési stratégiákra vonatkozó szabályok.

Az algoritmus teljesítményének, hatékonyságának bizonyítására a jól ismert PSPLIB (Kolisch és Sprecher, 1996) tesztkönyvtár 20 és 30 tevékenységet tartalmazó több-megvalósítási módú erőforrás korlátos projektjein végeztek számításokat. Optimális megoldást találtak az összes 20 tevékenységet tartalmazó esetre és a 30 tevékenységet tartalmazó 552 eset közül 506 esetben is sikerült optimális megoldást kapni. A 30 tevékenységet tartalmazó projekteknél 5 esetben jobb megoldást adtak a projekt időtartamára, mint az addig ismert legjobb érték, és 23 esetben kaptak rosszabb megoldást, mint a PSPLIB-ben publikált értékek. Sajnálatosan, ezeket az eredményeket nem rögzítették a PSPLIB fájlokban, amelyek a legjobb ismert eredmények gyűjteményét tartalmazzák. A számítási idők a B&C eljárásra vonatkozóan a szerzők közlése szerint jobbak, mint a Sprecher és Drexl (1998) által közölt B&B eredmények, de figyelembe véve a processzorórát, a B&B gyorsabb, mint a B&C.

Az egzakt eljárásokkal kapcsolatos igen részletes összehasonlító elemzés Hartmann és Drexl (1998) munkájában megtalálható, kivéve Zhu és ts. (2006) munkáját.

3.2 Alsókorlátok

Az alsókorlátok alkalmazása lehetővé teszi, hogy a keresőfa felépítése után kizárjuk a további vizsgálatokból azokat a csomópontokat, amelyeknek a kifejtése nem vezet elfogadható ütemezéshez. Az első módszer az alsókorlátokra Talbot (1982) nevéhez fűződik. Ez az alsókorlát a nem

erőforrás-korlátos kritikus út hosszával azonos érték, ahol a legrövidebb végrehajtási módot rendelték hozzá a tevékenységekhez.

Pesch (1999) felhasználta Talbot (1982) B&B modelljét, mint alap algoritmust, hogy számos különböző stratégiát teszteljen. Elemezni kívánta a paraméter beállítások hatását a leszámlálási eljárások minőségére vonatkozóan, ehhez különböző probléma osztályokat definiált. A fő következtetés a munkájából az, hogy valószínűleg nem létezik egyetlen olyan eljárás, amely hatékonyan tudja megoldani a különböző osztályokba tartozó eseteket.

Egy destruktív alsókorlátot közölt Bruckner és Kunst (2003) a több-megvalósítási módú erőforrás korlátos projektütemezési problémákra minimális és maximális késleltetési idővel. Az alsókorlát számítások két metóduson alapulnak, a korlátpropagáló technika alkalmazásán, valamint egy oszlopgeneráló eljáráson.

3.3 Heurisztikák

Az MRCPSP feladatok NP-nehéz természete miatt a heurisztikus módszerek alkalmazásával igen jó, az optimálishoz közeli eredményeket lehet elérni. Az NP-nehéz feladatok megoldása nem végezhető el polinomiális időben. Mivel az ilyen problémák megoldási ideje általában a feladat méretének exponenciális függvénye, ezért olyan közelítő algoritmusok kidolgozása a cél, amelyek elfogadható időn belül adnak kielégítő megoldásokat, amelyek általában nem optimális, csak optimálishoz közeli megoldások.

Kolisch és Hartmann (1999) osztályozási szempontjai alapján a következő csoportosításban mutatjuk be a több-megvalósítási módú erőforrás-korlátos

projektütemezési feladatok megoldásánál alkalmazott heurisztikus algoritmusokat:

- Prioritási szabályon alapuló
- Egyéb, lokális keresésen alapuló heurisztikák
- Metaheurisztikák

3.3.1 Prioritási szabály alapú heurisztikák

A prioritási szabályok az úgynevezett ütemezés generáló mechanizmus számára a tevékenységek sorrendjét határozzák meg, amely alapján a generáló mechanizmus létrehoz egy ütemezést. A szakirodalom soros vagy párhuzamos ütemezést generáló sémákat különböztet meg.

Az ütemezés egylépéses (*single-pass*) vagy többlépéses (*multi-pass*) módszer alkalmazásával jön létre. Az egylépéses módszer egyetlen egy prioritási szabályt alkalmaz, míg a többlépéses módszer prioritási szabályok kombinálásával és ütemezési mechanizmusok felhasználásával hajtja végre az ütemezést. A szakirodalomban számos publikáció található az egy- illetve többlépéses prioritási szabályokon alapuló eljárásokról.

Talbot (1982) korábban már ismertetett egzakt eljárásában az első fázisban alkalmaz egy prioritási szabály alapú heurisztikát, a tevékenységlista újrarendezésére, mellyel egy jó kezdő megoldást biztosít a felsőkorlát előállításához. A nyolc szabály, amit Talbot közzétett a következő:

- Maximális átlagos tevékenység időtartam (MAX ADUR)
- Tevékenység időtartamának maximuma (MAX DUR)

- Minimális késői befejezési idő (MIN LFT)
- Maximális átlagos erőforrásigény (MAX RD)
- Minimális korai befejezési idő (MIN EFT)
- Minimális késői befejezési idő a legkisebb időtartammal csökkentve (MIN(L-D)),
- Véletlen és minimális késői befejezési idő csökkentve az átlagos tevékenység időtartammal (MIN(L-ADUR))

A kísérleti futások alapján, amit 100 projektre vonatkozóan végeztek el a legjobb eredményeket a minimális késői befejezési időn alapuló prioritási szabály alkalmazásával érték el. Mindegyik projekt 10 tevékenységet, 3 megújuló erőforrást és 1-3 megvalósítási módot tartalmazott.

Boctor (1993) egy módosított párhuzamos ütemezési sémát alkalmazott, ahol egy tevékenység akkor került a döntési halmazba, ha legalább egy végrehajtási módban erőforrás-korlátos volt. A tevékenységeket az MSLK-szabály alapján, a módokat a minimális időtartamuk figyelembevételével választották ki. Boctor 21 prioritási szabályt próbált ki és 5 heurisztika kombinációját ajánlotta, amelyek nagy valószínűséggel a legjobb megoldást adják.

Drexl és Grünewald (1993) egy sztochasztikus ütemezési módszert mutatott be, amit STOCOM-nak nevezett, s amely részleges (*suboptimal*) optimális megoldásokat talált az MRCPSP-re. A módszer az ütemezhető tevékenységek halmazából véletlenszerűen, egy valószínűségi érték alapján választja ki a tevékenységeket, valamint a hozzárendelt

megvalósítási módokat. A valószínűségi érték kétféle módon származtatott súlymérték, melyet vagy az összes ütemezhető tevékenység összes megvalósítási módjai közül a leghosszabb időtartamú végrehajtási módok kombinációjaként, vagy az ütemezhető tevékenységek legutolsó befejezési ideje alapján számolnak ki.

A számítási eredmények azt mutatják, hogy a módszer jól alkalmazható más, determinisztikus ütemezési szabályokkal is. Az eljárást bővítették olyan esetek vizsgálatára is, ahol az erőforrás szükségletek az idővel arányosan változnak.

Boctor (1996a) heurisztikus algoritmust közölt a kritikus út módszerre (CPM) alapozva, de sajnos az eljárás nem talált lehetséges megoldást a 240 tesztprobléma egyikére sem. A sikertelenség oka, hogy a heurisztika egyaránt létrehoz lehetséges, illetve nem lehetséges erőforrás-korlátos ütemezést, mivel a tevékenységek legkorábbi kezdési időpontjaikra történő ütemezésekor csak az elsőbbségi korlátokat vették figyelembe, s az erőforráskorlátok kielégítését csak a végső ütemezéskor ellenőrizték.

Özdamar és Ulusoy (1994) heurisztikus megközelítése, amit lokális korlát alapú elemzés (*local constraint based analysis*, (LCBA)) néven publikáltak, csak egy nem-megújuló erőforrást tartalmazó több-megvalósítási módú erőforrás-korlátos projektütemezési probléma megoldására készült. Ebben az algoritmusban a tevékenységek kiválasztása és a megvalósítási módok tevékenységekhez rendelése lokálisan történik minden döntési pontban. Az LCBA az elsőbbségi szabályokat alkalmazza a tevékenységekre, amely garantálja, hogy erőforráskorlátoknak megfelelő lehetséges aktuális időtartam jöjjön létre a döntési pontokban valamint a teljes tevékenység ütemezésére. A párhuzamos ütemezés generáló sémát (SGS), amit Brooks algoritmusnak is hívnak, (lásd Kolisch 1996) dekódoló

szabályként alkalmazzák. Az eljárást különböző bővített modellekhez adaptálták, mint például a rugalmas erőforrásigény szintek. Az eljárást 95 példányból álló halmazon tesztelték, 20-57 tevékenységgel, 1-6 megújuló és egy nem-megújuló erőforrással. A számítási eredmények, amelyeket az LCBA-ra kaptak azt mutatják, hogy jobb eredményeket kaptak, mint a prioritás alapú szabályok alkalmazása esetén.

Azonban, a korlát alapú megközelítés két előnytelen tulajdonsággal rendelkezik, az egyik, hogy az eljárás időigénye exponenciálisan nő a feladat komplexitásának függvényében, a másik hátránya, hogy nem alkalmas az MRCPSP megoldására, ha többszörösen szűkös nem-megújuló erőforrás szerepel a modellben.

3.3.2 Lokális kereső heurisztikák

Kolisch és Drexl (1997) lokális kereső stratégiájában a megoldást két listával, a mód hozzárendelési listával, valamint a tevékenységek befejezési idejének listájával reprezentálja.

A bemutatott modell három fázisból áll. Az első fázisban a tevékenységekhez egy kezdeti megvalósítási módot rendelnek hozzá, és azután, ha ez nem sérti a nem-megújuló erőforráskorlátokat, egy gyors heurisztikus eljárást alkalmaznak az eredményként keletkező RCPSP-re.

A második fázisban előre megadott iterációs lépésben lokális keresést valósítanak meg, amely egy egymenetes szomszédsági keresést hajt végre a korlátos megvalósítási mód hozzárendelési halmazon.

Végül, az utolsó fázisban egy jobb célfüggvény értékkel rendelkező ütemezést keresnek, az előző fázisban előállított, legjobb megvalósítási mód hozzárendeléssel rendelkező tevékenységek halmazán.

Az algoritmusnak teljesítményét a PSPLIB (Kolisch és ts. 1995) tesztkönyvtár 10 és 30 tevékenységet tartalmazó projekthalmazán mérték. Mindegyik halmaz 640 egyedet tartalmaz, ahol mindegyik projektben két megújuló és két nem megújuló erőforrás kerül felhasználásra, és a tevékenységeknek 3 megvalósítási módjuk van. A kapott eredményeket összehasonlították a csonkított korlátozás és szétválasztás (*truncated branch and bound*) (Talbot, 1982) és STOCOM (Drexl és Grünwald,1993) módszerrel. A közölt heurisztika az egyetlen a három vizsgált megközelítésből, amely talált lehetséges megoldást az összes 10 tevékenységet tartalmazó példányra és a legtöbb 30 tevékenységet tartalmazó példányra.

3.4 Metaheurisztikák

A kombinatorikus optimalizálás területéhez számos olyan probléma tartozik, amelyek nehezen megoldhatók, a megoldási idejük rendszerint a feladat méretének exponenciális függvénye, ezért az esetek döntő többségében nem lehet hatékony, egzakt algoritmust alkalmazni a megoldásukra.

Az NP-teljes problémák esetén olyan közelítő algoritmusok kidolgozása a cél, amelyek elfogadható időn belül adnak jó minőségű megoldásokat, amelyek az esetek többségében nem optimális megoldások. A több-megvalósítási módú erőforrás-korlátos ütemezési problémák NP-nehéz problémák, és ahogy a korábbi pontokban már említésre került, azok a

modellek, amelyek két vagy több nem-megújuló erőforrást tartalmaznak az NP-teljes problémákhoz tartoznak.

A disszertációban bemutatásra kerülő algoritmus is a metaheurisztikák közé tartozik, ezért tartjuk fontosnak részletesen foglalkozni az irodalomban publikált metaheurisztikákkal.

A metaheurisztikák először az 1970-es években jelentek meg, fejlődésük azóta is töretlen és egyre inkább kiterjednek számos problématerületre. A metaheurisztikus módszerek sikere annak köszönhető, hogy rugalmasak, könnyen alkalmazhatók, ezért számos problématípus esetén jó eredményeket biztosítanak.

A legsikeresebb metaheurisztikák közé tartoznak a

- Evolúciós algoritmusok (genetikus algoritmusok, genetikus programozás, evolúciós stratégiák)
- Tabulistás kereső algoritmusok (Tabu Search)
- Hegymászó algoritmus (Hill Climbing)
- Hangyaboly rendszerek (Ant Colony systems)
- Szimulált hűtés (simulated annealing)
- Szórt keresés (scatter search)

A metaheurisztikák közös jellemzői, hogy a keresés során megtalált megoldásokat memorizálják, valamilyen adatszerkezetben tárolják. Az evolúciós algoritmusok és a szórt keresés a populációkban őrzik meg a

keresés során kapott megoldásokat, a tabulistás eljárásoknál maga a tabulista, míg a hangyaboly rendszereknél a feromon-mátrix szolgál a korábbi megoldások tárolására. A memóriában tárolt információk felhasználásra kerülnek a további megoldások előállítása során. A metaheurisztikák önmagukban nem képesek jó közelítő megoldások előállítására, általában több klasszikus módszert ötvöznek, s ezek irányított alkalmazásával működnek.

3.4.1 Evolúciós algoritmusok

Az evolúciós algoritmusok alapelve a természetes evolúciós folyamatokon alapszik. Az evolúció egy rendszer fokozatos fejlődésének a folyamata. A természetben lezajló evolúciós folyamatok során a kiválasztás vagy szelekció, a keresztezés és a mutáció révén egyedek generációjából új egyedek keletkeznek, amelyek különböznek az eredeti egyedektől. A származás során kialakulnak olyan egyedek, amelyek életképesebbek más utódoknál, nagyobb eséllyel maradnak életben és nagyobb eséllyel szaporodnak, mint a gyengébb egyedek. Egy adott populációban a folyamat eredményeként a sikeres egyedek válnak meghatározóvá, a gyengébbek háttérbe szorulnak, a populáció minősége javul.

Az evolúciós algoritmusok is hasonló elven működnek, a természetes szelekció mechanizmusát szimulálják, alkalmazva a darwini elvet, a legrátermettebb túlélésének elvét.

Számos ismert probléma, mint a gráfszínezés, az utazó ügynök, a hátizsák feladat, megoldható evolúciós algoritmusokkal.

Az evolúciós algoritmusok leggyakoribb típusai a következők:

- Evolúciós stratégiák (ES)
- Genetikus algoritmusok (GA)
- Genetikus programozás (GP)

A továbbiakban ezek közül csak a genetikus algoritmusokat részletezzük, mivel ez a legnépszerűbb típusa az evolúciós algoritmusoknak.

3.4.1.1 Genetikus algoritmusok

A genetikus algoritmusok iránti érdeklődés oka széleskörű felhasználási lehetőségükben és viszonylagos egyszerűségükben rejlik. A probléma független metaheurisztikák csoportjába tartoznak.

A genetikus algoritmusok széles spektrumú optimalizáló eljárások, amelyek a keresési tér minél rátermettebb egyedeit hivatottak felfedezni. Az aktuális populációból minden lépésben egy új populációt állítanak elő úgy, hogy a véletlenszerű szelekciós operátor által kiválasztott legrátermettebb elemeken (szülőkön) alkalmazzák a rekombinációs és mutációs operátorokat. Az alapgondolat az, hogy mivel általában minden populáció az előzőnél rátermettebb elemeket tartalmaz, a keresés folyamán egyre jobb megoldások állnak majd rendelkezésre.

A genetikus algoritmusok által használt terminológiát a biológiából kölcsönözték.

A biológiában egy fajon belüli csoportot *populációnak* nevezünk. A populációt egyedek alkotják, amelyeket a *kromoszómájuk* határoz meg. A

kromoszóma *gének* sorozatából áll, amely az egyed tulajdonságainak utódokra való *átörökítését* biztosítja. Egy egyedben lévő gének összességét *genotípusnak* nevezzük. A *fenotípus* a genotípus megnyilvánulása az adott egyedben, és magában foglalja a szerzett tulajdonságokat is. A *keresztezés* és a *mutáció* során új egyedek jönnek létre.

A genetikus algoritmusok alkotóelemei

- A megoldás egy kromoszóma reprezentációja
- Kezdeti populációt létrehozó módszer
- Értékelő függvény vagy jóság (*fitnesz*) függvény, ennek a függvénynek a globális optimumát keressük
- Genetikus operátorok, amelyek a leszármazott összetevőit változtatják meg, a reprodukálás során
- Paraméterek (pl. populáció mérete)

Genetikus operátorok

A genetikus operátorokat három nagy csoportba sorolhatjuk. Az első csoportba tartoznak a *szelekciós* műveletek, amelyek valamilyen sztochasztikus módszer segítségével a legrátermettebb egyedeket választják ki a populációból. Ezekre a kiválasztott egyedekre használják a *rekombinációt*, amelynek során a szülők tulajdonságait ötvözve egy jobb tulajdonságokkal rendelkező (rátermettebb) utód jön létre. A leszármazottak más módon is létrejöhetnek a *mutációs* operátor használatával.

Szelekció

A kiválasztás a populációkon működik, így legtöbbször független a reprezentációtól. Legismertebb fajtái a következők:

- Rulett kerék (*roulette wheel selection*): a kiválasztás valószínűsége a rátermettség (fitnesz) értékkel arányos, amely szerint egy megoldás kiválasztásának valószínűsége annál nagyobb, minél nagyobb a rátermettsége a populáció rátermettségi átlagához képest. A kiválasztás úgy működik, mintha az egyedeket egy olyan rulettkerék egy-egy mélyedéséhez (cikkelyéhez) rendelnénk, ahol a mélyedés nagysága arányos az egyed rátermettségével. (Ezt az elvet alkalmazza az algoritmusunk „karmestere" is egy dallam repertoárból történő választásakor.)

- Rang szelekció (*rank selection*) Hasonló a rulett kerékhez, de a kiválasztás valószínűségét nem a rátermettségi függvény értéke határozza meg, hanem az egyed rangsorban elfoglalt pozíciója.

- Verseny szelekció (*tournament selection*) egy csoportot (rendszerint 2-7 egyedet) választunk ki a populációból és a fitnesz értékük alapján versenyeztetjük őket. A nagyobb fitnesz értékű egyed nagyobb valószínűséggel győz. Ha a kiválasztott egyedek száma kettő, akkor pár-verseny (*binary tournament selection*) szelekcióról beszélünk.

- Elitista (*elitist*) az eddig kapott legjobb egyedet mindig áthelyezi az új populációba.

Mutáció

A mutáció unáris operátor, egyetlen megoldást igényel, és ebből állít elő egy újat. A mutáció célja a populáció frissítése. Ezzel a művelettel biztosítható, hogy egy eljárás képes legyen kitörni a lokális megoldás környezetéből. A mutációt úgy oldják meg, hogy rendszerint a kromoszóma egy véletlen kiválasztott génje változik meg, vagyis a kiválasztott pozíción lévő allél értékét változtatjuk meg.

Rekombináció, keresztezés

Több szülőből állít elő egy újabb megoldást, a szülők kódjainak valamilyen kombinációja segítségével. A rekombináció célja, hogy a különböző egyedek tulajdonságait ötvözve egy jobb tulajdonságokkal rendelkező, esetleg rátermettebb egyedet hozzon létre.

Többféle rekombinációs operátor létezik, az egyik az egypontos keresztezés (1-*point crossover*), a másik az egyenletes keresztezés (*uniform crossover*).

Az egypontos keresztezésnél véletlenszerűen választunk egy kereszteződési pontot (*crossover point*) és a kapott két fél kódból új megoldást rakunk össze. A leszármazottak mindkét szülőtől örökölnek tulajdonságokat. A kereszteződési pont előtti tulajdonságokat az egyik szülőtől, a kereszteződési pont utáni tulajdonságokat a másik szülőtől.

Az egyenletes keresztezés a mutációkhoz hasonlóan a kiválasztott pozíciókon a kiindulási kódokat cseréli le. A mutációtól eltérően egy pozíció kiválasztásának valószínűsége 0,5, azaz átlagosan a kódok fele cserélődik le.

3.4.2 Szimulált hűtés

Kirkpatrick, Gelatt és Vecchi 1983-ban javasolták a szimulált hűtés (*Simulated Annealing*) módszert a kombinatorikus optimalizálási feladatok megoldására. A szimulált hűtés egy véletlenszerűen kiválasztott megoldásból indul ki és keres annak környezetében egy új megoldást. A keresést a hőmérséklet vezérli. Ha magas hőmérsékletű a rendszer, akkor képes elhagyni a lokális optimumot, bizonyos valószínűségi feltételek teljesülése esetén.

A szimulált hűtés a termodinamikai folyamaton alapszik. Magas hőmérsékleten a folyadék molekulái egymáshoz viszonyítva szabadon mozoghatnak. Ha a folyékony halmazállapotú anyagot lassan hűtjük, a hőenergia, vagyis a molekulák mozgási energiája csökkenni fog. A lassan hűtött rendszer esetén az anyag közel optimálisan rendezett atomi szerkezetet vesz fel. Ezen a természeti jelenségen alapul a szimulált hűtés algoritmusa.

A szimulált hűtés esetében a populáció egyelemű és a kereső operátor a mutáció. Az iterációk során, ha az új megoldás rosszabb, mint a régi, akkor is elfogadjuk bizonyos valószínűséggel, amit a *hőmérséklet* paraméter szabályoz. Ha a hőmérséklet nulla, akkor csak akkor fogadjuk el, ha nem rosszabb, mint a régi megoldás. A hőmérséklet a keresés folyamán fokozatosan csökken.

3.4.3 A tabulistás keresés

A tabulistás keresés iteratív javító algoritmus. A lokális kereső algoritmusok közé sorolható. A keresést a megoldás közvetlen környezetében található legjobb megoldás felé végezzük, még akkor is, ha az a vizsgált megoldásnál rosszabb. Olyan típusú feladatokra alkalmazható, ahol a megoldások egy gráf csomópontjain helyezkednek el. Mivel ennél a módszernél sok lehetőséget kell megvizsgálni, ezért meg kell őrizni, hogy melyik irányban járt már a keresés, és melyik irányban még nem. Ezt a problémát tabulista alkalmazásával oldjuk meg, amelyek tabu táblákból állnak. A tabulista alapján kiküszöbölhető, hogy ne járjuk be ugyanazt az utat, ahol már egyszer végighaladtunk.

A tabulista mérete az algoritmus szabad paramétere. A tabulistás keresés egyelemű populációt használ, és a kereső operátor a mutáció. Az új populáció kiválasztásához megnézzük, hogy szerepel-e a tabulistában. Ha igen, akkor elvetjük, egyébként, ha nem rosszabb, mint a régi, akkor elfogadjuk. Az új megoldás felkerül a tabulistára és a tabulista legrégebbi eleme törlődik.

3.4.4 Hangyaboly (Ant colony systems) rendszerek

A populáció alapú módszerek közé tartozik. A hangyaboly rendszerek is a természetben megfigyelt jelenségen alapulnak. A hangyaboly koncepció a táplálékukat kereső hangyák viselkedésén alapul.

A hangyák feromon hormon kibocsátásával képesek kommunikálni egymással. A hangyák a különböző táplálékforrásokhoz vezető útvonalakat

visszafelé haladva a bolyhoz feromon hormonnal jelölik meg. A feromon útvonalakat a többi hangya képes érzékelni, és nagy valószínűséggel követni fogja. Így a megjelölt útvonal feromon koncentrációja növekszik. A hangyák által kibocsátott feromon folyamatosan párolog, így az útvonal újabb bejárása nélkül folyamatosan csökken annak feromon szintje. Több út is vezethet egy táplálékforráshoz, a hangyák felismerik a legrövidebb utat, s mivel azon többször tudnak fordulni, annak feromon szintje növekszik, a hosszabb utaké pedig minimálisra csökken.

Az optimalizálási problémákra alkalmazott hangyakolónia metaheurisztikákban a mesterséges hangyák a valódi hangyák viselkedését utánozzák. A hangyák jó megoldásokat építenek fel valamilyen konstruktív heurisztika alapján. A megoldás építőelemeit feromon-mátrixokban tárolják. Az itt tárolt adatokat valószínűségi értékek alapján más hangyák figyelembe veszik, saját megoldásuk készítése során. A megoldás során a mesterséges hangyák a probléma-specifikus memórián túl saját memóriával is rendelkeznek, ahol a felépített megoldás minden fontos tulajdonságát tárolják.

A hangyaboly rendszerekben a bolyhoz tartozó hangyák egy keresési lépésben együtt állítják elő a megoldást a feromon értékek alapján. Az újabb lépéseket egy előre meghatározott lépésszám eléréséig folytatjuk, de az algoritmus leállítható akkor is, ha hosszabb ideig nem javul a legjobb megoldás.

3.4.5 A részecske rajzás (Particle swarm optimization)

A részecske rajzás optimalizáló módszert Kennedy és Eberhart dolgozta ki 1995-ben. A módszer a különböző állatrajok, halak, madarak természetben való vonulásának megfigyelésén alapszik. A részecske rajzás algoritmus bizonyos hasonlóságot mutat a genetikus algoritmusokkal és a hangyaboly rendszerekkel, de jóval egyszerűbb azoknál, mivel nem alkalmaz mutációs illetve keresztező operátorokat vagy feromont. A részecske rajzás valós véletlenszámokat és globális kommunikációt használ a raj elemei között. Nincs kódoló és dekódoló eljárás a bináris sztringgé alakításhoz, mint a hangyaboly algoritmusnál.

Az algoritmus a keresési térben az egyedi részecskék trajektóriájának változtatásával keresi az optimális megoldást kvázi-sztochasztikus módon. A raj részecskéjének mozgását két fő komponens határozza meg: egy sztochasztikus komponens és egy determinisztikus komponens. Mindegyik részecske az éppen aktuális legjobb globális pozícióhoz és az ő saját legjobb korábbi helyzetéhez vonzódik, miközben véletlenszerűen mozog.

Amikor egy részecske egy jobb pozíciót talál bármelyik korábbi megtalált pozíciójához képest, azt lecseréli a jobb, újabb pozícióra. Minden iterációs lépésben van aktuális legjobb pozíciója minden részecskének. A cél az, hogy megtalálják a legjobb globális megoldást az aktuális legjobb megoldások közül, adott iterációs lépés után. Az algoritmus előre meghatározott időkorlát alapján leállítható.

A továbbiakban a szakirodalom alapján áttekintjük az előbbiekben röviden ismertetett módszereken alapuló megoldásokat.

3.4.6 A több-megvalósítási módú erőforrás-korlátos projektütemezési feladatok metaheurisztikái

Özdamar (1999) kétféle genetikus algoritmust tett közzé:

- Tiszta genetikus algoritmus - PGA
- Hibrid genetikus algoritmus - HGA

A HGA –ban a megoldást két lista képviseli, az egyik a mód hozzárendelési lista, a másik a prioritási szabályok listája. A második listán szereplő *i*-dik elem jelöli a prioritási szabályt, amit az *i*-dik ütemezési döntésnél használunk fel. A következő prioritási szabályokat használják a döntések során: MIN LSK, MIN LFT, SPT, RAND, WRUP, MIN LST, MIN EST, MIN EFT és MTS.

Az algoritmus, dekódolási szabályként, egy előre-hátrahaladó ütemezést párhuzamos SGS alkalmazásával hajt végre. A két-pontos keresztezés és az egyenletes keresztező operátorok egyaránt alkalmazásra kerülnek. Egy mutációs operátor véletlenszerűen változtat egy pozíciót a mód hozzárendelési listában és egy pozíciót a prioritási szabályok listában.

A PGA-ban egy megoldást szintén két lista, a hozzárendelési lista, és a tevékenység lista reprezentál. Dekódoló szabályként a soros SGS kerül alkalmazásra. Az eljárás a kétpontos lineáris keresztezés alkalmazásával hoz létre leszármazott (*offspring*) megoldásokat. Ha a leszármazottban a tevékenységlista nem elégíti ki az elsőbbségi feltételeket, egy javító eljárás fut le. A PGA mutációs operátora véletlenszerűen változtat egy pozíciót a végrehajtási mód hozzárendelési listában és egy párcsere (*pair-wise*) történik két, a tevékenységlistán szereplő tevékenység között, azonban olyan csere, ami az elsőbbségi korlátoknak nem felel meg, nem lehetséges.

Mind a HGA, mind a PGA algoritmusban, az eljárás ellenőrzi az eredményt, s ha a leszármazott nem elégíti ki valamely nem-megújuló erőforráskorlátot, akkor nem fogadja el. Csak a nem-megújuló erőforráskorlátot kielégítő végrehajtási mód hozzárendelésekkel rendelkező megoldások a megengedettek. Az algoritmust 536 projekten tesztelték, amelyek 10 tevékenységet tartalmaztak, minden tevékenységnek 3 megvalósítási módja volt és két megújuló és két nem-megújuló erőforrást használt fel. Az algoritmust kipróbálták még 32 olyan projekt egyeden, amelyben 90 tevékenység, tevékenységenként két megvalósítási mód és két megújuló és két nem-megújuló erőforrást használtak fel. Az összes példányt a ProGen algoritmussal hozták létre a PSPLIB könyvtárban. Az eredményeket összehasonlították Kolisch és Drexl (1997) által publikált eredményekkel és azt tapasztalták, hogy a HGA felülmúlja a többi algoritmust a vizsgált esetekben.

Hartmann (2001) genetikus algoritmust hozott létre. Az algoritmus futását megelőzően egy elő-feldolgozás történik, amely arra hivatott, hogy csökkentse a keresési teret. Az elő-feldolgozást követően az algoritmus egy kezdeti populációt állít elő, majd meghatározza a fitnesz értékeket. A megoldást egy elsőbbségi korlátokat kielégítő tevékenységlista és egy végrehajtási mód hozzárendelési lista reprezentálja. Az algoritmus alkalmazása során létrejöhetnek olyan ütemezések is, amelyek a nem-megújuló erőforráskorlátokat megsértik. Az ilyen esetekben egy büntető függvény alkalmazásával számítják ki a nem-lehetséges megoldás jósági (fitnesz) függvény értékét. A következő populációt egypontos keresztezéssel és mutációval állítja elő az algoritmus, ami két szomszédos tevékenységlistán szereplő tevékenységet cserél fel (ha azok az elsőbbségi korlátoknak megfelelnek) és véletlenszerűen megváltoztat egy, a megvalósítási módok listáján szereplő módot. Az eljárás a rangsor módszert alkalmazza szelekciós operátorként. Az ütemezést a soros SGS

alkalmazásával végzi az algoritmus, és egylépéses vagy többlépéses lokális keresést hajt végre, amely a több-megvalósítási módú balra léptetés műveleten alapszik. Ha a kapott ütemezés jobb lett, mint egy korábbi ütemezés, akkor egy kódoló szabályt alkalmaznak, amellyel az új ütemezésnek megfelelő tevékenység és mód hozzárendelési lista jön létre.

Az algoritmust a PSPLIB teszthalmaz 10, 12, 14, 16, 18, 20 és 30 tevékenységet tartalmazó egyedeire próbálták ki, s a futások során finomhangolással az eljárás paramétereit a tapasztalatoknak megfelelően állították be. A legjobb beállításokkal végzett futtatások eredményeit összehasonlították a más megközelítésekkel, nevezetesen Kolisch és Drexl (1997), Özdamar (1999) genetikus algoritmusával, Bouleimen és Lecocq (2003) szimulált hűtés és Hartmann és Drexl (1998) levágó korlátozás és szétválasztás (*Truncated Branch and Bound*) módszerének eredményeivel. Az eredmények alapján megállapítható volt, hogy az új genetikus algoritmus egyértelműen jobb teljesítményt mutat, mint a többi heurisztika.

Alcaraz és ts. (2003) genetikus algoritmusa hasonlóan másokéhoz tevékenység és végrehajtási mód hozzárendelési listát kezel. Az alkalmazott jóság (fitnesz) függvény jobbnak mutatkozott, mint a Hartmann algoritmusában alkalmazott függvény. Hartmanntól eltérően az algoritmus nem alkalmazza a több-megvalósítási módú balra léptetést. Úgynevezett kétpontos előre-hátrahaladó keresztezéssel hozza létre az utódot, vagy az ütemezés elejéről (*head*) vagy a végéről (*tail*), attól függően, hogy a szülő előre/hátra génje milyen értékű.

A futási eredmények összehasonlítását más kutatók eredményeivel, mint Kolisch és Drexl (1997), Hartmann (2001) és Özdamar (1999) csak a PSPLIB teszthalmaz 10 tevékenységet tartalmazó eseteire végezték el. Az eredmények azt mutatják, hogy az algoritmusuk ezekre az esetekre a

lokális keresési eljárásnál jobb teljesítményt mutatott, míg Hartmann genetikus algoritmusánál valamivel gyengébbet.

Szimulált hűtés algoritmust fejlesztett ki az MRCPSP problémák megoldására Józefowska és ts. (2001), valamint Bouleimen és Lecocq (2003).

Jozefowska és ts. (2001) két változatát is elkészítette a szimulált hűtés eljárásnak, egy büntetőfüggvény nélküli és egy büntetőfüggvényes változatot. A megoldást két lista reprezentálja hasonlóan más eljárásokhoz, egy tevékenység és egy megvalósítási mód lista. Dekódolási szabályként a soros SGS-t alkalmazta. A büntetőfüggvényes változatban, abban az esetben, ha az ütemezés megsérti a nem-megújuló erőforrás-korlátokat, akkor egy büntető értéket ad hozzá az ütemezés időtartamának (*makespan*) felső korlátjához. Az algoritmus a szomszédos megoldásokat a kiválasztott tevékenység véletlen léptetésével és/vagy a kiválasztott megvalósítási mód véletlenszerű megváltoztatásával hozza létre. A keresési folyamatot egy adaptív hűtési séma vezérli. Mindkét algoritmus változatnál egy elő-feldolgozással kezdődik a folyamat. Az algoritmusok ugyanazokon a teszthalmazokon kerültek kipróbálásra, mint Hartmann genetikus algoritmusa, a tevékenységeknek három megvalósítási módja volt és két megújuló és két nem-megújuló erőforrás került felhasználásra. Az eredmények alapján a büntetőfüggvényes változat bizonyult jobbnak és az algoritmus versenyképes volt Hartmann genetikus algoritmusával.

Bouleimenn és Lecocq (2003) szintén egy szimulált hűtésen alapuló metaheurisztikát hozott létre. Publikációjukban mind az egy-megvalósítási módú erőforrás-korlátos, mind a több-megvalósítási módú erőforrás-korlátos projektekre kidolgozott eljárásukat ismertetik. Ugyanazt a dekódoló szabályt alkalmazták, mint Jozefowska és ts., (2001). A

megoldás reprezentálására egy tevékenység és egy megvalósítási mód listát alkalmaztak, azzal az eltéréssel, hogy erőforrás-korlátos megvalósítási módot engedtek csak meg. A teljes ütemezést a két előbb említett listán kívül egy harmadik lista segítségével, a kezdési időpontok listájával egészítik ki, s ebből számolják a projekt időtartamát. Az algoritmus a szomszédos megoldásokat kétfázisú eljárással generálja. Az első fázisban a véletlenszerűen kiválasztott tevékenységet olyan tevékenységre cseréli az algoritmus, amely kielégíti az erőforráskorlátot. Ezt követően rögzítik a tevékenységhez rendelt erőforrás-korlátos megvalósítási módot és eltávolítják a nem-megújuló erőforráskorlátot. Ezután egy jobb ütemezést keres az eljárás, ennek érdekében egy szomszédos tevékenységlistát generál, véletlen léptetési művelet alkalmazásával. A keresési eljárást egy hűtési séma vezérli. Az „újramelegítés" abban az esetben megengedett, ha lokális optimumba való „beragadás" veszélye áll fenn. A lokális optimum olyan megoldás, amelytől kicsit eltávolodva mindig rosszabb megoldásokat kapunk, azonban nagyobb távolságra jobb megoldások is létezhetnek. Ha az új megoldás rosszabb, mint a régi, akkor is elfogadjuk egy bizonyos valószínűséggel, amit a hőmérséklet paraméter szabályoz.

Az algoritmus eredményeit csak az egy-megvalósítási módú erőforrás-korlátos projektekre vonatkozóan hasonlították össze más megközelítésekkel. A több-megvalósítási módú projektekre összehasonlítást nem végeztek más eljárásokkal, de eredménynek tartják, hogy a 30 tevékenységből álló eseteknél 32 projektegyedre jobb megoldást találtak az addig ismert legjobb eredményeknél, ugyanakkor 38 esetben rosszabb eredményeket kaptak.

Az első tabulistás kereső megközelítés a több-megvalósítási módú erőforrás-korlátos projektekre vonatkozóan Nonobe és Ibaraki (2002) tanulmányában került ismertetésre. A teszt eredményeket a PSPLIB

teszthalmaz 30 tevékenységet tartalmazó projektjein végzett futtatásokra alapozva közölték. Összehasonlítást más eljárásokkal nem végeztek.

A metaheurisztikák közül a hangyaboly rendszerek alkalmazása a projektütemezési problémákra szintén felkeltette a kutatók érdeklődését. Chyu és ts. (2005) valamint Chiang és ts. (2008) mutattak be hangyaboly algoritmusokat.

Chyu és ts. (2005) egy hibrid hangyaboly optimalizáló eljárást (H-ACO) publikált, melyben a szétválasztás és korlátozás (B&B) és a hangyaboly optimalizálás (*ant colony optimization*, ACO) módszerét együtt alkalmazzák. Először a szétválasztás és korlátozás módszerével egy lehetséges (a nem-megújuló erőforráskorlátot kielégítő) megvalósítási mód halmazt állítanak elő. Ezután ezen a halmazon egy *CPM* alapú ütemezést hajtanak végre, majd az AON gráfon élek hozzáadásával az erőforrás konfliktusok kiküszöbölését végzik. Ezt követően kiszámolják a projekt időtartamát, majd a hangyaboly algoritmust alkalmazzák minden potenciálisan jó eredményt adó végrehajtási módú tevékenységen az előre-hátrahaladó javító (*forward-backward improvement*) eljárással együtt. A legvégén minden végrehajtási módra kiszámolt legjobb ütemezést (ahol a legjobb az erőforrás-korlátos legrövidebb időtartamú ütemezést jelenti) összehasonlítják és azok közül a legjobbat tekintik a probléma megoldásának. Az eredményeket Jozefowska és ts. (2001) szimulált hűtés algoritmusával hasonlították össze és ennek eredményeként megállapították, hogy a H-ACO eljárás hatékonyabban működik, mint a szimulált hűtés.

Chiang és ts. (2008) ACO-MRCPSP néven tett közzé egy hangyaboly optimalizáló algoritmust. A megoldás reprezentációja egy végrehajtási mód hozzárendelési lista és egy véletlen kulcs lista segítségével történik.

Az algoritmus négy fázisból áll. Az algoritmust a PSPLIB tesztkönyvtár 10, 20 és 30 valódi tevékenységet tartalmazó halmazán vizsgálták. Az eredmények azt mutatják, hogy hatékonyabb az algoritmus, mint Jozefowska, Özdamar és Alcaraz algoritmusa.

Egy másik heurisztikát, a részecskerajzást (*particle swarm optimization*, PSO) alkalmazta Zhang és ts. (2006) a több-megvalósítási módú erőforrás korlátos projektek ütemezési problémájának megoldására. A megoldás reprezentációja részecske pár pozíciójának megadásával történik. A pár egyik eleme a tevékenység prioritás kombinációkat írja le, míg a másik a végrehajtási módok kombinációját. A legjobb részecske sebességét és helyét figyelembe véve új pozíció és sebesség kerül meghatározásra a részecske pár felhasználásával. A modell mind a megújuló, mind a nem-megújuló erőforráskorlátokat figyelembe veszi, így az algoritmus kizárja azokat a lehetséges megoldásokat, amelyek megsértik a nem-megújuló erőforráskorlátokat. Ez a vizsgálat csökkenti a keresési tér méretét, az algoritmus sebessége javul. A részecskék által reprezentált lehetséges megoldást áttranszformálják ütemezéssé, amelyben a tevékenységekhez a megvalósítási módjuknak megfelelő erőforrásokat is hozzárendelik. Az algoritmust a PSPLIB könyvtár 10, 12, 14, 16, 18 és 20 tevékenységet tartalmazó projekt esetein tesztelték. A projektekben lévő tevékenységek 2 megújuló és 2 nem-megújuló erőforrást használtak fel. A tevékenységek megvalósítási módjainak száma 3. A futási eredményeket Sprecher és Drexl (1998) B&B algoritmusával, Bouleimen és Lecocq (2003) szimulált hűtés és Hartmann (2001) genetikus algoritmusával hasonlították össze. Az eredmények azt igazolták, hogy az algoritmus jobb teljesítményt mutatott a B&B algoritmusnál.

Kombinatorikus részecskerajzás (*Combinatorial Particle Swarm optimization* (CPSO)) algoritmust ismertet Jarboui és ts. (2008)

tanulmányában. A CPSO algoritmus első fázisában megvalósítási módot generál a tevékenységekhez. Ezt követően egy lokális kereső eljárás optimalizálja a tevékenységek végrehajtási sorrendjét. Minden részecske egy döntést reprezentál a megvalósítási módok tekintetében. A nem-megújuló erőforráskorlátok megsértését vizsgálja az algoritmus és egy büntető függvény alkalmazásával tereli a keresési folyamatot a lehetséges megoldások irányába. Az algoritmust a PSPLIB tesztkönyvtár 10, 12, 14, 16, 18 és 20 tevékenységet tartalmazó projektesetekre vizsgálták. Minden tevékenységnek három megvalósítási módja volt, s két megújuló és két nem-megújuló erőforrást használtak fel. A minta projekteket az erőforrás faktor (*resource factor*) és az erőforrás erősség (*resource strength*) paraméterek változtatásával állították elő. A 10, 12 és 14 tevékenységet tartalmazó projektesetekre optimális megoldást találtak. A többi projekteset futási eredményei közel volt Zhang eredményeihez.

3.4.7 A harmóniakereső algoritmus

A harmóniakereső (Harmony Search (HS)) algoritmus egy új, a metaheurisztikák csoportjába tartózó algoritmus, amelyet Lee és ts. (2005) fejlesztettek ki elsősorban mérnöki optimalizálási feladatok megoldására. Számos gyakorlati problémánál bizonyult eredményesnek, mint például a vízellátási hálózatok, földalatti vizek modellezése, energia megtakarítási rendszer, szerkezettervezés.

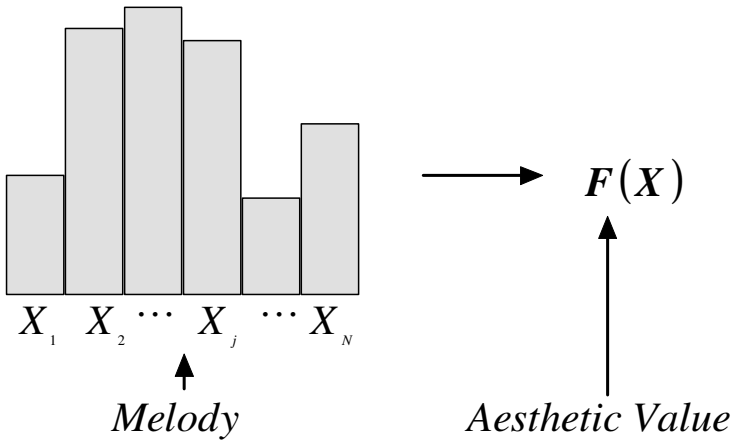
A harmóniakereső algoritmus a zenei improvizáció koncepcióját követi a minél jobb előadásra való törekvés, a tökéletes harmónia állapot keresésében. A zenei harmónia, a tökéletes hangzás elérése analóg az optimalizáló folyamatok esetében a globális optimum megtalálásával. A hangszerek hangmagassága (*pitch*) meghatározza a hangzás esztétikai minőségét, úgy ahogy a döntési változók értékeinek halmaza meghatározza a célfüggvény értékét.

Az optimalizálás során alkalmazott keresési eljárás veszi át a zenében alkalmazott improvizáció szerepét. Az improvizáció során minden zenész ötletszerűen valamilyen hangot játszik a hangszerén. A megszólaló hang a hangszer által előállítható hangok értéktartományából származik. A zenészek által lejátszott hangok együttesen egy harmónia vektort alkotnak. Amennyiben a lejátszott hangok együttesen egy jó harmóniát, dallamot alkotnak, akkor ezt az élményt minden zenész eltárolja a saját memóriájában. Egy jó minőségű dallamból kiindulva, megnő az esélye annak, hogy egy még jobb minőségű dallamot hozunk létre.

Hasonlóan történik ez a mérnöki optimalizálás során is, minden döntési változóhoz hozzárendelünk egy induló értéket adott értéktartományból, így képezve egy induló megoldásvektort. Ha a döntési változók értéke egy jó megoldást ad, akkor a megoldásvektor tárolásra kerül, és nő az esélye annak, hogy a következő lépésben egy jobb megoldást kapjunk.

A 2. ábrán az optimális függvényérték keresés és a zenei harmóniakeresés analógiája látható. A döntési változóknak a zenészek, illetve az általuk játszott dallamok, az optimális megoldásnak, a célfüggvény maximumának a tökéletes harmonikus hangzás felel meg.

$$\max\left\{\mathbf{F}(\mathbf{X}) \mid \mathbf{X} = \left\{ X_j \mid \underline{X}_j \le X_j \le \overline{X}_j,\ j \in \{1,2,\ldots,N\} \right\}\right\}$$



2. ábra Az optimumkeresés zenei analógiája (Forrás: Lee&Geem (2005)

A harmóniakereső algoritmus legfontosabb komponense a *repertoár* (*harmony memory* (*HM*)). Ebben a repertoárban mindegyik zenésztől tárolunk bizonyos mennyiségű hangot. Amikor egy zenész véletlenszerűen

kiválaszt egy hangot, és megszólal a zenekar, akkor egy új dallamot keletkezik. Ha ez a dallam jobb, mint egy korábbi, akkor bekerül a repertoárba, és a legrosszabb pedig eltávolításra kerül. Ez a folyamat addig ismétlődik, amíg meg nem találjuk a tökéletes hangzást, a legjobb harmóniát.

Fontos eleme az algoritmusnak az improvizáció folyamata, amely három szabály alapján zajlik. Ha egy valóságos zenekarra, például egy jazz zenekarra gondolunk, ahol az előadás közbeni improvizáció meglehetősen gyakori, a zenészek az improvizáció során a következő gyakorlatot követik:

- emlékezetből lejátszanak egy ismert dallamot, zenedarabot
- más hangfekvésben játszanak egy ismert dallamot, esetleg más ritmusban
- teljesen ötletszerűen, érzéseikre hagyatkozva játszanak

Hasonló elven működik az improvizáció a harmóniakereső algoritmusban, bár a három alapszabály egy picit más:

- a zenész a repertoárból lejátszik egy ismert hangot
- módosít egy ismert hangot (megváltoztatja a hangmagasságot)
- teljesen véletlen hangot játszik

Az optimalizálás során tehát a következő komponensekre alapozva kell végrehajtani az improvizációt: a *repertoár felhasználása*, a *hangmagasság szabályozása* és a *véletlenszerűség*.

A repertoár használatával van lehetőség a korábbi dallamokból való választás révén egy új, jobb dallam előállítására. A repertoár szerepe

hasonló a legjobban illeszkedő egyedek kiválasztásához a genetikus algoritmusokban. Az improvizáció során keletkező legjobb dallamok bekerülnek az új repertoárba. Azért, hogy még hatékonyabban használjuk a repertoárt, hozzárendelünk egy paramétert, $HMCR \in [0,1]$ amelyet harmónia elfogadási (*Harmony Memory Consideration Rate* $(HMCR)$) tényezőnek nevezünk. Ez a tényező azt adja meg, hogy a következő elemet a repertoárból válasszuk, vagy nem. Ha ez a tényező túl alacsony értékű, akkor csak kevés dallam kerül kiválasztásra a repertoárból és az algoritmus lassan konvergál. Ha ez az érték túl magas, közel 1, akkor majdnem az összes hang felhasználásra kerül a repertoárból, s ekkor csökken más dallamok felfedezésének lehetősége, amely potenciálisan rossz megoldásokhoz vezethet. Ezért a tipikus $HMCR$ érték a következő: $HMCR = 0{,}7 \sim 0{,}95$. A $HMCR$ értékkel adjuk meg, hogy mekkora valószínűséggel választunk hangot a repertoárból. Annak a valószínűsége, hogy egy teljesen véletlen, a repertoárban még nem szereplő hangot választunk, $1 - HMCR$.

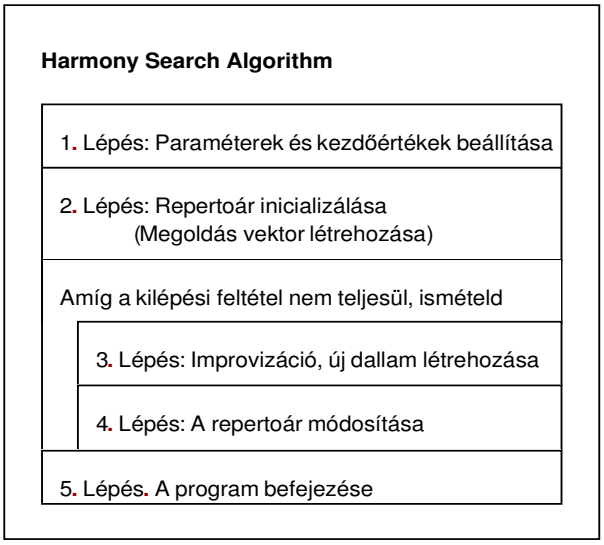
A második komponens a hangmagasság szabályozása, változtatása. Elméletben, a hangmagasság lineárisan vagy nemlineárisan változtatható, de a gyakorlatban, a lineáris hangolást használják. Ha $\mathbf{x}_{old}$ az aktuális megoldás (vagy hangmagasság), akkor az új megoldás (hangmagasság), az $\mathbf{x}_{new} = \mathbf{x}_{old} + bw * \textbf{\textit{Random()}}$ összefüggés szerint jön létre, ahol ***Random()*** egy $[-1,1]$ közötti véletlen szám, bw a sávszélesség.

A hangmagasság változtatása hasonlít a genetikus algoritmusok mutációs operátorára. A hangmagasság megváltoztatásának a mértékét is szabályozza az algoritmus, ezt a mértéket hangmagasság szabályozó mérték, PAR (*pitch adjusting rate*) nevezzük. Ez az érték adja meg, hogy változtatunk-e a hangmagasságon, vagy nem, s milyen mértékben. Ha a

PAR értéke túl alacsony, akkor ritka bármilyen változtatás. Az elfogadott PAR érték a $0{,}1 - 0{,}5$ intervallumba eső érték. A PAR értéke határozza meg, hogy a memóriából választott hangot mekkora valószínűséggel változtatjuk meg a korábban megadott képlet szerint. Annak a valószínűsége, hogy a kiválasztott hangon nem változtatunk: $1 - PAR$.

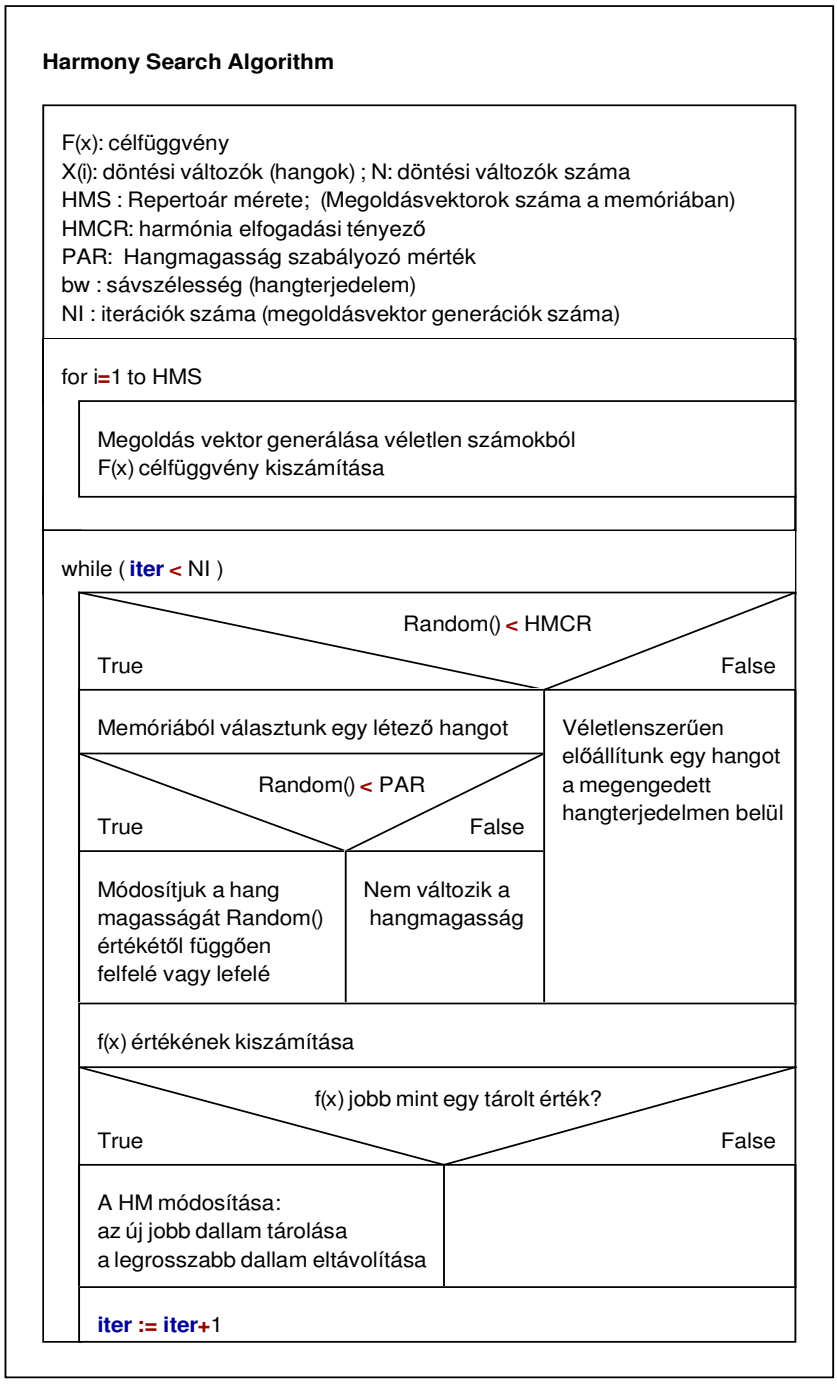
A harmadik komponens a véletlenszerűség. A véletlen elemek mindig a repertoár gazdagítását segítik, s növelik a megoldások változatosságát, a lehetséges megoldások számát. A véletlenszerűség használata tovább segíti az algoritmust, hogy viszonylag gyorsan egy nagyon jó megoldást találjon.

A harmóniakereső algoritmus 5 fő lépésből áll, amelyet a következő struktogramon (*3. ábra*) mutatunk be. (A disszertációban bemutatott struktogramok vagy más néven Nassi-Schneiderman Diagramok egy Open Source szoftverrel, a Structorizer algoritmustervező eszközzel készültek. Forrás: http://structorizer.fisch.lu)



3. ábra. A harmóniakereső algoritmus fő lépései

Az egyes lépések részletesebb kifejtése a *4. ábrán* látható.



4. ábra. A Harmóniakereső algoritmus

1. *Lépés*. A paraméterek értékének inicializálása, megadjuk a repertoár méretét (*HMS*), a *HMCR* és *PAR* értékét, döntési változók számát, iterációk maximális számát (*NI*) vagy egyéb kilépési feltételt.

2. *Lépés*: Véletlen megoldásokkal feltöltjük a memóriát, kiszámoljuk a célfüggvény értékeket, majd sorba rendezzük az értékeket a célfüggvény alapján.

3. *Lépés*: Új harmóniákat, dallamokat hozunk létre improvizációval, a három improvizációs szabálynak megfelelően, amit korábban már részleteztünk. Az egyik lehetőség, hogy a memóriából választunk dallamot *HMCR* valószínűséggel, majd ezt *PAR* valószínűséggel módosítjuk, a másik lehetőség, hogy teljesen véletlen dallamot választunk $1 - HMCR$ valószínűséggel.

4. *Lépés*: Frissítjük a repertoárt. Ha a 3. lépésben kapott dallam jobb, mint a repertoárban tárolt eddigi legrosszabb, akkor a legrosszabb dallamot eltávolítjuk a repertoárból, és az új, jobb dallamot eltároljuk.

5. *Lépés*: Amikor a kilépési feltétel teljesül, azaz elértük a maximális ismétlés számot, az algoritmus befejeződik. Amennyiben még nem teljesül a kilépési feltétel a 3. és 4. lépést megismételjük.

A Harmóniakereső metaheurisztikát eredményesen alkalmazzák különböző optimalizálási feladatok megoldására. A sikerének titka, hogy más heurisztikákban megjelenő elemek alkalmas ötvözete, amelyet könnyen

érthető keretbe foglaltak. Hasonlóan a tabulistás kereséshez, a memóriában tárolja a megoldásvektorokat, képes a *HMCR* érték alapján az elfogadás mértékét változtatni, ami a szimulált hűtésre emlékeztet. Szimultán képes kezelni több megoldásvektort, ami a genetikus algoritmusokkal való hasonlóságot bizonyítja. Ez a tulajdonsága tág teret biztosít a kereséshez, megkönnyíti a globális optimumhoz való konvergálást. Azonban lényeges eltérés a genetikus algoritmusokhoz képest, hogy a harmóniakeresés az összes létező vektorból állít elő új vektort, míg a genetikus algoritmusok két vektorból, a szülőkből származtatnak új egyedet. A harmóniakereső potenciálisan hatékonyabb, mint a genetikus algoritmusok, mivel nem használ bináris kódolást és dekódolást.

Az új vektorok elemeit egymástól függetlenül kezeli, hasonlóan a PMBGA-hoz (*Probabilistic Model Building Genetic Algorithm*). Ellentétben a PMBGA-val, a harmóniakereső algoritmus számára nincs szükség valószínűségi modellre az ígéretes egyedek populációból való kiválasztására, mivel vagy a saját memóriájából választ vagy véletlen értéket választ egy megadott tartományból.

A harmóniakereső algoritmus gradiens keresés helyett sztochasztikus véletlen keresést használ. Így kevesebb matematikai igénye van és komplex feladatok megoldására alkalmas.

A harmóniakereső (*Harmony Search*) algoritmus jó tulajdonságait Mahdavi és ts. (2007) nevéhez fűződő továbbfejlesztés lényegesen megnövelte. Az általuk *Improved Harmony Search* (*IHS*) algoritmusnak nevezett metaheurisztika két új elemet tartalmaz. Az újítás abban rejlik, hogy az algoritmus futása során a *PAR* értéke és a *bw* (sávszélesség) is a generációk számának függvényében változik. A *PAR* értéke a futás során nő, míg a *bw* értéke pont fordítva, csökken. Ez a módosítás jelentős

javulást eredményezett az algoritmus hatékonyságában, melyet a szerzők számítási eredmények közlésével támasztottak alá.

A disszertációban bemutatásra kerülő, általunk kifejlesztett metaheurisztika a harmóniakereső egyik továbbfejlesztése a több-megvalósítási módú erőforrás korlátos projektek optimális ütemezésére.

3.5 Erőforrás kiegyenlítési problémák

A kutatási célkitűzések egyike az volt, hogy az elsődleges szempont szerint történő optimalizáláson túl, tegyük alkalmassá az algoritmust másodlagos szempont szerinti optimalizálásra. A másodlagos szempont a megújuló erőforrás felhasználási hisztogram simítása, kiegyenlítése. Az erőforrás felhasználási hisztogram kiegyenlítése a gyakorlat szempontjából fontos kérdés. Ha az erőforrás felhasználási hisztogram túlságosan dimbes-dombos, az azt jelenti, hogy az erőforrások gyakran leállnak, várakoznak, majd ismét felhasználásra kerülnek. Ennek az egyenetlen erőforrás felhasználásnak pedig komoly pénzügyi vonzatai vannak, így teljesen természetes, hogy a projektvezetők az egyenletes, folyamatos munkát részesítik előnyben.

A másodlagos cél indokolja, hogy áttekintsük az erőforrás kiegyenlítési problémák lényegét.

Kiegyenlítési problémáról beszélünk akkor, amikor a kapott projektütemterv megvalósítását erőforráskorlátok nem akadályozzák, azonban megoldandó problémát jelentenek az erőforrásokban jelentkező ingadozások. A projekt befejezési időpontjának rögzítése mellett, a nem-

kritikus tevékenységek mozgatásával olyan ütemtervet keresünk, amelyhez tartozó megújuló erőforrás felhasználási hisztogramok alakja megközelíti a „kívánatos” alakot, amely a legideálisabb esetben téglalap.

A kiegyenlítési problémák részben vagy teljes egészében nulla-egy típusú optimalizálási feladat megoldását jelentik. A modell célfüggvénye irreguláris, a célfüggvény a tevékenységek kezdési időpontjainak függvényében nem monoton növekvő függvény.

Az erőforrás kiegyenlítési problémák alapmodelljei két csoportba sorolhatók, az erőforrás felhasználási hisztogram alakjával kapcsolatos elvárásoknak fontosságának megfelelően.

Abban az esetben, ha a kiugró erőforrás felhasználást tekintjük hangsúlyos elemnek, és más elvárásaink a hisztogram alakjával kapcsolatosan nincsenek, akkor csupán az erőforrás felhasználási hisztogram lelapításáról beszélünk. Az erőforrás felhasználási hisztogram lelapítása, nem jelenti azt, hogy az erőforrások felhasználásában nem lesznek ingadozások. Ebben az esetben a kiegyenlítési probléma célfüggvénye implicit módon függ a tevékenységek kezdési időpontjától, s a feladat megoldása olyan eljárások alkalmazását teszi szükségessé, amely implicit célfüggvénnyel is képes optimalizálni.

Amennyiben az optimalizálás célja az, hogy az erőforrás felhasználás ingadozásait simítsa ki, akkor a célfüggvény az eltérések mértékének valamilyen módon történő megfogalmazását fejezi ki. A célfüggvény felírható az átlagos erőforrás felhasználástól való négyzetes eltérések minimalizálásaként, vagy megfogalmazhatjuk úgy is a modell célját, hogy minimalizáljuk az egymásután következő időintervallumok erőforrás

felhasználásának négyzetes eltérését. Mindkét esetben a célfüggvény kvadratikus függvény, a modell feltételrendszere pedig lineáris.

Számos egzakt és heurisztikus algoritmust dolgoztak ki a kiegyenlítési problémák megoldására. Az egzakt algoritmusok elsősorban a leszámlálási eljárásokon, egészértékű programozás vagy dinamikus programozási technikán alapulnak. Leszámlálási eljárást publikált Ahuja (1976), Csébfalvi és Konstantinidis (1998), Easa (1989) egészértékű LP eljárást ajánl, amely közvetlenül megoldható, Bandelloni és ts. (1994) dinamikus programozást, Neumann és Zimmermann (2000) szétválasztás és korlátozás (B&B) módszerét publikálta a probléma megoldására.

Heurisztikán alapuló megoldást tettek közzé többek között Burgess és Killebrew (1962), Harris (1990), mely megoldások többsége az egyszerű léptetési technikákat alkalmazó heurisztikát vagy prioritási szabályokon alapuló heurisztikus módszert alkalmazott.

Neumann és Zimmermann (1999b) polinomiális prioritási szabályokon alapuló heurisztikákat közölt a kiegyenlítési problémára, számos különböző célfüggvényre alkalmazva. Megoldásukban minimális és maximális késleltetési időt (*minimum maximum time lag*) definiáltak a tevékenységek között és explicit erőforráskorlátokat adtak meg. A célfüggvényeket három csoportba sorolták.

A célfüggvények első csoportja az időperiódusonkénti erőforrás felhasználás minimalizálását fejezi ki. A célfüggvény súlyfaktorokat foglal magában, amelyek az egyes időperiódusokban a maximális erőforrás felhasználás költségét tükrözik. Ezt a célfüggvény típust alkalmazzák legtöbbször az erőforrás befektetési problémákban.

Neumann és Zimmermann második vizsgált célfüggvény típusa az eltérések mérésén alapul. A célfüggvény egy adott erőforrás átlagos erőforrás felhasználástól való eltérését fejezi ki. Pozitív és negatív eltéréseket egyaránt figyelembe lehet venni, de lehet cél a négyzetes eltérések mértékének minimalizálása is.

A harmadik célfüggvény típus, amit munkájuk során vizsgáltak, a túlmunka időperiódusról időperiódusra történő változásának mértékét fejezi ki. A célfüggvényben megadható a pozitív vagy negatív, vagy a négyzetes eltérés mértékeket minimalizáló összefüggés.

Anagnostopoulos és Koulinas (2010) szimulált hűtésen alapuló hiper-heurisztikát fejlesztett ki az erőforrás felhasználási hisztogram kisimítására. Eljárásuk abból a feltételezésből indul ki, hogy a tevékenységek időtartama állandó és az elsőbbségi kapcsolatok rögzítettek. Minden tevékenységhez egyetlen erőforrás van hozzárendelve, s a felhasznált erőforrásegységek száma állandó egy tevékenység végrehajtása során. A rendelkezésre álló erőforrásegységek száma periódusonként is állandó. Algoritmusukat az MS-Project 2003 szoftvercsomagba illesztették be, így annak keretrendszerében futtatható.

A célfüggvény az átlagos erőforrás felhasználási szükséglettől való eltérés négyzetösszegét minimalizálja, minden időperiódusra vonatkozóan.

Geng és ts. (2011) egy hangyaboly optimalizáló algoritmust, az irányított hangyakolónia optimalizáló algoritmust (*directional ant colony optimization* DACO) mutatja be. A DACO megközelítés egy hangyaraj alkalmazásával egy irányban, csomópontról csomópontra haladva keres ígéretesnek mondható utat a keresési fában, s a kritikus tevékenység csomópontokat egy tabulista alkalmazásával kezeli. A hangyák száma a

projektháló topológiájának komplexitásával és az iterációk számával van szoros összefüggésben.

A hangyák a tevékenységek korai és a legkésőbbi kezdési időpontjai között keresik a legjobb kezdési időpontot a nem kritikus tevékenységek halmazán. Az algoritmus a napi erőforrás felhasználás maximális értéke és a napról-napra történő erőforrás felhasználás eltéréseit használja fel jóság (fitnesz) függvényként.

4. Egy új harmóniakereső eljárás

A harmóniakereső metaheurisztikát elsősorban mérnöki problémák megoldására fejlesztették ki. A harmóniakereső eljárás nagyon hatékonyan alkalmazható a projektütemezési problémák megoldására is. Erre találunk példát Csébfalvi és ts. (2008a), Csébfalvi és ts. (2008b), Szendrői (2009, 2010a, 2010b.2010c). Ezek a megoldások a disszertációban bemutatott algoritmus fejlődésének különböző állomásai. Ezekben a tanulmányokban a szerzők Csébfalvi és ts.(2008a) erőforrás-korlátos projektütemezési problémákra adott „Sounds of Silence" metaheurisztikus algoritmusát fejlesztik tovább a fenti problémák megoldására. Szendrői a több megvalósítási módú erőforrás-korlátos projektütemezési probléma megoldásával, míg Láng (2010) az erőforrás-korlátos projektütemezés nettó jelenérték maximalizálására ad megoldást.

4.1 A matematikai modell

A több megvalósítási módú erőforrás korlátos projektek (MRCPSP) ütemezésének célja olyan megvalósítási módokat rendelni az egyes tevékenységekhez, hogy az erőforráskorlátok betartása mellett, megtaláljuk azt az optimális ütemezést, amely esetén a projekt végrehajtásának teljes időszükséglete minimális. A modell megoldásánál a legfontosabb cél az volt, hogy hatékony heurisztikát találjunk a feladat megoldására.

A vizsgált modellünk a következő tulajdonságokkal jellemezhető:

- A projekt N darab valós tevékenységből áll, amelyeket 1-től *N*-ig sorszámozunk.

- Minden *i* tevékenység, $i \in \{1,2,...,N\}$ az M féle megvalósítási mód valamelyikében valósul meg.

- Jelölje a 0-dik és az $N+1$-dik látszólagos tevékenység a projekt egyedi kezdetét és végét.

- A tevékenységek végrehajtását megelőző-rákövetkező (elsőbbségi) feltételek és erőforrás korlátok befolyásolják. A megelőző-rákövetkező kapcsolat meghatározza, hogy egy adott tevékenység addig nem kezdődhet meg, amíg az azt megelőző összes tevékenység be nem fejeződik. Jelölje ***PS*** halmaz, $\boldsymbol{PS} = \{ i \to j \mid i \neq j, i \in \{0,1,\ldots,N\}, j \in \{1,2,\ldots,N+1\}\}$, a tevékenységek közötti elsőbbségi feltételek halmazát, ahol a nyíl szimbólum azt jelzi, hogy a j tevékenység csak az i tevékenység befejezése után kezdődhet el.

- A megújuló erőforrásfajták számát jelölje R, a nem-megújuló erőforrásfajták számát jelölje C.

- Az i-edik tevékenység végrehajtása, $i \in \{1,2,...,N\}$ az m-edik megvalósítási módban, $m \in \{1,2,...,M\}$, D_{im} időegységet igényel.

- Jelölje R_r az időegység alatt rendelkezésre álló r-edik megújuló erőforrás korlátját, ahol $r \in \{1,2,...,R\}$.

- Legyen C_c a projekt teljes erőforráskorlátja a c-edik, $c \in \{1,2,...,C\}$ nem-megújuló erőforrásra vonatkozóan.

- Jelölje R_{imr} az i-edik tevékenység időegységre eső megújuló erőforrásigényét az m-edik megvalósítási módban, az r-edik erőforrásból.

- Legyen C_{imc} az i-edik tevékenységnek a nem-megújuló erőforrásigénye a c-edik erőforrásból az m-edik megvalósítási módban.

- Jelölje $\overline{T}$ az elsőbbségi és erőforráskorlátokat kielégítő projekt időtartamának felső korlátját:

$$\overline{T} = \sum_{i=1}^{N} \max\left(D_{im} \mid m \in \{1,2,\ldots,M\}\right). \qquad (1)$$

- Jelölje X_i az i-edik, $i \in \{1,2,...,N\}$ tevékenység kezdetét és legyen $\left[\underline{X}_{im}, \overline{X}_{im}\right]$ az az időintervallum, amelyben az i-edik tevékenység az m-edik megvalósítási módban elkezdődhet $m \in \{1,2,...,M\}$. Az $\underline{X}_{im}$ ($\overline{X}_{im}$) jelöli az i-dik tevékenység legkorábbi (legkésőbbi) lehetséges kezdőidőpontját az m-edik módban, az erőforráskorlátok nélküli esetben rögzített legkésőbbi projektbefejezésnek megfelelően.

- Jelöljük a modell döntési változóit az X_{ims} bináris változókkal, amelyek értéke 1, ha az i-edik tevékenység az m-edik $m \in \{1,2,...,M\}$ megvalósítási módban kerül ütemezésre s kezdési idővel, egyébként az értéke legyen 0.

Az MRCPSP célja, hogy megfelelő megvalósítási módot találjon a tevékenységek számára, figyelembe véve az elsőbbségi- és erőforráskorlátokat úgy, hogy a projekt végrehajtásának időszükséglete minimális legyen.

A megoldandó matematikai modell a következő alakban adható meg:

$$\boldsymbol{min}\left[X_{N+1}\right] = X_{N+1}^{*} \tag{2}$$

$$X_{N+1} \le \overline{T} + 1 \tag{3}$$

$$\sum_{m=1}^{M} \sum_{s=\underline{X}_{im}}^{\overline{X}_{im}} X_{ims} = 1,\ i = 1,2,\ldots,N \tag{4}$$

$$\sum_{m=1}^{M} \sum_{s=\underline{X}_{im}}^{\overline{X}_{im}} s * X_{ims} \le X_{N+1},\ i \to N+1 \in \boldsymbol{PS} \tag{5}$$

$$\sum_{m=1}^{M} \sum_{s=\underline{X}_{im}}^{\overline{X}_{im}} \left(s + D_{im}\right) * X_{ims} \le \sum_{m=1}^{M} \sum_{s=\underline{X}_{jm}}^{\overline{X}_{jm}} s * X_{jms},\ i \to j \in \boldsymbol{PS} \tag{6}$$

$$\sum_{i=1}^{N} \sum_{m=1}^{M} \sum_{s=\underline{X}_{im}}^{\overline{X}_{im}} W_{imst} * R_{imr} * X_{ims} \le R_r, t = 1,2,\ldots,T,\ r = 1,2,\ldots,R \tag{7}$$

$$W_{imst} = \begin{cases} 1 & s \le t \ \wedge \ t < s + D_{im} \\ \quad ha & \\ 0 & t < s \ \vee \ t \ge s + D_{im} \end{cases} \tag{8}$$

$$\sum_{i=1}^{N} \sum_{m=1}^{M} \sum_{s=\underline{X}_{im}}^{\overline{X}_{im}} C_{imc} * X_{ims} \le C_c,\ c = 1,2,\ldots,C \tag{9}$$

$$X_i = \sum_{m=1}^{M} \sum_{s=\underline{X}_{im}}^{\overline{X}_{im}} s * X_{ims},\ i = 1,2,\ldots,N \tag{10}$$

$$M_i = \sum_{m=1}^{M} \sum_{s=\underline{X}_{im}}^{\overline{X}_{im}} m * X_{ims} \ , \ i = 1,2,\ldots,N \tag{11}$$

$$\boldsymbol{X} = \{X_1, X_2, \ldots, X_N\} \tag{12}$$

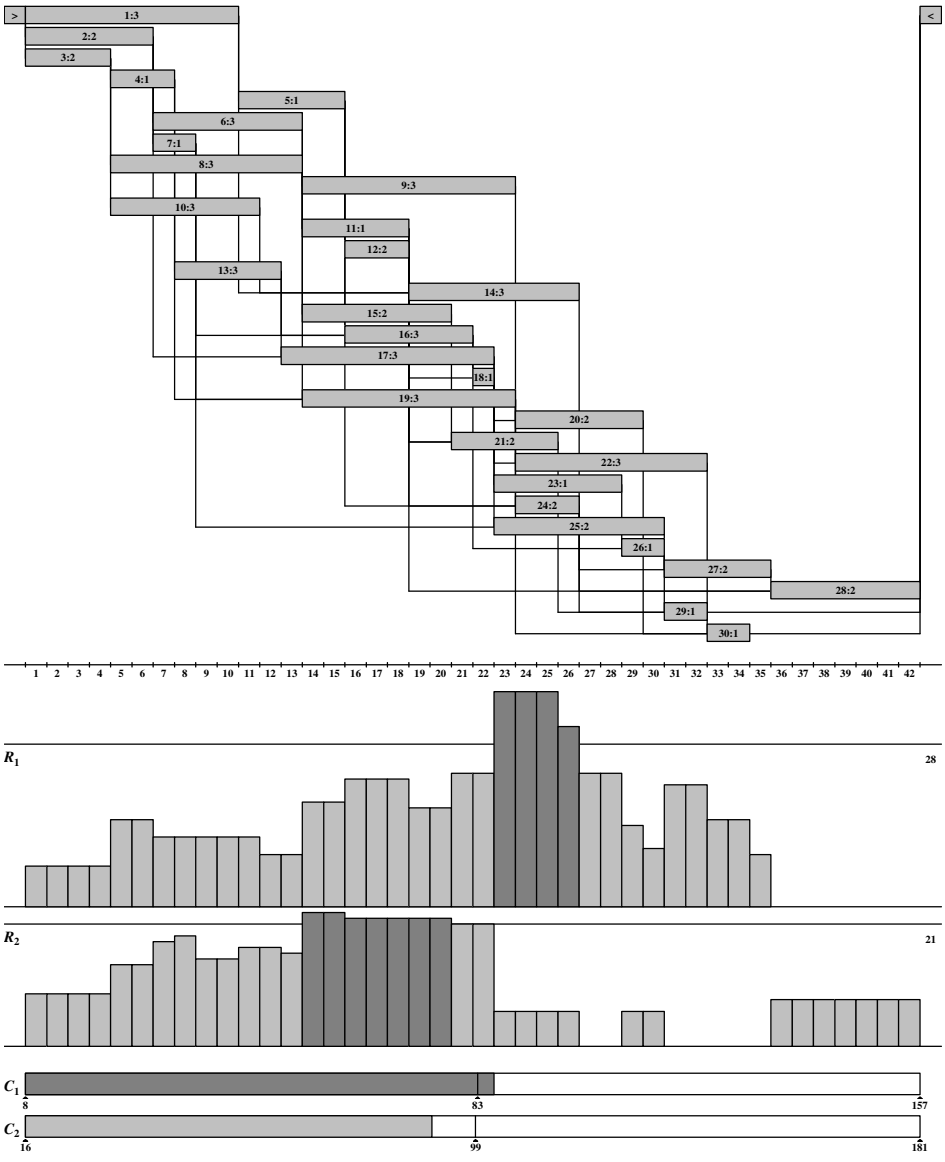
$$\boldsymbol{M} = \{M_1, M_2, \ldots, M_N\} \tag{13}$$

$$X_{ims} \in \{0,1\}, \ i = 1,2,\ldots,N \ , \ m = 1,2,\ldots,M \ , \ s = \underline{X}_{im},\ldots,\overline{X}_{im} \tag{14}$$

A (4) feltétel biztosítja, hogy minden i-edik tevékenység végrehajtása pontosan egyszer, egy adott megvalósítási módban és a megvalósítási módjának megfelelő időintervallumban kezdődjék el. A (5-6) feltételek az elsőbbségi (megelőzési-rákövetkezési) relációkat írják le. Az időperiódusokban rendelkezésre álló megújuló erőforrás típusokra vonatkozó korlátokat a (7-8) feltételek adják meg. A (9) feltétel a teljes nem-megújuló erőforrás kapacitás korlátot írja le. A döntési változókra vonatkozó feltételeket a (10-14) kifejezések tartalmazzák. Végül, de nem utolsó sorban, a (2) kifejezés a modell célfüggvénye, amely a projekt végrehajtásának időszükségletét minimalizálja.

Az MRCPSP modell szemléltetésére egy 30 valóságos tevékenységből álló projektet mutatunk be. A projektben 2 megújuló és 2 nem-megújuló erőforrás használható fel a tevékenységek végrehajtásakor. A valós tevékenységek mindegyike a három megvalósítási mód valamelyikében hajtható végre. A bemutatásra kerülő projektpélda a J30MM-10-1, amely a „legnehezebb” kategóriába tartozik a PSPLIB tesztkönyvtár J30MM több megvalósítási módú projekthalmazából, amelyet Kolisch és Sprecher (1996) hozott létre. A *5. ábrán* látható ütemezés az erőforráskorlátok figyelembe vétele nélkül kapott legkorábbi J30MM-10-1 projektütemezés, véletlenszerűen generált megvalósítási módokkal. A tevékenységeket téglalapokkal, a tevékenységek közötti kapcsolatokat vonalakkal

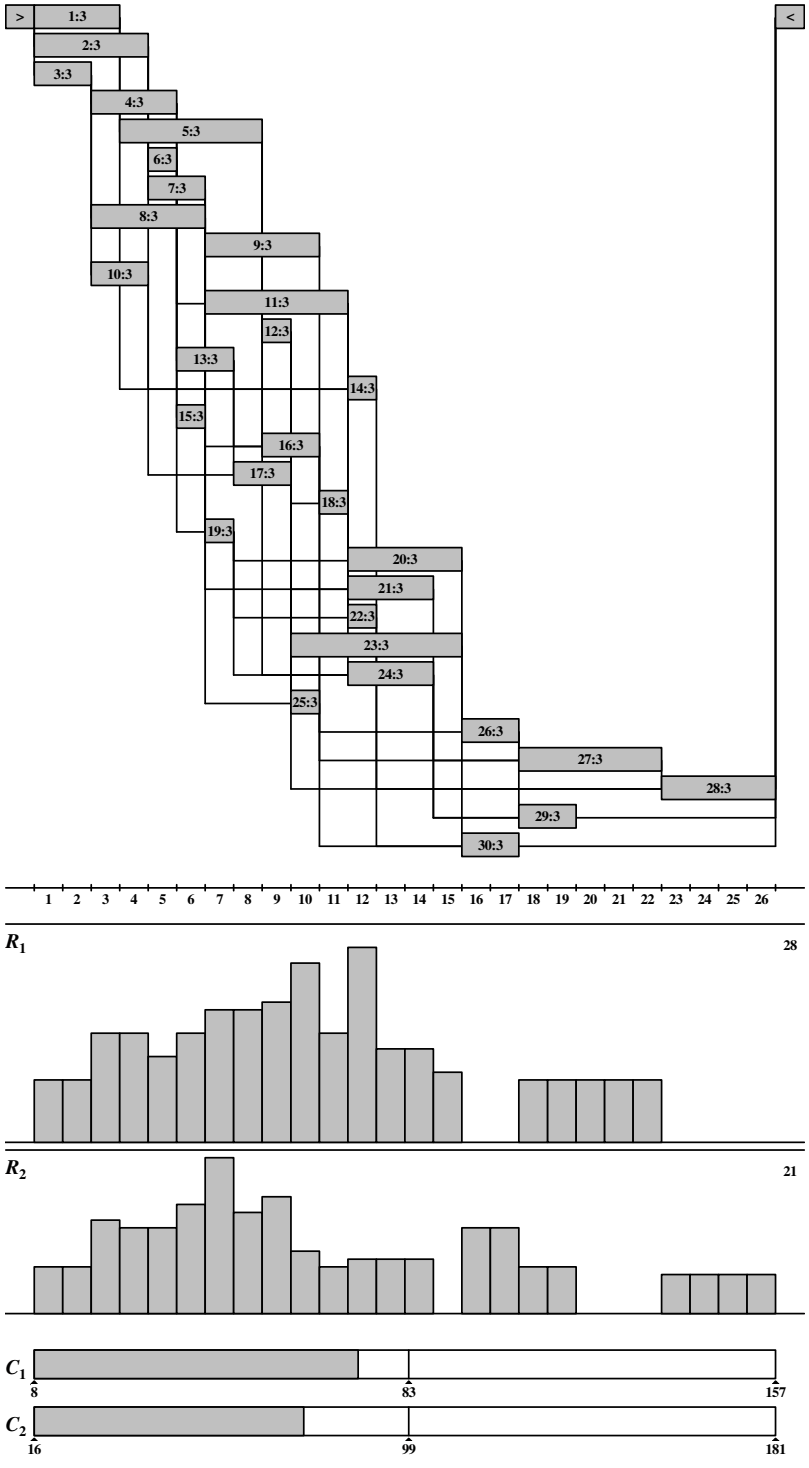
szemléltetjük. A valódi tevékenységeket az $i:m$ formában jelöljük, ahol i a tevékenység sorszámát, m a tevékenység megvalósítási módját jelöli. A projekt kezdetét és végét jelző általevékenységeket a > (<) szimbólumok jelzik. Az általevékenységeknek nincs erőforrásigényük.



5. ábra Legkorábbi J30MM-10-1 projektütemezés a véletlenszerűen generált működési módokkal

A megújuló (nem-megújuló) erőforrás felhasználási hisztogramokban sötétszürke színnel jelöltük azokat az időperiódusokat, amelyekben az erőforrásigény az erőforráskorlátokat meghaladja.

A *6. ábra* az első ábrán bemutatott modell egy optimális (minimális időtartamú erőforrás korlátokat kielégítő) ütemezését, megoldását mutatja. Meg kell jegyezni, hogy egy projektnek természetesen több ekvivalens megoldása lehet, amit másodlagos szempont szerint sorba lehet rendezni.



6. ábra A J30MM-10-1 projekt optimális ütemezése

4.2 Az algoritmus

Ebben a szakaszban a hibrid harmóniakereső algoritmust ismertetjük. Először a projektütemezés és a zenei analógia kapcsolatát részletezzük, majd az algoritmus legfontosabb komponenseinek részletes ismertetésére kerül sor. Végül az algoritmus fő lépéseit taglaló stuktogram kerül bemutatásra.

4.2.1 A harmóniakereső analógia

A harmóniakereső (HS) algoritmust Lee és Geem (2005) dolgozta ki a zenei improvizáció analógiájára, ahol a zenészek egy jobb harmónia elérésére törekednek. A harmóniakereső eljárásban az optimalizálási feladat a következőképpen írható le:

$$\boldsymbol{max}\left\{ f(\boldsymbol{X}) \mid \boldsymbol{X} = \left\{ X_i \mid \underline{X}_i \le X_i \le \overline{X}_i, i \in \{1,2,...,N\} \right\} \right\} \qquad (15)$$

A zene nyelvén $\mathbf{X}$ egy dallam, amelynek esztétikai értékét az $f(\mathbf{X})$ függvény írja le. Minél magasabb $f(\mathbf{X})$ értéke, annál jobb a hangzás minősége. A zenekarban a zenészek száma N, és az i-dik zenész, $i = \{1,2,\ldots,N\}$ az X_i dallam megszólalásáért felel. Az improvizációs folyamatot két paraméter vezérli:

(1) A repertoár figyelembe vételi rátának megfelelően (RCR, *repertoire consideration rate*) minden zenész választ egy dallamot a saját

repertoárjából az RCR rátának megfelelő valószínűséggel, vagy egy teljesen véletlen érték alapján $(1-RCR)$ valószínűséggel;

(2) A hangmagassági rátának megfelelően (SAR, *sound adjusting rate*) egy, a zenész saját repertoárjából választott hang SAR valószínűséggel módosul.

Az algoritmus egy teljesen véletlenszerű „repertoár feltöltő" fázissal kezdődik, ezt követően a zenekar improvizálni kezd. Az improvizáció során, ha egy új dallam jobb, mint a repertoár legrosszabb darabja, akkor a repertoár legrosszabb darabját helyettesítjük a jobb dallammal. A HS algoritmus két legfontosabb paramétere a repertoár mérete és az improvizációk száma. Az eredeti HS algoritmus egy „explicit" algoritmus, mivel közvetlenül a hangokon fejti ki hatását.

A Csébfalvi féle SoS algoritmus és ebben a dolgozatban szereplő algoritmus is implicit módon kezeli a hangokat, (az erőforrás felhasználási hisztogram maga a zene, s az erőforrás felhasználási hisztogramokat nem lehet közvetlenül manipulálni) így be kellett vezetni a „karmester" fogalmát a probléma megoldásához. A HS-beli improvizáció véletlenszerűen választott hangok véletlenszerű módosítását jelenti. Az SoS-ben és az itt bemutatott algoritmusban is az improvizáció egy a karmester által választott dallam módosítása.

Először megmutatjuk, hogyan lehet az eredeti feladatot a zene nyelvén megfogalmazni. A zene világában az erőforrásprofilok egy „többszólamú dallamot" alkotnak. Ha feltesszük, hogy minden szólamban csak a „magas hangok" hallatszanak, akkor a feladat a következő lesz: Keressük a legrövidebb „Sounds of Silence (Csend hangjai)" melódiát az improvizáció során, vagyis a legrövidebb csendet! Természetesen a zenei analógiában

szereplő „magas hang” a projektütemezésben az erőforráskorlát átlépését (túlmunka) jelenti.

A zenei analógia nyelvén fogalmazva a több megvalósítási módú projektütemezési probléma a következőképpen írható le:

Az MRCPSP esetben minden i zenész, $i \in \{1,2,...,N\}$ jellemezhető egy diszjunktív többszólamú hanghalmazzal, és a nem-megújuló erőforrást egy adott energiatípusból származó, az előadáshoz szükséges „energiaként” interpretálva, a nem-megújuló erőforrást egy hang megszólaltatásához szükséges „fizikai energiának” tekinthetjük, vagy az előadáshoz szükséges „spirituális” energiaként foghatjuk fel. Természetes feltételezés az is, hogy mindegyik energia fajtából a zenekar „teljes” energiája korlátozott és a teljes energiát az előadás során felhasználják. A nem-megújuló erőforrások kezelése rendkívül nehéz probléma, s igen érdekes, hogy milyen természetes módon megoldható ez a probléma azzal, hogy a zene lejátszásához szükséges energiaként kezeljük, amely energia az előadás végére elfogy, elfáradnak a zenészek.

A következő táblázatban a projektütemezés és a zenei analógia egymásnak megfelelő fogalmait foglaltuk össze.

3. táblázat Projektütemezés zenei analógiája

Zenei analógia	MRCPSP modell
A zenekar N zenészből áll	A projekt N tevékenységből áll
Zenész	Tevékenység
Minden zenész jellemezhető egy diszjunktív többszólamú hanghalmazzal és a hang lejátszásához szükséges energiával	Minden tevékenység több megvalósítási móddal és erőforrásigénnyel rendelkezik
Polifonikus dallam	Erőforrás hisztogram
Magas, hallható hang	Túlmunka, erőforráskorlát túllépése
Szólam	Erőforrástípus (megújuló)
Hangok megszólalási sorrendje	Tevékenységek sorrendje, ütemezés
Zenész belépési időpontja	A tevékenység kezdési időpontja
Előadáshoz szükséges energia	Nem-megújuló erőforrás
A dallam hossza	A projekt teljes időszükséglete (makespan)

Minden lépésben minden zenész egy IS_i, $IS_i \in [1, M]$ valószínűségi értéket ad (módosít) a „legjobb" M_i melódiáról, és egy IP_i, $IP_i \in [-1,+1]$ valószínűségi értéket ad arról, hogy mikor kívánja megszólaltatni a dallamot, azaz a „legjobb" belépési X_i időpontról nyilatkozik. A nagy pozitív (negatív) érték azt jelenti, hogy a zenész amilyen korán (későn) csak lehet, olyan korán (későn) kíván belépni a dallamba. Az elképzelés

lényege a *7-8. ábrákon* látható. A *7. ábrán* szereplő π_{im} súlyérték, amely annak a valószínűsége, hogy a kiválasztott (x) érték az $m \pm 0{,}5$ környezetébe esik, a következőképpen definiálható:

$$\pi_{im} = \int_{m-0.5}^{m+0.5} \mathbf{Gauss}(x, IS_i, \sigma)\, dx\,,\ i = \{1,2,\ldots,N\}, m = \{1,2,\ldots,M\}. \qquad (16)$$

ahol IS_i a várható érték, σ a szórás.

Az eredeti változatban a repertoár feltöltési fázisban IS_i értékét normális eloszlásból generáljuk,

$$IS_i = \mathbf{RandomGauss}(\mu, \sigma),\ 1 \le IS_i \le M\,, \qquad (17)$$

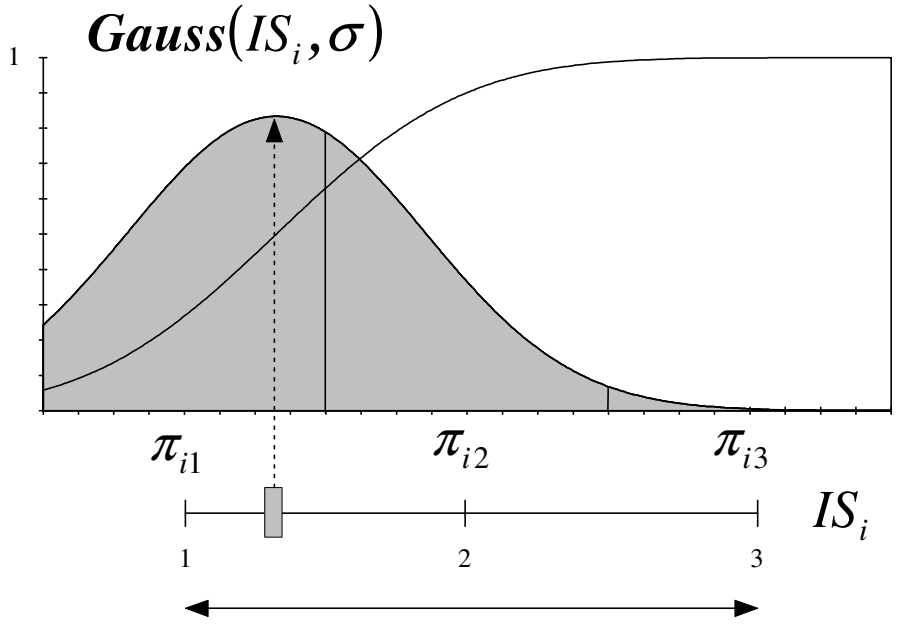
$\mu = 1$ és $\sigma = 1$ paraméterekkel.

A bemutatott módosított változatban a relaxált megoldásból véletlenszerű perturbációval egy elő-optimalizált repertoárt hozunk létre:

$$IS_i = \mathbf{RandomGauss}(\tilde{M}_i, \sigma), 1 \le IS_i \le M\,, \qquad (18)$$

ahol $\tilde{M}_i, 1 \le \tilde{M}_i \le M$, $i \in \{1,2,\ldots,N\}$ a relaxált mód érték a relaxált megoldásból. Az előoptimalizáláskor a repertoárt az adott több megvalósítási módú szituációnak megfelelően állítjuk elő, kikeverjük azokat a valószínűségeket, amelyek diszjunktív dallamokhoz tartozhatnak. Az elő-optimalizálás jelentős újítás az algoritmusunkban, amellyel nagymértékű hatékonyság javulást értünk el. Az előoptimalizálás során a keresési tér leszűkíthető azokra a tevékenységek és megvalósítási mód kombinációkra, amelyek nem sértik a nem-megújuló erőforráskorlátokat. Az algoritmusunk módosított változatában σ a heurisztikánk egy "bűvös

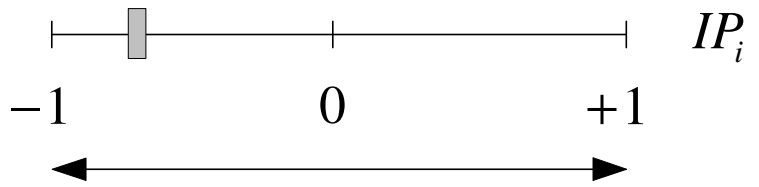
száma" mivel a kezdő repertoár változatosságára nagy hatással van ez az érték. Minden heurisztikának vannak állítható paraméterei, a mi esetünkben állítható paraméter a σ értéke. Az előzetes eredményeinknek megfelelően egy jó beállítás a következő értéktartomány: $0{,}1 \le \sigma \le 0{,}2$. Kezdetben a választás szabadsága nagy, majd az improvizációk során az elképzelések szabadsága lépésről lépésre csökken, azaz a σ szórás értéke folyamatosan csökken.



7. ábra Egy IS_i elképzelés a „legjobb" megvalósítási módról ($M = 3$)

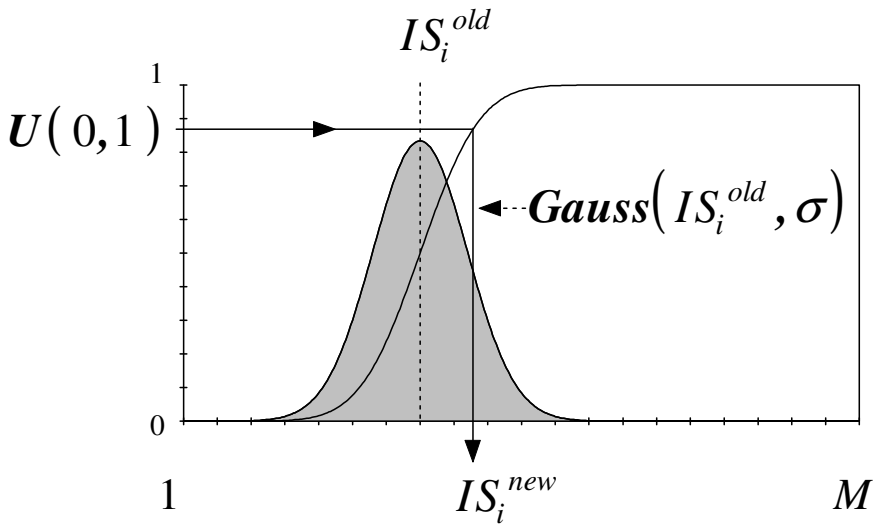
Az előoptimalizálás eredményeképpen a 7. ábrán jól látható, hogy csak olyan megvalósítási módok választhatók valamilyen adott valószínűséggel, amelyek a lehetséges megvalósítási módok halmazába tartoznak.

Hasonlóan a megvalósítási módhoz, minden improvizációs lépés során a zenészek nyilatkoznak a „legjobb” pozícióról, vagyis arról mikor kívánnak belépni a dallamba, mikor kezdődjék egy adott tevékenység. Az elképzelések szabadsága itt is lépésről lépésre csökken, $IP_i = \textbf{\textit{RandomGauss}}(IP_i, \sigma)$, ahol σ szórás értéke folyamatosan csökken.

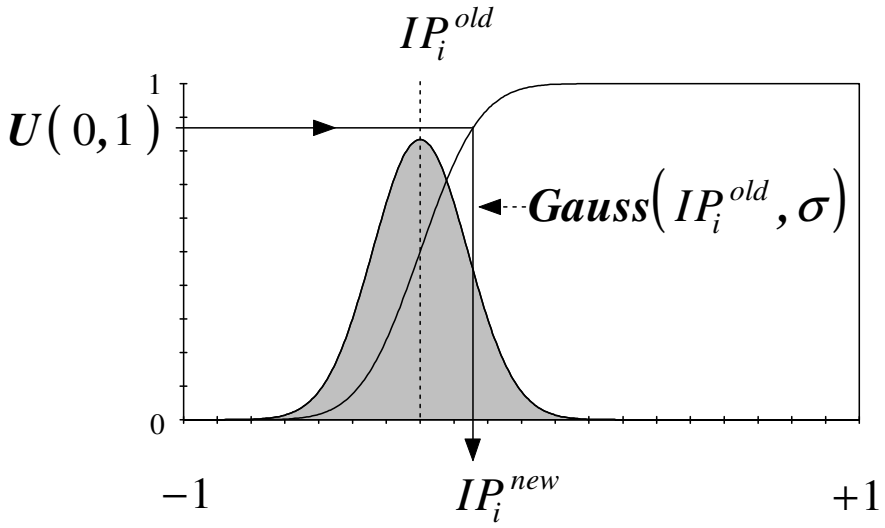


8. ábra Egy IP_i elképzelés a "legjobb" pozícióról

Az improvizációs fázisban a régi $IS_i(IP_i)$ az eloszlás várható értéke lesz, amelyből az új, perturbált $IS_i(IP_i)$ értékek keletkeznek, ahol a σ szórás értéke a korábban említetteknek megfelelően folyamatosan csökken (lásd 9.-*10. ábra*).



9. ábra IS_i perturbációja



10. ábra IP_i perturbációja

4.2.2 Az improvizációt követő lépések

Az eredeti HS algoritmusban a zenészek szabadsága maximális, és az improvizáció definíciója távol van a valóságtól. A HS-ben egy improvizáció a véletlenszerűen választott hangok véletlenszerű módosításainak halmaza. Az alkalmazott megközelítésünkben az improvizáció egy többé-kevésbé harmonikus melódia véletlenszerű perturbációját jelenti, ezért az improvizáció valóban közelebb van a valósághoz.

Az improvizációs fázisok során a karmester felelőssége a dallam kiválasztása és a zenészek elképzeléseinek összehangolása is.

Az algoritmus lényege nagyon egyszerű: az eljárás a repertoár feltöltési fázisával kezdődik. Ezután minden improvizációs lépésben a karmester kiválaszt egy dallamot, minél rövidebb a dallam, annál nagyobb az esély, hogy azt választja a karmester. A karmester tulajdonképpen egy rulettkerék (rulet wheel) eljárásnak megfelelően választ, minél kisebb a projektütemezés időtartama, a makespan, annál nagyobb a tortaszelet, s annál nagyobb a valószínűsége, hogy a karmester azt választja.

Ezt követően a zenészek módosítják saját elképzeléseiket, majd a karmester összegyűjti a módosított elképzeléseket és egy MILP és egy LP problémát old meg, azért hogy kiegyenlítse a többé vagy kevésbé ellentétes elképzeléseket a jobb harmóniáról.

A MILP a következő összefüggésekkel írható le:

$$\boldsymbol{max}\left[\sum_{i=1}^{N}\sum_{m=1}^{M}\pi_{im} * Y_{im}\right] \tag{19}$$

$$\sum_{m=1}^{M} Y_{im} = 1,\ i = \{1,2,\ldots,N\} \tag{20}$$

$$\sum_{i=1}^{N}\sum_{m=1}^{M} C_{imc} * Y_{im} \le Cc,\ c \in \{1,2,\ldots,C\} \tag{21}$$

$$Y_{im} \in \{0,1\} \tag{22}$$

A MILP eredménye egy energia korlátot kielégítő dallamkombináció $M = \{M_1, M_2, \ldots, M_N\}$, amely maximalizálja a zenészek megelégedettségét. A projektütemezési problémára visszautalva a MILP eredménye az erőforráskorlátokat kielégítő megvalósítási módok halmaza lesz. Elméletileg ez a MILP modell egy úgynevezett több-választásos többdimenziós hátizsák probléma (MMKP), számos gyors és hatékony problémamegoldó lehetőséggel. Csébfalvi G. és Csébfalvi A. (2008) egy egyszerű probléma-specifikus heurisztikát fejlesztett ki a probléma megoldására, amely versenyképes az elterjedt MILP megoldó szoftverekkel (például: CPLEX).

Az LP probléma, amely maximalizálja a zenészek megelégedettségét a hang pozícionálásával kapcsolatosan, a következő:

$$\boldsymbol{min}\left[\sum_{i=1}^{N} IP_i * X_i\right] \tag{23}$$

$$X_i + D_i \leq X_j, i \rightarrow j \in \boldsymbol{PS}\ ,\ D_i = D_{iM_i} \tag{24}$$

$$\underline{X}_i \leq X_i \leq \overline{X}_i,\ i \in \{1,2,\ldots,N\} \tag{25}$$

A feladat változóit a hangok (tevékenységek) kezdési időpontjai alkotják. A feltételrendszer a hangok (tevékenységek) közötti kapcsolatokat (követő hang kezdési időpontja nem lehet kisebb, mint a megelőző hang kezdési időpontja+hossza) írja le. Az optimalizálás eredménye egy ütemezés (dallam), amelyet a karmester arra használ, hogy meghatározza a hangok (zenészek) végső kezdési (belépési) sorrendjét. A karmester létrehoz egy „nem hallható", az energia korlátnak megfelelő melódiát a kiválasztott hangok (tevékenységek) adott sorrendbe helyezésével, és ütemezi őket a lehetséges legkorábbi (legkésőbbi) kezdési időpontra.

Ezután a jól ismert *Forward-Backward Improvement* (röviden: *FBI*) eljárással (Tormos és Lova (2001)) és az új „head-tail" helyi minőségjavító (local improvement) heurisztikával, a karmester megpróbálja javítani a létrehozott dallam minőségét. Természetesen a karmester megjegyzi az addigi legrövidebb lehetséges melódiát, azaz ütemezést. A korábbi algoritmusban az eljárásnak ezen a pontján egy egyszerű forward-

backward eljárás szerepelt, ez lett lecserélve az *FBI* eljárással, valamint a head-tail, a projekt elejét és végét kezelő, minőségjavító heurisztikával. A head-tail eljárás alkalmazása jelentősen javította a metaheurisztikánk hatékonyságát, amit később a futási eredményekkel bizonyítunk is.

4.2.3 A projekt elejét-végét kezelő minőségjavító eljárás

A projekt elejét-végét kezelő, „head-tail" helyi minőségjavító algoritmus csak egyetlen hangolható paraméterrel rendelkezik, amely a „head-tail" mérete. Ha ez a paraméter egyenlő eggyel, akkor csak a projekt kezdő és befejező elemeit használja fel az újraallokálási folyamatban. Ha ez az érték egyenlő kettővel, akkor a kezdő tevékenységek és az őket közvetlenül követők valamint a befejező tevékenységek és az ő közvetlen megelőző tevékenységei alkotják a „head-tail" halmazt.

A head-tail algoritmus segítségével újraoszthatjuk a nem-megújuló erőforrásokat a projekt eleje és vége között, ha ez azt eredményezi, hogy a projekt időtartama rövidebb lesz. Természetesen minél nagyobb a „head-tail" mérete, annál nagyobb az esély arra, hogy a projekt időszükséglete csökkenthető az erőforrások újraallokálásával. Meg kell jegyezni, a számítási költségek nagymértékben megnövekedhetnek a „head-tail" méretének függvényében, az NP-nehéz jellegnek megfelelően. Így tehát egyensúlyt kell teremtenünk a költségek és a minőség között.

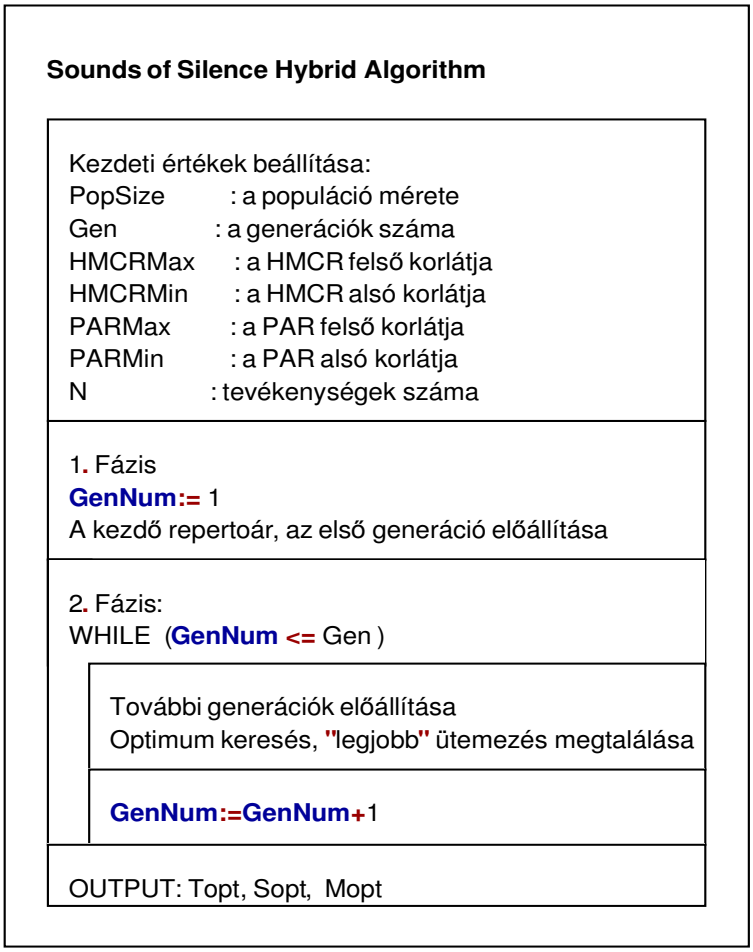
4.2.4 Az algoritmus struktogramja

Ebben a pontban az algoritmus fő lépéseit részletezzük, s a legfontosabb modulok struktogramját is bemutatjuk.

Az algoritmusban globális paramétereket definiáltunk a populáció méretének (*PopSize*), a generációk számának (Gen) rögzítésére, a hangelfogadási tényező $(HMCR)$ valamint a hangmagasság szabályozó (PAR) alsó és felső határának $(HMCRMin,\ HMCRMax,\ PARMin,\ PARMax)$ és a véletlenszám eloszlási határértékek $(MinDev,\ MaxDev)$ beállítására.

Az algoritmus két fő fázisból áll, a kezdeti repertoár feltöltési fázisból és az optimalizáló fázisból. Mindegyik fázis több lépésből áll. Az egyes fázislépésekben különböző eljárások hívásával, alkalmazásával hajtjuk végre az optimumkeresés lépéseit. Az algoritmus leállási feltételét a *Gen* paraméter értékével adjuk meg. Ha a ciklusszámláló eléri az előre definiált generációszámot, az algoritmus befejeződik.

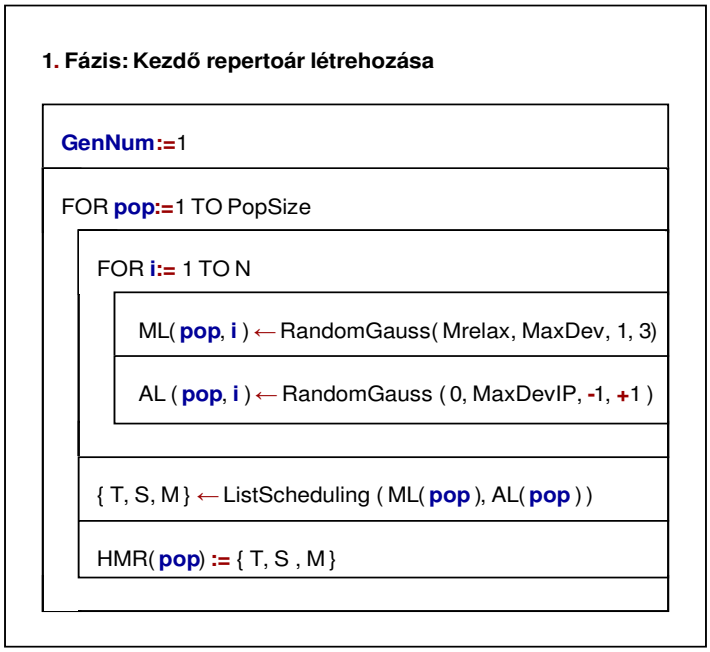
A *11. ábra* az algoritmus fő fázisait mutatja.



11. ábra Az SoS hibrid algoritmus fázisai

Az algoritmus eredményeként a több-megvalósítási módú projektre egy optimális, minimális időtartamú erőforrás korlátos ütemezést kapunk. Az optimális, minimális időtartamot Topt, az optimális ütemezést Sopt, az optimális ütemezéshez tartozó megvalósítási módokat az Mopt szimbolizálja.

A következő, *12. ábra* az algoritmus első fázisának, a kezdő repertoár létrehozásának legfontosabb lépéseit tartalmazza.



12. ábra Kezdő Repertoár előállítása

A kezdő repertoár előállításának célja a Harmónia repertoár feltöltése. A Harmónia repertoár egy mátrix, amely sorainak számát a *PopSize* értéke határozza meg, ennyi kezdeti ütemezést hozunk létre a vizsgált több-megvalósítási módú projektre vonatkozóan. A belső ciklus minden populáció minden tevékenységére (tevékenységek száma N) vonatkozóan, véletlenszerűen generál egy relaxált megvalósítási mód értéket az előoptimalizálással nyert relaxált megoldásból. Az *előoptimalizálás* egy nagyon fontos lényeges *újítása* az algoritmusunknak. Erről az újításról a későbbiekben még részletesen szó lesz. A korábbi verzióban egy ún.

truncated normális eloszlásból véletlenszerűen állítottuk elő a mód értékeket. A mód értékeken kívül minden populáció minden tevékenységére vonatkozóan, szintén véletlen szám generátor függvény alkalmazásával a tevékenység megkezdésére (mikor szándékozik a zenész belépni a dallamba) vonatkozóan adunk értékeket.

A véletlen mód és belépési időpont értékek előállítását a $\boldsymbol{RandomGauss}(\mu, \sigma, MinX, MaxX)$ véletlen szám generátor függvény végzi. A függvény egy x véletlen számot generál a $[MinX, MaxX]$ tartományban, μ középértékkel és σ szórással. A függvényt nemcsak a repertoár feltöltési fázisban, hanem az improvizáció során is használja az algoritmusunk. A függvény bemenő paraméterei a következőképpen alakulnak a megvalósítási mód, illetve a tevékenység belépési idejének generálásakor:

4. táblázat

A ***RandomGauss*** függvény paraméterei a megvalósítási módhoz

μ	σ	***MinX***	***MaxX***
Mrelax	MaxDev	1	3

A MaxDev értéke a $[0{,}1; 0{,}2]$ értéktartományból származik.

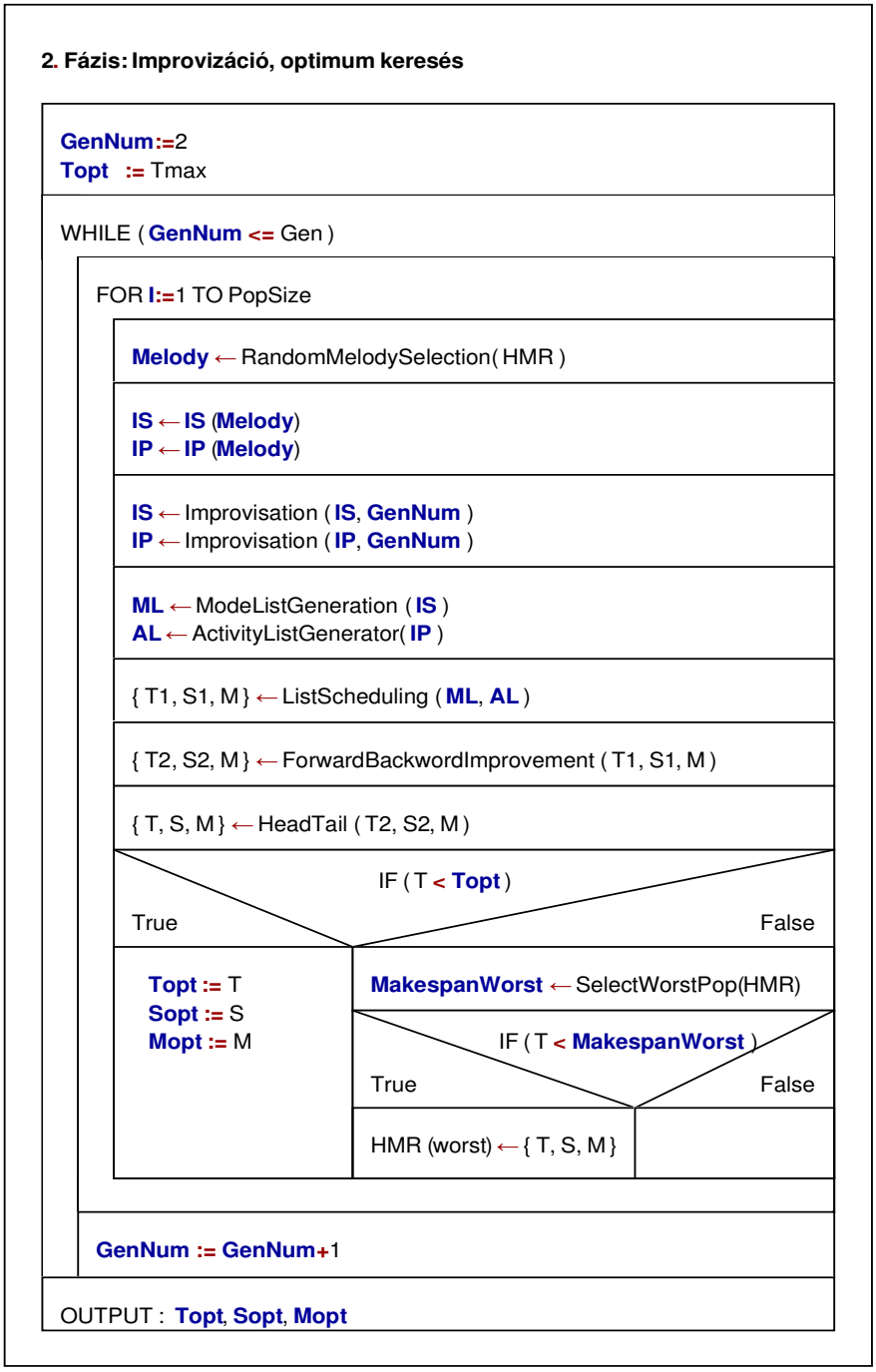
5. táblázat

A ***RandomGauss*** függvény paraméterei a tevékenységek kezdetéhez

μ	σ	***MinX***	***MaxX***
0	MaxDevIP	-1	1

A külső ciklusban a populációk tevékenységeihez generált megvalósítási módok és kezdési időpontok alapján elvégzünk egy ütemezést, a ***ListScheduling***$(ML(pop), AL(pop))$ függvény segítségével. Ezután minden populációhoz kiszámításra kerül a projekt időtartama (T). A populációhoz tartozó időtartamot (T), ütemezést (S) és megvalósítási módokat (M) tároljuk a harmónia repertoárban. Az improvizációk során ezekből a populációkból hozunk létre újabbakat és keressük közülük a célunknak megfelelő legrövidebb időtartamú, adott megvalósítási módú, erőforrás korlátos ütemezést.

A *13. ábra* az algoritmus második fázisának, az optimumkeresésnek a folyamatát mutatja.



13. ábra Improvizáció, optimumkeresés

A második fázis a repertoár feltöltése után következik. Az egyes lépéseket a teljes populációra vonatkozóan hajtjuk végre. A fázis első lépése, hogy

meghívjuk a ***RandomMelodySelection****(HMR)* eljárást, amely a repertoárból véletlenszerűen kiválaszt egy dallamot. Minél rövidebb a dallam (a projekt időtartama) annál nagyobb az esély, hogy kiválasztásra kerül.

A következő lépésben, mind a megvalósítási módra, mind a dallamba való belépés idejére vonatkozóan egy –egy elképzeléshalmazt hozunk létre.

Ezután következik az improvizáció, amit az ***Improvisation****(IS, GenNum) és* ***Improvisation****(IP, GenNum)* más-más paraméterekkel hívott függvény végez. Az improvizációban a Mahdavi (2007) által javasolt javított harmóniakereső eljárás alapján képezzük az elképzelések halmazának improvizációval módosított változatát. Az eljárás második paramétere, a *GenNum* segítségével biztosítjuk azt, hogy generációról generációra csökkenjen a „zenészek" szabadsági foka a szabad improvizálást tekintve. Ennek elvi alapja a *9.-10. ábrán* korábban látható volt.

A legfontosabb lépései a második fázisnak ezután következnek. Ez nem más, mint a megvalósítási módok valamint a tevékenységek listájának a generálása. Ezt a feladatot a ***ModeListGeneration****(IS)* és az ***ActivityListGenerator****(IP)* eljárások végzik. Az *IS* és *IP* improvizációval nyert elképzeléshalmazokból egy MILP és egy LP feladat megoldásával jön létre a két lista (*ML* és *AL*). A MILP eredménye az erőforráskorlátokat kielégítő megvalósítási módok halmaza lesz. Elméletileg ez a MILP modell egy több-választásos több-dimenziós hátizsák probléma. Az LP eredménye egy ütemezés lesz, melyet a karmester arra használ, hogy az egyes tevékenységek végső sorrendjét meghatározza.

A listák generálását követi a ***ListScheduling*** *(ML, AL)* eljárás hívása. Ebben az eljárásban egy erőforráskorlátokat kielégítő minimális időtartamú

(T1) ütemezés *(S1)* jön létre, a tevékenységekhez rendelt megvalósítási módok figyelembevételével *(M)*.

Ezután a jól ismert ***ForwardBackwardImprovement****(T1,S1,M)* (Tormos és Lova (2001)) eljárás segítségével a karmester megpróbál javítani a dallam, az ütemezés minőségén. Az eljárás kimenete egy az előzőnél valószínűleg jobb ütemezés *(T2, S2, M)* lesz. Az algoritmus korábbi változatánál egy egyszerű forward-backward eljárást alkalmaztunk, s ezt cseréltük le a hatékonyabb *FBI* eljárásra.

A karmesternek további lehetősége van a dallam, ütemezés minőségének javítására, a ***Head-Tail****(T2,S2,M)* eljárás felhasználásával. Az eljárás kimenete egy legjobb ütemezés *(T, S, M)* lesz. A helyi minőségjavító algoritmus segítségével újraoszthatjuk a nem-megújuló erőforrásokat a projekt eleje és vége között, ha ez azt eredményezi, hogy a projekt időtartama kisebb lesz.

Az algoritmus következő lépése a repertoár frissítése. Ennek célja, hogy a populációt javítsuk, a rossz minőségű dallamokat távolítsuk el, cseréljük le egy jobbra, ezzel javítva a keresési tér minőségét. Ezt megelőzi egy vizsgálat, ahol az előző lépésekkel létrehozott ütemezés időtartamát összehasonlítjuk az addigi legjobb (*Topt*) időtartammal. Amennyiben az új ütemezés időtartama kisebb, mit az addigi legjobb, akkor az új ütemezést tekintjük a legjobbnak. Ha ez a feltétel nem teljesül, megvizsgáljuk, hogy jobb-e az új ütemezés az addigi legrosszabbnál, ha igaz az állítás, akkor az addigi legrosszabb ütemezést eltávolítjuk a memóriából és az új ütemezést tekintjük az aktuális legrosszabbnak.

Ezeket a lépéseket addig folytatjuk, amíg a kilépési feltétel nem teljesül. Az algoritmus leállítása az előre megadott generációszám elérésekor

történik meg, ez a kilépési feltétel. A legjobb ütemezést az algoritmus befejeződésekor a (*Topt*, ***Sopt***, ***Mopt***) tartalmazzák.

4.2.4.1 Az előoptimalizálás

Külön említést érdemel az alkalmazás kapcsán az előoptimalizálás, hiszen ennek alkalmazásával, ahogy majd a számítási eredmények is mutatják, lényeges gyorsulást értünk el a számítási időket tekintve.

Az előoptimalizálás lényege, hogy a zenészek által az improvizáció, illetve a repertoár feltöltés során véletlenszerűen előállított tevékenység megvalósítási módok merítési bázisát szűkítsük olyan lehetőségekre, amelyek lehetséges megoldást biztosítanak. Vagyis zárjuk ki azokat a megvalósítási módokat, amelyek biztosan nem megvalósítható ütemezést eredményeznek. Az előoptimalizálásnál azonban óvatosnak kell lenni, ügyelni kell arra, hogy olyan kiinduló megoldások jöjjenek létre, amelyek nem hasonlítanak túlzottan egymásra, mert akkor a keresési tér széles változatosságát veszítjük el, s esetleg egy lokális optimum környezetében végzett értelmetlen keresés szituációjába kerülünk.

A megoldandó probléma több-korlátos 0-1 hátizsák problémaként (MKP) való kezelése bizonyult eredményesnek. Az MKP problémák megoldására néhány egzakt, heurisztikus és számos genetikus algoritmus született.

A több-korlátos 0-1 hátizsák probléma (*Multiconstrained 0-1 Knapsack Problem*, (MKP)) egy jól ismert NP-teljes kombinatorikus optimalizálási probléma, amely a következő alakban adható meg (Reidl 1998):

$$\max \mathbf{f}(x_1,\ldots,x_n) = \sum_{j=1}^{n} p_j x_j \tag{26}$$

$$C_i: \quad \sum_{j=1}^{n} w_{i,j} x_j \le b_i \tag{27}$$

$$\text{ahol, } i = 1,\ldots,m, \qquad x_j \in \{0,1\}, \qquad j = 1,\ldots,n$$

$$\text{és } p_j > 0, \quad w_{i,j} \ge 0, \quad b_i \ge 0$$

Az $\mathbf{f}(x_1,\ldots,x_n)$ célfüggvény maximumát keressük, az m számú korlátozó feltétel (C_i) mellett. A döntési változók értéke csak 0 vagy 1 lehet. Az általános 0-1 egészértékű feladatoktól abban tér el a modell, hogy a $w_{i,j}$ értékek mind pozitívak.

Számos alkalmazási területe van az MKP problémáknak, azonban szakirodalmi vizsgálataink szerint a több-megvalósítási módú erőforrás-korlátos projektek ütemezésénél még sehol sem alkalmazták, az irodalom jelenlegi állása szerint.

A hátizsák probléma „meséje" rendkívül egyszerű és szemléletes. Különböző méretű tárgyakat szeretnénk belerakni egy hátizsákba úgy, hogy minél több tárgy beleférjen, azonban a hátizsák kapacitása korlátozott. A mi feladatunk esetében a „hátizsákba" a tevékenységeket a különböző megvalósítási módjaikkal, amely más-más tevékenység időtartamot és nem-megújuló erőforrás szükségletet jelent (a tárgy méretei), próbáljuk berakni úgy, hogy minél több tevékenység (tárgy) beleférjen. Esetünkben nem a tárgyakat befogadó hátizsák mérete a korlátozó feltétel, hanem a nem-megújuló erőforrások rendelkezésre álló mennyisége.

Az ily módon történő előoptimalizálással egy jó kiinduló megoldást szeretnénk kapni, amely kizárja azokat a megvalósítási módokat, amelyek esetében a nem-megújuló erőforrásokra vonatkozó korlátok nem teljesülnének.

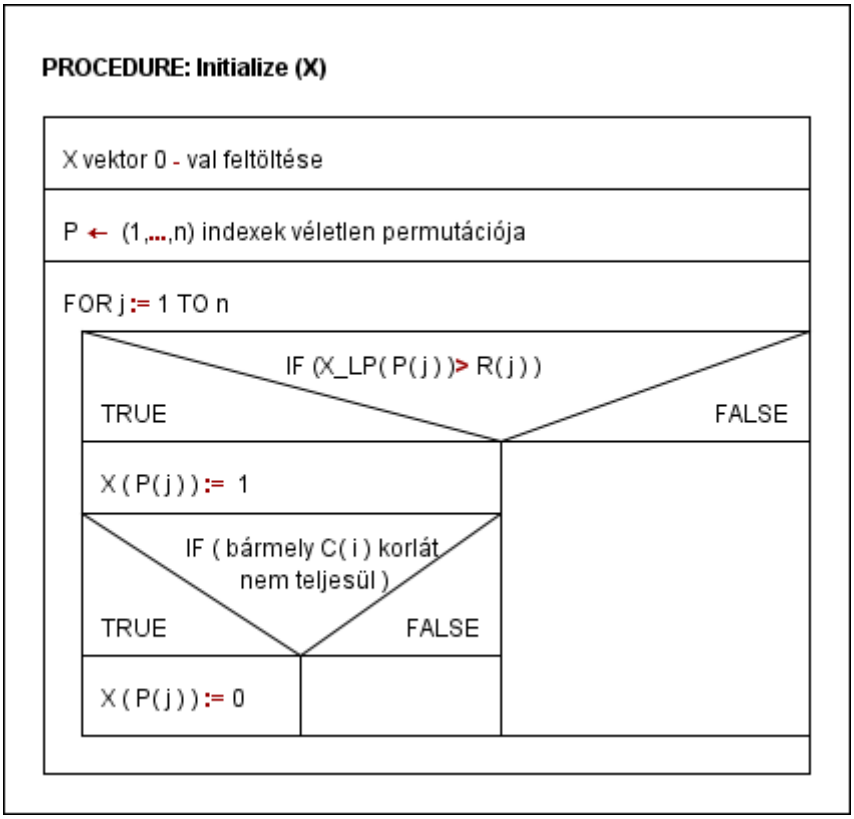
A metaherusztikus algoritmusunkba a Raidl (1998) genetikus algoritmusában alkalmazott kezdeti megoldást adó eljárását adaptáltuk. Az eljárás az MKP feladat LP-relaxált megoldásán alapszik. Ez a megközelítés természetesen nem új, de itt újdonság.

A kezdő megoldáshalmazban minden x_j változó értéke nulla. Ezután, egy véletlen P permutációját képezzük az $1,\ldots,n$ indexeknek, hogy a permutációval előállított sorrendben vizsgáljuk meg az x_j változókat.

A következő lépésben minden permutált $x_{P[j]}$ változó értékének 1 értéket adunk az LP megoldás, $x_{P[j]}^{LP}$ -vel azonos értékű valószínűséggel. $\Re_j$ pseudo-random számokat alkalmazunk erre a célra. $\left(0 \le \Re_j < 1\right)$.

Amennyiben az 1 értéket tartalmazó $x_{P[j]}$ változó bármely C_i korlátot megsért, értékét visszaállítjuk nullára. A véletlen elemeknek köszönhetően, az algoritmus különböző kezdő megoldásokat állít elő, biztosítva a populáció megfelelő diverzifikáltságát.

Az algoritmus stuktogramja a 14. ábrán látható:



14. ábra. Előoptimalizálás algoritmusa

4.3 Számítási eredmények

A számítási eredményeket az algoritmus egy 1.8 GHz Pentium IV IBM PC típusú számítógépen történő futtatásával állítottuk elő. A gép 256 MB memóriával rendelkezett és Microsoft Windows XP® operációs rendszer futott rajta.

A bemutatott algoritmus Visual C++® 6.0 és Visual Basic® 6.0 nyelven készült.

Az algoritmust a jól ismert PSLIB tesztkönyvtár J30MM alkönyvtárának (http://129.187.106.231/psplib/) projektjein teszteltük. Miért erre a teszthalmazra esett a választás? Az ok rendkívül egyszerű, mert jelenleg ez a legtöbb tevékenységet tartalmazó, nem-megújuló (elfogyó) és megújuló erőforrásokat is tartalmazó több-megvalósítási módú tevékenységeket magában foglaló projekthalmaz. Ez idő szerint a J30MM képezi a nehéz és nagy feladatokat. (Természetesen ezek a feladatok a valóságban létező nagy problémáktól messze vannak.) A J20MM jelzésű teszthalmaz egy választóvonalat képez. A 20 tevékenységnél kevesebb tevékenységet tartalmazó teszthalmazoknak optimális megoldása egzakt algoritmusokkal viszonylag könnyen előállítható. Ezek ma már nem jelentenek erőpróbát. A J20MM-tól felfelé, azaz 20 tevékenységnél többet tartalmazó több-megvalósítási módú elfogyó erőforrásokat is tartalmazó halmazok esetén csak heurisztikát alkalmazhatunk A 3. fejezetben található 2. táblázat alapján megállapítható, hogy egyik kutató által közölt módszernél sincs hivatkozás olyan teszthalmazra, amely 30 tevékenységnél többet tartalmazott volna, amennyiben nem-megújuló (elfogyó) erőforrást is

tartalmazott a projekt. Boctor készített 50 és 100 tevékenységből álló két projekthalmazt, de ezekben nincs nem-megújuló erőforrás. A táblázatban Özdamar kísérleteinél találunk 30-nál több tevékenységet, de nem használható fel összehasonlításul, mivel nem reprodukálható teszthalmazt hoztak létre eredményeik igazolására.

Ugyanakkor a kutatókban kialakult az igény arra, hogy nagyobb méretű ismert paraméterekkel rendelkező teszthalmazok jöjjenek létre, amely standardként szolgál arra, hogy összehasonlíthatókká váljanak a kutatók közzétett eredményei. A Gentben folyó kutatások Vanhoucke és Peteghem vezetésével már létrehoztak egy 50 és 100 tevékenységből álló halmazt (MMLIB50, MMLIB100, és MMLIB+) de a dolgozat megírásakor még nem álltak rendelkezésre futási eredmények és még nem ismertek a teszthalmazok pontos paraméterei. A tesztprojektek tanulmányozása alapján állítható, hogy tartalmaz valódi kihívást jelentő feladatokat. Különösen igaz ez az MMLIB+ halmazra, amely 50 és 100 tevékenységből álló projekteket tartalmaz. Minden tevékenységhez 2 vagy 4 megújuló és nem-megújuló erőforrás rendelhető hozzá, és minden tevékenységnek 3, 6 vagy 9 megvalósítási módja van. A tesztprojektek letölthetők a www.projectmanagement.ugent.be/mrcpsp.html oldalról. Ezek az új teszthalmazok további, jövőbeli kutatások alapját képezhetik.

Az általunk a PSPLIB könyvtárból kiválasztott a J30MM teszthalmazban a projektpéldányok 30 valódi tevékenységet tartalmaznak, s az egyes tevékenységek két megújuló és két nem-megújuló erőforrást igényelnek. Mindegyik valódi tevékenység három megvalósítási móddal rendelkezik. A J30MM halmaz 640 esetet tartalmaz, amely 64 projekt 10 ismétlése. Néhány esetnek nincs, az elsőbbségi feltételeknek és az erőforráskorlátoknak megfelelő, lehetséges megoldása, ezeket kizártuk a

vizsgálatból. Ebben a halmazban nem ismert az összes optimális megoldás, így először ezeket az egzakt megoldásokat állítottuk elő.

Az egzakt megoldások generálására a jól ismert MILP megoldó szoftvert (CPLEX 8.1) használtuk, alapértelmezett beállításokkal. A megoldásra szánt időt 900 sec időkorlátban állapítottuk meg. A megadott időkorláton belül 382 esetben sikerült az optimális megoldást elérni a CPLEX használatával, amely jól jelzi a több megvalósítási módú erőforrás-korlátos (MRCPSP) projektütemezési feladatok nehézségi fokát, NP-hard jellegét.

A J30MM teszthalmaz futtatási eredményeit három csoportra osztottuk a következő tényezők alapján:

U (Unfeasible) csoport: Ebbe a csoportba tartoznak azok az esetek, amelyekre a 900 másodperces időkorlátot alkalmazva, a CPLEX-et használó eljárás nem tudott megvalósítható (feasible), azaz az elsőbbségi feltételeknek és az erőforráskorlátoknak megfelelő megoldást adni. Ezeket az eseteket kizártuk a további vizsgálatból.

F (Feasible) csoport: Ebbe a csoportba tartoznak azok az esetek, amelyekre a 900 másodperces időkorláton belül nem kaptunk optimális megoldást, de lehetséges, megvalósítható megoldást igen. Ezekre az esetekre is megmutatjuk az algoritmusunk hatékonyságát.

O (Optimal) csoport: Ebbe a csoportba tartoznak azok az esetek, amelyekre a CPLEX –et használó eljárás optimális megoldást adott a 900 másodperces időkorláton belül. Erre a csoportra vonatkozóan megadjuk a számítási eredményeket, hogy szemléltessük az algoritmusunk hatékonyságát és gyorsaságát.

A második (**F**easible) és a harmadik (**O**ptimal) csoportba tartozó eseteket a következő pontokban részletesen megvizsgáljuk.

4.3.1 Az időkorláton belül optimális megoldást adó esetek (O csoport) futási eredményei

Ebben a pontban harmadik csoportba tartozó eseteknek a vizsgálatával foglalkozunk, ahol a megadott, 900 sec-os időkorláton belül a CPLEX –et használó egzakt algoritmus optimális megoldást adott. A 640 J30MM esetből 382 eset volt ilyen. Ez az eredmény is mutatja a probléma nehézségi fokát, NP-hard jellegét. A futtatások során a generáció értékét 10-nek, a populáció méretét 10-nek választottuk. Így az ismétlések, iterációk száma 100.

Az O csoportra vonatkozó paraméter beállításokat az ***6***. *táblázatban* foglaltuk össze.

6. Táblázat

O csoportra vonatkozó futási paraméter értékek

Paraméter név	**Érték**
Populáció mérete	10
Generációk száma	10
Iterációk száma	100
Független futások száma	30
Min. Repertoire Rate, Max. Repertoire Rate	0,1 ; 0,9
Min. Adjusting Rate, Max. Adjusting Rate	0,1 ; 0,9

A *7. táblázatban* a hibrid algoritmusunk futási eredményeit foglaltuk össze. A táblázat sorai tükrözik az algoritmus továbbfejlesztése során elért

eredményeket, mely fejlesztések minőségi javulást eredményeztek az algoritmus képességeiben. A táblázatban az eredmények minőségét az egzakt optimális megoldástól való átlagos eltérés százalékában, illetve a futási idők átlagában adjuk meg. A standard repertoár alkalmazásával is a futási idők átlaga egy nagyságrenddel kisebb, mint az egzakt eljárással kapott megoldások átlagos futási ideje. Jól látható, hogy az előoptimalizálás minőségileg és a futási időket tekintve is jelentősen megnövelte az algoritmus hatékonyságát. Az algoritmus további bővítése a head-tail eljárás alkalmazásával, újabb javulást hozott a megoldás minőségében, azonban ez együtt járt a megoldási idő növekedésének elfogadható mértékével. A táblázat utolsó sorából igazolódni látszik az az állításunk, hogy a head-tail méretének a függvényében nő a számítás ideje is. A számok azt mutatják, hogy a head-tail méretét 2-re növelve, a megoldás minősége egy nagyságrendet javult, mindössze 0,5%-al tért el az optimális megoldás értékétől, azonban ez a minőségi javulás a számítási idő rovására történt.

7. Táblázat

„Sounds of Silence" eredmények

Iterációk	Módszer	Megoldás minősége Eltérés (%)	Megoldási idő	
			μ (sec)	σ (sec)
100	standard repertoár	8,14	3,581	7,781
100	elő-optimalizált repertoár	1,53	1,550	2,779
100	elő-optimalizált repertoár + head-tail -1	1,38	4,140	6,614
100	elő-optimalizált repertoár + head-tail -2	0,50	24,267	9,345

A 8. *táblázat* a CPLEX által adott egzakt megoldások minőségi paramétereit, a futási idők átlagát, szórását, illetve a minimális és maximális futási időket tartalmazza.

A futási idők nagy szórást mutatnak, a szinte azonnali optimumot adó megoldástól a 665 másodpercig tartanak. Érdekes, hogy a megadott 900 másodpercnyi korlátot egyik esetben sem közelítette meg a futás ideje. Vagy jóval korábban leállt az eljárás, vagy nem volt elegendő a 900 másodperc, és leállítottuk a futást, s így vagy az **U** csoportba vagy a **F**-be került a vizsgált eset.

8. Táblázat

Megoldási idő CPLEX 8.1 (sec)

Átlag	**Szórás**	**Minimális idő**	**Maximális idő**	**Megoldott esetek**
25,44	98,82	0,02	665,00	382

Az átlagos futási idő értéke egy nagyságrenddel nagyobb, mint a hibrid algoritmusunké, ami megerősíti azt az állítást, hogy hatékony, gyors algoritmus a Sounds of Silence algoritmus. Megállapítható, hogy az adaptáció a több-megvalósítási módú projektek ütemezésére eredményes volt.

4.3.2 Az időkorláton belül lehetséges megoldást adó esetek (F csoport) futási eredményei

Ebben a pontban a J30MM teszthalmaz második csoportba sorolt eseteinek futási eredményeit részletezzük. Erre a csoportra az a jellemző, hogy a CPLEX-et használó egzakt eljárás a megadott 900 sec időkorláton belül nem volt képes optimális megoldást adni, de lehetséges, megvalósítható megoldást igen.

Minden projekt előfordulásra vonatkozóan 30 egymástól független futtatást végeztünk, hogy az eredményeink statisztikailag szignifikánsak legyenek. Mind a generáció mind a populáció értékét 10-nek választottuk, az iterációk száma ennek következtében 100. A *9. Táblázat* tartalmazza a hibrid algoritmusunk futási paramétereit.

9. Táblázat

Paraméter értékek a F csoportra vonatkozóan

Paraméter név	Érték
Populáció mérete	10
Generációk száma	10
Iterációk száma	100
Független futások száma	30
Min. Repertoire Rate, Max. Repertoire Rate	0,1 ; 0,9
Min. Adjusting Rate, Max. Adjusting Rate	0.1 ; 0,9

A hibrid eljárásunk és a CPLEX-et használó eljárás eredményeinek összehasonlíthatósága érdekében egy minőségi mutatót vezettünk be. A megoldás minőségét a projekt időszükségletének és az optimális időszükséglet értékének százalékos eltérésével mértük (QM). A QM értékét a következő összefüggéssel számítottuk ki:

$$QualityMeasure = 100 \cdot \frac{PrimaryMeasure - CplexPrimaryMeasure}{(CplexPrimaryMeasure - 1)} \qquad (28)$$

ahol a *PrimaryMeasure* a projekt időtartama és a *CplexPrimaryMeasure* egy jó megoldási érték.

Az osztályozásunk alapján a második (F) csoportba a J30MM teszthalmaz következő 28 eleme került:

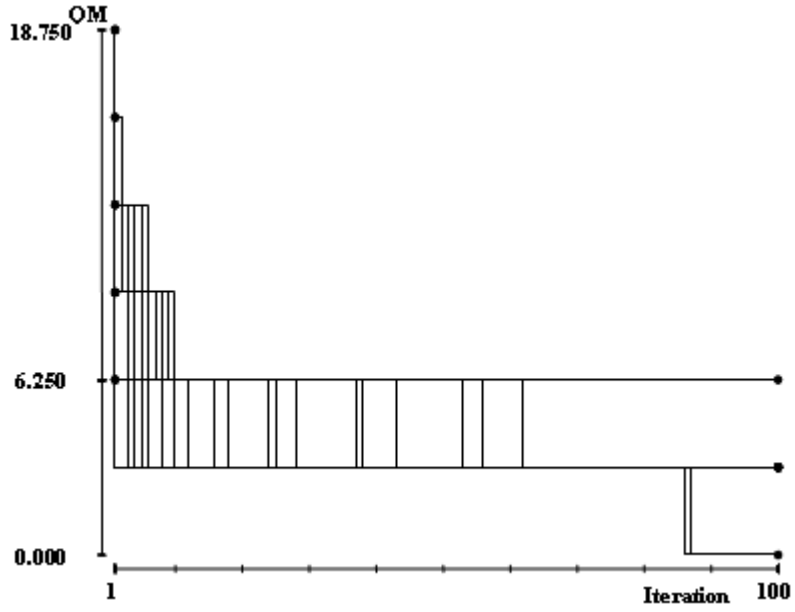
{ 14-2, 14-10, 22-2, 22-8, 25-1, 34-5, 35-9, 38-10, 39-3, 39-5,39-8, 40-1, 40-3, 41-5, 42-2, 42-5, 42-7, 43,6, 46-1, 46-3, 46-5, 46-9, 47-3, 47-5, 48-3, 54-3, 62-2, 62-3}

A kötőjel előtti szám a 64 projektegyed előfordulás valamelyikét, a kötőjel utáni szám a 10 ismétlés valamelyikét jelöli.

Mielőtt a csoport eredményeit részletezzük, fontosnak tartjuk, hogy legalább egy egyedet az ebbe a csoportba tartozó egyedekből részletesebben megvizsgáljunk. A sorrendben elsőt, a J30MM-14-2 esetet választottuk. Ennek az esetnek mind a 30 független futásának az eredményét bemutatjuk, táblázatos és grafikus formában egyaránt, hogy hatásosan tudjuk szemléltetni az algoritmus jó tulajdonságait.

4.3.2.1 A J30MM-14-2 példány futási eredményei

A J30MM-14-2 példány 30 független futásának eredménye látható az *15. ábrán* és a futási adatok a 10. táblázatban. A grafikon a 100 iterációs lépés alakulását ábrázolja a 30 független futtatásra vonatkozóan. A *QualityMeasure*(*QM*) értéke a függőleges tengelyen látható, míg az iterációk számát (*population***generation*) a vízszintes tengelyen ábrázoltuk.



15. ábra 30 független futás eredménye a J30MM-14-2 esetre

A grafikonon jól látható, hogy a *QM* értéke meglehetősen gyorsan konvergál három különböző értékhez (6,25; 3,12; 0,00), növekvő iteráció szám mellett. Ez nagyon jó eredmény 100 iteráció és 30 független futásra vonatkozóan. Hasonló grafikonokat kaptunk a csoport többi esetére vonatkozóan is.

Az ábrán látható eredmények jól tükröződnek a *10. táblázatból* is. A *QM* értéke 6,25 és 0,00 mindössze két-két esetben volt a 30 független futásból. A 30 független futásból 26 esetben a QM értéke azonos, 3,12 lett, ami azt mutatja, hogy az algoritmus stabil és robosztus.

10. Táblázat

Iteráció története a J30MM-14-2 példány 30 független futására

Futások száma	**CPLEX Megoldás**	**Hibrid eljárás**	**QM (%)**
1	33	34	3,12
2		35	6,25
3		34	3,12
4		34	3,12
5		34	3,12
6		34	3,12
7		34	3,12
8		34	3,12
9		34	3,12
10		35	6,25
11		34	3,12
12		34	3,12
13		34	3,12
14		34	3,12
15		34	3,12
16		34	3,12
17		34	3,12
18		34	3,12
19		34	3,12
20		33	0,00
21		34	3,12
22		34	3,12
23		34	3,12
24		34	3,12
25		34	3,12
26		34	3,12
27		34	3,12
28		34	3,12
29		33	0,00
30		34	3,12

Az átlagos CPU idő 3,19 másodperc, a legkisebb érték 2,391 másodperc, míg a leghosszabb futási idő 4,093 másodperc volt. Ezeket az értékeket a *11. táblázatban* foglaltuk össze.

11. Táblázat

J30MM-14-2 eset megoldási ideje (sec)

Átlag	Szórás	Minimális idő	Maximális idő
3,192	0,499	2,391	4,093

Az eredmények azt igazolják, hogy az algoritmus nemcsak robosztus, hanem gyors is.

A következő pontban a teljes, második (F) csoportba sorolt esetek futási eredményeit foglaltuk össze.

4.3.2.2 A második csoport (F csoport) összes példányára vonatkozó futási eredmények

A *12. táblázatban* a *Quality Measure* értékek alakulása látható. A minimális, a maximális és az átlagos eltéréseket, valamint a szórást mutatjuk be 30 egymástól független futtatásra vonatkozóan, standard repertoár esetében. A negatív értékek azt mutatják, hogy a szimplex eredménynél jobb megoldást talált a bemutatott hibrid algoritmus.

12. Táblázat

Minőségi mutató (QM) értékek 30 független futtatásra

Példány neve	Átlag QM (%)	Szórás	Minimum QM (%)	Maximum QM (%)
J30MM-14-2	3,12	1,1606	0,00	6,25
J30MM-14-10	1,73	1,7545	0,00	3,45
J30MM-22-2	-5,13	0,0000	-5,13	-5,13
J30MM-22-8	8,69	1,3155	3,03	9,09
J30MM-25-1	8,82	3,3684	2,94	14,71
J30MM-34-5	5,56	1,5192	2,22	6,67
J30MM-35-9	-2,70	0,0000	-2,70	-2,70
J30MM-38-10	-7,50	0,0000	-7,50	-7,50
J30MM-39-3	2,07	0,3651	2,00	4,00
J30MM-39-5	-10,81	0,0000	-10,81	-10,81
J30MM-39-8	-2,22	0,0000	-2,22	-2,22
J30MM-40-1	-5,45	1,0496	-7,32	-4,88
J30MM-40-3	1,39	2,1596	0,00	5,56
J30MM-41-5	7,50	0,0000	7,50	7,50
J30MM-42-2	-4,37	1,5517	-6,90	-3,45
J30MM-42-5	6,00	0,8727	5,71	8,57
J30MM-42-7	-1,41	2,6069	-6,06	3,03
J30MM-43-6	6,32	2,7281	0,00	10,34
J30MM-46-1	2,10	1,4887	0,00	5,71
J30MM-46-3	-9,65	3,1945	-15,79	-5,26
J30MM-46-5	7,15	2,5236	2,94	11,76
J30MM-46-9	-0,08	0,4144	-2,27	0,00
J30MM-47-3	3,01	2,3862	-3,23	6,45
J30MM-47-5	0,00	0,0000	0,00	0,00
J30MM-48-3	-3,06	2,7255	-5,41	2,70
J30MM-54-3	-6,93	2,7660	-12,00	0,00
J30MM-62-2	0,00	0,0000	0,00	0,00
J30MM-62-3	2,70	1,7945	0,00	3,85

A táblázatban látható értékek azt igazolják, hogy a J30MM-22-2, J30MM-35-9, J30MM-38-10, J30MM-39-5, J30MM-39-8, J30MM-40-1, J30MM-42-2, és a J30MM-46-3 esetekre vonatkozóan a hibrid algoritmus a szimplex eredménynél jobb eredményt talált.

Fontos megjegyezni, hogy a J30MM-22-2, J30MM-35-9, J30MM-38-10, J30MM-39-5, J30MM-39-8 esetekben a szórás értéke 0,0000, ami azt jelzi, hogy mind a 30 független futtatás ugyanazt a megoldást adta. A J30MM-47-5 és J30MM-62-30 esetekben a szórás szintén 0,000, és a 30 független futásnál kapott eredmény megegyezik a CPLEX által adott eredménnyel.

Mindez a bemutatott hibrid algoritmus robosztus és hatékony tulajdonságát bizonyítja.

Az *13. táblázatban* a vizsgált esetek CPU felhasználási idejét tesszük közzé, szintén 30 egymástól független futásokra vonatkozóan.

13. Táblázat

Felhasznált CPU idő(sec) 30 egymástól független futtatásra vonatkozóan

Példány neve	Átlag	Szórás	Min Idő	Max Idő
J30MM-14-2	3,1921	0,4990	2,3910	4,0930
J30MM-14-10	1,2883	0,6468	0,7490	3,5540
J30MM-22-2	0,4259	0,0828	0,2520	0,6060
J30MM-22-8	0,4336	0,0995	0,1860	0,6420
J30MM-25-1	0,3976	0,0772	0,2050	0,5270
J30MM-34-5	4,4782	0,6140	3,4280	5,8310
J30MM-35-9	7,1133	0,9328	5,4790	9,4320
J30MM-38-10	3,7818	0,3704	2,8890	4,5220
J30MM-39-3	8,2220	1,2037	6,3240	11,4020
J30MM-39-5	3,0148	0,3526	2,343	4,0290
J30MM-39-8	1,8616	0,2755	1,4050	2,6730
J30MM-40-1	8,1606	1,2722	5,8620	10,9770
J30MM-40-3	5,6851	0,8806	4,0190	7,8760
J30MM-41-5	5,2309	0,8387	3,4060	7,8730
J30MM-42-2	8,9140	2,1945	5,7620	15,2710
J30MM-42-5	16,8075	6,8167	8,9740	44,1380
J30MM-42-7	3,9662	0,6649	3,1460	6,4170
J30MM-43-6	6,6160	1,96654	3,3730	10,7850
J30MM-46-1	4,9304	0,6255	3,7170	6,4900
J30MM-46-3	5,4374	1,0986	3,3970	7,6860
J30MM-46-5	5,8117	1,1988	4,0500	8,7030
J30MM-46-9	4,9696	0,3703	3,8870	5,7170
J30MM-47-3	6,2602	1,2650	4,2940	9,7590
J30MM-47-5	6,9102	1,4175	5,0200	11,2050
J30MM-48-3	4,4953	0,5164	3,4510	5,5500
J30MM-54-3	9,0400	2,2133	6,2750	16,8730
J30MM-62-2	0,4501	0,0740	0,2940	0,6100
J30MM-62-3	0,4831	0,0676	0,3560	0,6870
Átlagok:	4,9421	1,2922	0,1860	44,1380

Öt esetben (J30MM-22-2, J30MM-22-8, J30MM-25-1, J30MM-62-2 és J30MM-62-3) a maximális futási idő (CPU idő) 1 másodpercnél kisebb volt. Hat esetben (J30MM-39-3, J30MM-40-1, J30MM-42-2, J30MM-43-6 J30MM-47-5 J30MM-54-3) a maximális futási idő 10 és 17 másodperc közé esett. A J30MM-42-5 eset kivételével, az átlagos futási idők 10 másodpercnél kisebbek voltak. Az átlagos futási idő az összes előfordulás 30 független futtatására vonatkozóan 4,9421 másodperc volt.

Összességében a J30MM teszthalmazon végzett futtatások eredménye alapján állíthatjuk, hogy mind a második (**F** csoport), mind a harmadik (**O** csoport) eseteinek részletes vizsgálata megerősíti azt az állításunkat, hogy algoritmusunk gyors, hatékony és robosztus.

5. Az algoritmus továbbfejlesztése másodlagos szempont szerinti optimalizálásra

A továbbiakban az algoritmus továbbfejlesztési lehetőségeit részletezzük, bemutatva a továbbfejlesztésben eddig elért eredményeket.

Ha rögzítjük a tevékenységek megvalósítási módjait, amelyeket az optimalizálás során kaptunk, valamint a projekt időtartamát, felmerül annak a lehetősége, hogy valamilyen másodlagos szempont alapján, további optimalizálási kritériumokat adjunk meg. A megvalósítási módok és a projekt időtartamának rögzítésével marad annyi szabadsági foka a tevékenységek mozgathatóságának, hogy másodlagos optimalizálási kritériumot kielégítő optimális megoldáshoz jussunk, természetesen az erőforráskorlátok megsértése nélkül. Olyan hozzárendelést kell definiálnunk, amiben nincs rejtett megszakítás, nincs rejtett konfliktus. A cél tehát egy konfliktusjavító modell alkalmazása. A dolgozatban szereplő konfliktusjavító modell több-megvalósítási módú adaptációja a Láng Blanka Phd. dolgozatában (Pécs, 2010) szereplő konfliktusjavító modellnek, hiszen a két kutatómunka közös tőről fakad.

5.1 Rejtett konfliktusok kiküszöbölése

A Sounds of Silence algoritmus további bővítésként a rejtett erőforrás felhasználási konfliktusokat elsőbbségi kapcsolatok beépítésével oldja fel. A „forward-backward listaütemező eljárás” eredménye egy olyan ütemezés, amelyben az összes olyan tevékenység, amely mozgatható, mozgatható lesz az erőforrás korlátok megsértése nélkül. Az így kapott

erőforráskorlátoknak megfelelő megoldáshalmazon elvégezhetjük a másodlagos kritérium szerinti optimalizálást.

5.2 Erőforrás felhasználás kiegyenlítése, simítása

Az algoritmus továbbfejlesztésének egyik fő eleme az erőforrások felhasználási módjának kezelése.

Az erőforrásprofil alakjának „szépsége” nem csak esztétikailag fontos, hanem a projektvezetés szempontjából is. A menedzsment érdeke az, hogy egy munkát lehetőség szerint ugyanaz a munkás végezzen el, menet közben ne változzék a tevékenységet végrehajtó erőforrás. Az is rendkívül fontos szempont, hogy az erőforrások lehetőség szerint egyenletesen legyenek felhasználva, ne legyenek tétlen várakozási idők, majd újbóli munkavégzés, hiszen ez a gyakorlatban általában plusz költségekkel jár. Modellünk szempontjából másodlagos optimalizálási kritériumként azt a célt tűztük ki, hogy a megvalósítási módokat, az ütemezés sorrendjét és a projekt időtartamát rögzítve, minimalizáljuk az erőforrásegységek indítási-újraindítási eseményeinek számát az előző pontban megfogalmazott mozgatható tevékenységek halmazán, mely mozgatható tevékenységek nem sértik az erőforráskorlátokat. Ezzel a kritériummal a folyamatos munkát részesítjük előnyben, s a választott mérték jól szimbolizálja a hatékony erőforrás felhasználást.

A modellünk, függetlenül az erőforrásprofil simításától, egy másik fontos gyakorlati problémát is megold, mégpedig azt a jogos követelményt, hogy egy tevékenység egységnyi erőforrásigényét pontosan egy erőforrás elégítse ki. Ezzel a követelménnyel a dedikált erőforrás hozzárendelési probléma megoldására is felkészítettük az algoritmust.

Ebben a fejezetben ismertetett továbbfejlesztett modell célja a következőképpen fogalmazható meg. Keressük a több-megvalósítási módú erőforrás-korlátos projekt legjobb, azaz minimális időtartamú ütemezését, melyen az erőforrás felhasználási hisztogram alakja legjobban megközelíti az ideális téglalap alakot.

Az új modell alkalmassá teszi az algoritmusunkat arra, hogy egy optimális ütemezésből kiindulva, rögzítve a megvalósítási módot, a tevékenységek sorrendjét, valamint a projekt időtartamát, anélkül, hogy megsértenénk az erőforráskorlátokat a tevékenységek mozgatásával kiválasszunk egy olyan ütemezést a lehetséges ütemezések halmazából, amely a második optimalizálási kritériumnak is megfelel.

A továbbiakban bemutatjuk az új modellünket, az algoritmus bővítését és a számítási eredményeket.

Az algoritmus a Sounds of Silence algoritmus eddig bemutatott konfliktusjavító változatára épült, amely egy harmóniakereső metaheurisztika és egy MILP formulán alapuló erőforrás kiegyenlítő-hozzárendelő eljárás kombinációja. Ez a kiegyenlítő-hozzárendelő eljárás Csébfalvi és Konstantinidis (1998) megközelítésén alapszik.

A másodlagos kritérium bevezetése kihasználja az ütemezésben rejlő tartalékot, és a mozgatható tevékenységek halmazán, a tevékenységek mozgatásával az erőforrás felhasználási hisztogram kisimítását teszi lehetővé. A másodlagos kritérium szerinti optimalizálást végző algoritmus más heurisztikus keretben is alkalmazható, túlmutat az alkalmazott heurisztika keretein, önmagában is megáll.

5.2.1 Az erőforrás kiegyenlítési probléma modellje:

$$\min\left[LM = \sum_{r=1}^{R} \sum_{i=1}^{T} CU_{rt}^{+} \right] = LM^{*} \tag{29}$$

$$X_i + D_i \le X_j,\ i \to j \in \boldsymbol{PS}^{*} \tag{30}$$

$$X_i = \sum_{t \in T_i} X_{it} * t,\ T_i = \left\{ \underline{X}_i, \underline{X}_i + 1, \ldots, \overline{X}_i \right\},\ i \in \{1,2,...,N\} \tag{31}$$

$$\sum_{t \in T_i} X_{it} = 1,\ X_{it} \in \{0,1\},\ i \in \{1,2,...,N\} \tag{32}$$

$$A_t = \left\{ i \,\middle|\, X_i \le t < X_i + D_i, i \in \{1,2,...,N\} \right\},\ t \in \{1,2,...,T\} \tag{33}$$

$$U_{tr} = \sum_{i \in A_t} R_{ir},\ t \in \{1,2,...,T\},\ r \in \{1,2,...,R\} \tag{34}$$

$$U_{t\,\mathrm{r}} - CU_{tr}^{+} + CU_{tr}^{-} = U_{t-1\,\mathrm{r}},\ t \in \{2,3,...,T\},\ r \in \{1,2,...,R\} \tag{35}$$

$$U_{1\,\mathrm{r}} - CU_{1r}^{+} = 0,\ r \in \{1,2,...,R\} \tag{36}$$

$$X_{it} \in \{0,1\},\ t \in T_i,\ i \in \{1,2,...,N\} \tag{37}$$

A modellben CU_{tr}^{+} (CU_{tr}^{-}) jelöli a t időperiódusban újrainduló (leálló) erőforrásegységek számát az r-edik erőforrásból. $\boldsymbol{PS}^{*}$ jelöli a lokális konfliktusjavító eljárás után kapott megelőző-rákövetkező relációk

halmazát. Bevezetésre került egy az i-dik tevékenységhez tartozó időablak, $T_i = \{\underline{X}_i, \underline{X}_i + 1, \ldots, \overline{X}_i\}$, ahol X_i X_i az i tevékenység legkorábbi/legkésőbbi kezdési ideje rögzített megvalósítási mód, és rögzített projekt időtartam mellett.

A „dedikált erőforrás hozzárendelési" modell a következőképpen fogalmazható meg. Tekintettel arra, hogy az erőforrás kiegyenlítő eljárás pontosan megadja a tevékenységek kezdési idejét, így csak arra kell törekednünk, hogy minden erőforrásigényt kielégítsünk és ügyeljünk arra, hogy minden erőforrásigényt legfeljebb egy erőforrásegységhez rendeljük hozzá. Ez a probléma az „egységnyi idejű végrehajtási modell" (UET, unit execution time) változata, így polinomiális idő alatt megoldható.

5.2.2 A bővített algoritmus

A továbbiakban a korábban bemutatott algoritmus bővített változatát ismertetjük. Azokat a lényeges lépéseket említjük meg, amelyben az algoritmus eltér a korábbitól.

A továbbfejlesztett algoritmusban a harmóniakeresést egy MILP formulán alapuló az erőforráskiegyenlítés-hozzárendelést végző eljárással bővítettük. Hogy ne növekedjék az algoritmus számítási idő igénye túlzottan, a hibrid algoritmusunk „karmestere" csak akkor hívja meg ezt az eljárást, ha az optimalizálás során az éppen aktuális „legjobb" megoldást cseréljük le egy még jobbra. Ekkor a javított erőforrás-korlátos megoldáshalmazon, - rögzítve a tevékenységek megvalósítási módját, az ütemezés sorrendjét és a projekt időtartamát az éppen aktuális optimális megoldásnak megfelelően - megoldjuk az erőforrás kiegyenlítési problémát. A tevékenységek megvalósítási módjának valamint a projekt időtartamának rögzítésével

marad még valamennyi mozgástér, hogy a tevékenységek mozgatásával az erőforrás felhasználási hisztogramot kisimítsuk.

Az algoritmus kihasználja azt a tényt, hogy az erőforrás kiegyenlítés és a dedikált erőforrás hozzárendelési probléma egymástól teljesen függetlenül kezelhető, mivel az adott erőforrás kiegyenlítő mérték invariáns az erőforrásegységek permutációjára, a dedikált erőforrás hozzárendelés nem képes megváltoztatni az erőforrás kiegyenlítési mértéket. Az alkalmazott kisimító eljárás a folyamatos munkát részesíti előnyben és minimalizálja az induló-újrainduló dedikált erőforrásegységek számát.

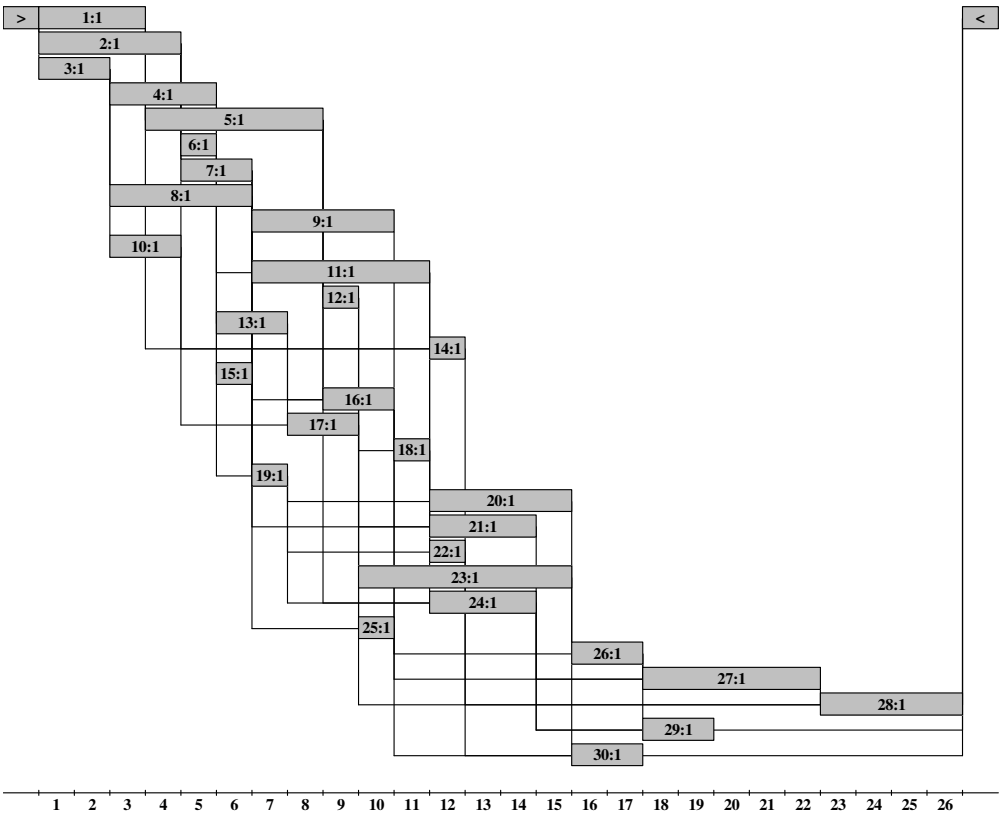
5.2.3 Az erőforrás kiegyenlítési modell számítási eredményei

A továbbfejlesztett algoritmus képességének bizonyítására a már korábban bemutatott J30MM halmazon, annak is a J30MM-10-1 példányára vonatkozó eredményeket adunk meg.

A *16.* és *17. ábrán* a projekt korai ütemezése és erőforrás felhasználási hisztogramja látható az első megvalósítási módra vonatkozóan. Az ábrákon a korábban már ismertetett jelölési módokat alkalmaztuk. A valódi tevékenységeket téglalapok jelölik, a téglalapokban látható első szám a tevékenység sorszámára, míg a második szám a megvalósítási módra utal. A téglalapok hossza arányos a tevékenységek időtartamával. Az időperiódusok száma a vízszintes tengelyen látható. A projekt kezdetét és végét szimbolizáló áltevékenységeknek nincs megvalósítási módjuk, és időtartamuk nulla. Az elsőbbségi feltételeket folytonos vonallal szemléltetjük.

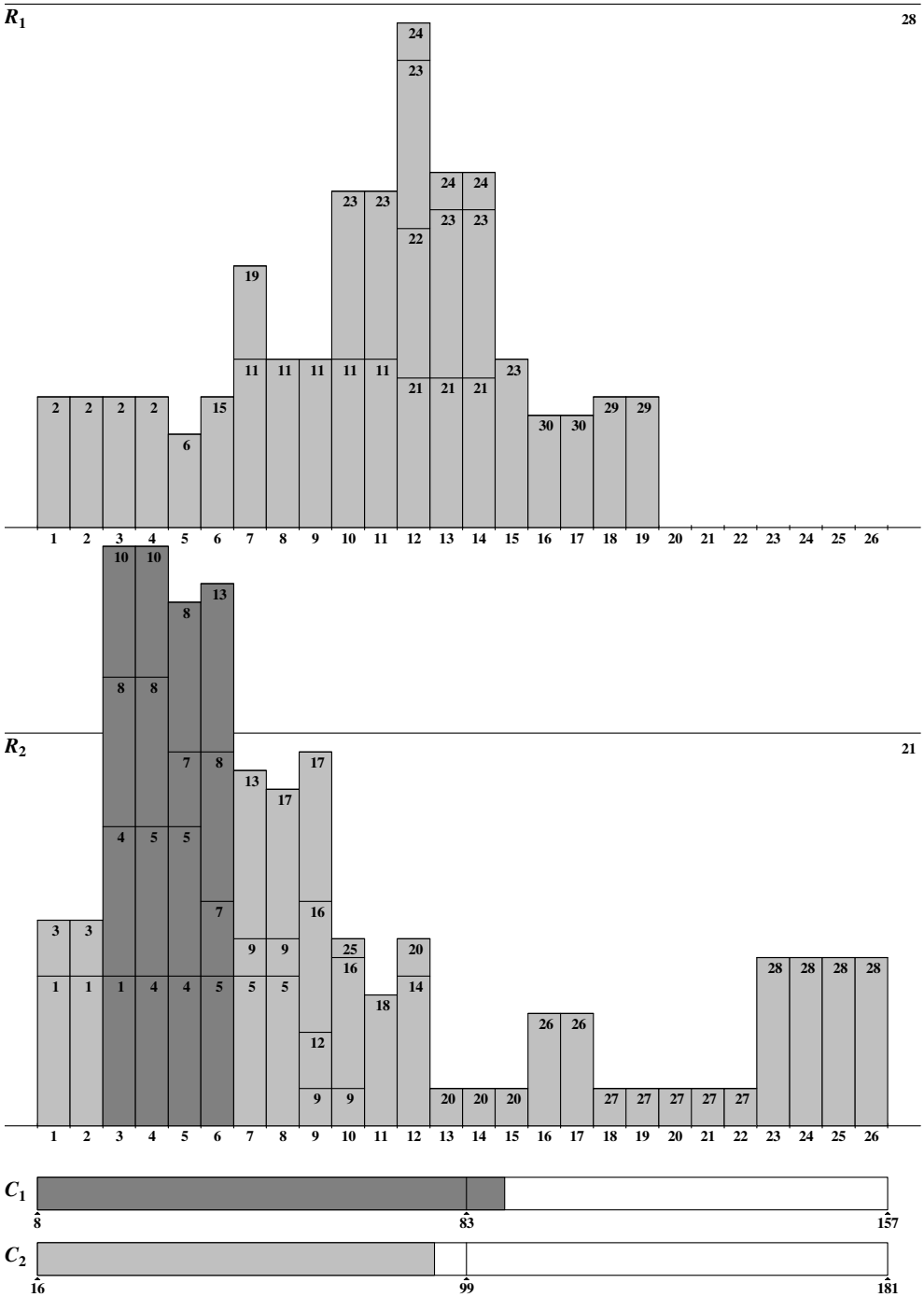
Az erőforrás felhasználási hisztogramokon az erőforráskorlátokat meghaladó erőforrás felhasználást sötétszürke téglalap jelöli. A téglalapokban szereplő számok annak a tevékenységnek a sorszámát mutatják, amelyhez az adott időperiódusban az erőforrás hozzárendelésre került. A hisztogramon egy vízszintes vonal jelzi az erőforráskorlátot. Az ábra alján, a lefektetett hisztogramok a nem-megújuló erőforrások felhasználásának alakulását szemléltetik.

A korai ütemezés időtartama 26 időegység.



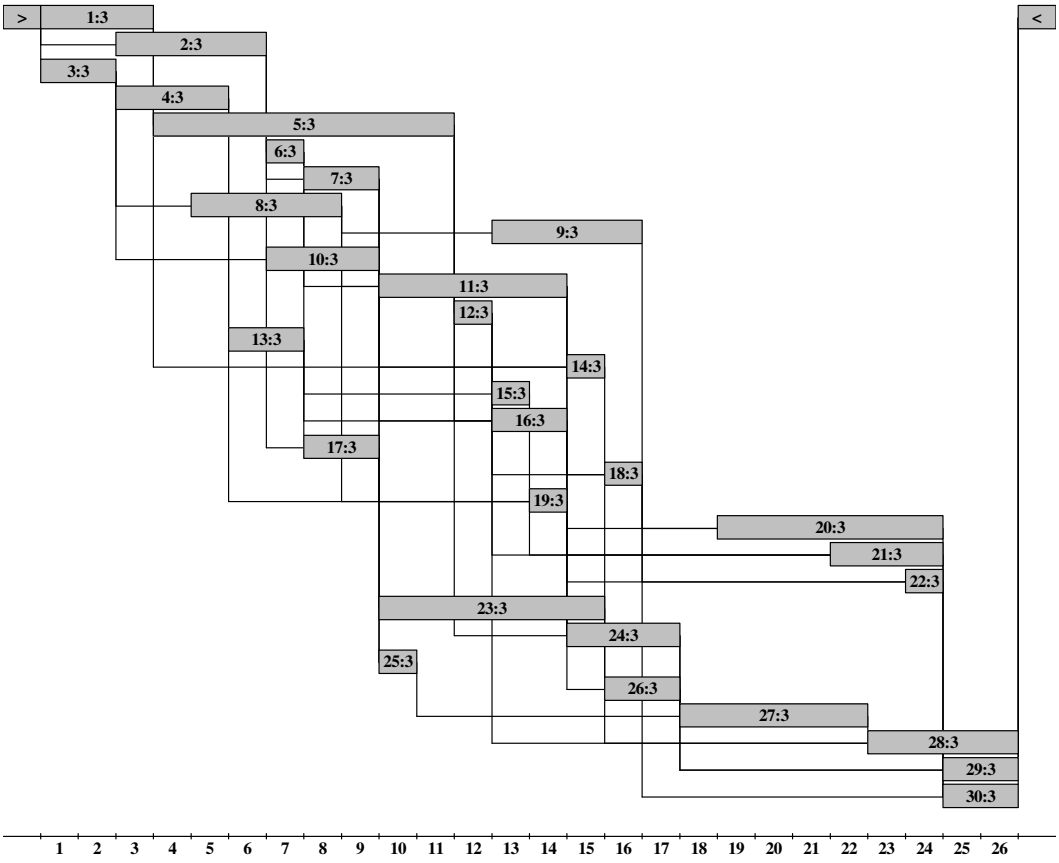
16. ábra Korai ütemezés az első megvalósítási módra, a J30MM-10-1 esetre

Az ábráról jól látható, hogy a legkorábbi ütemezés nem erőforrás korlátos.

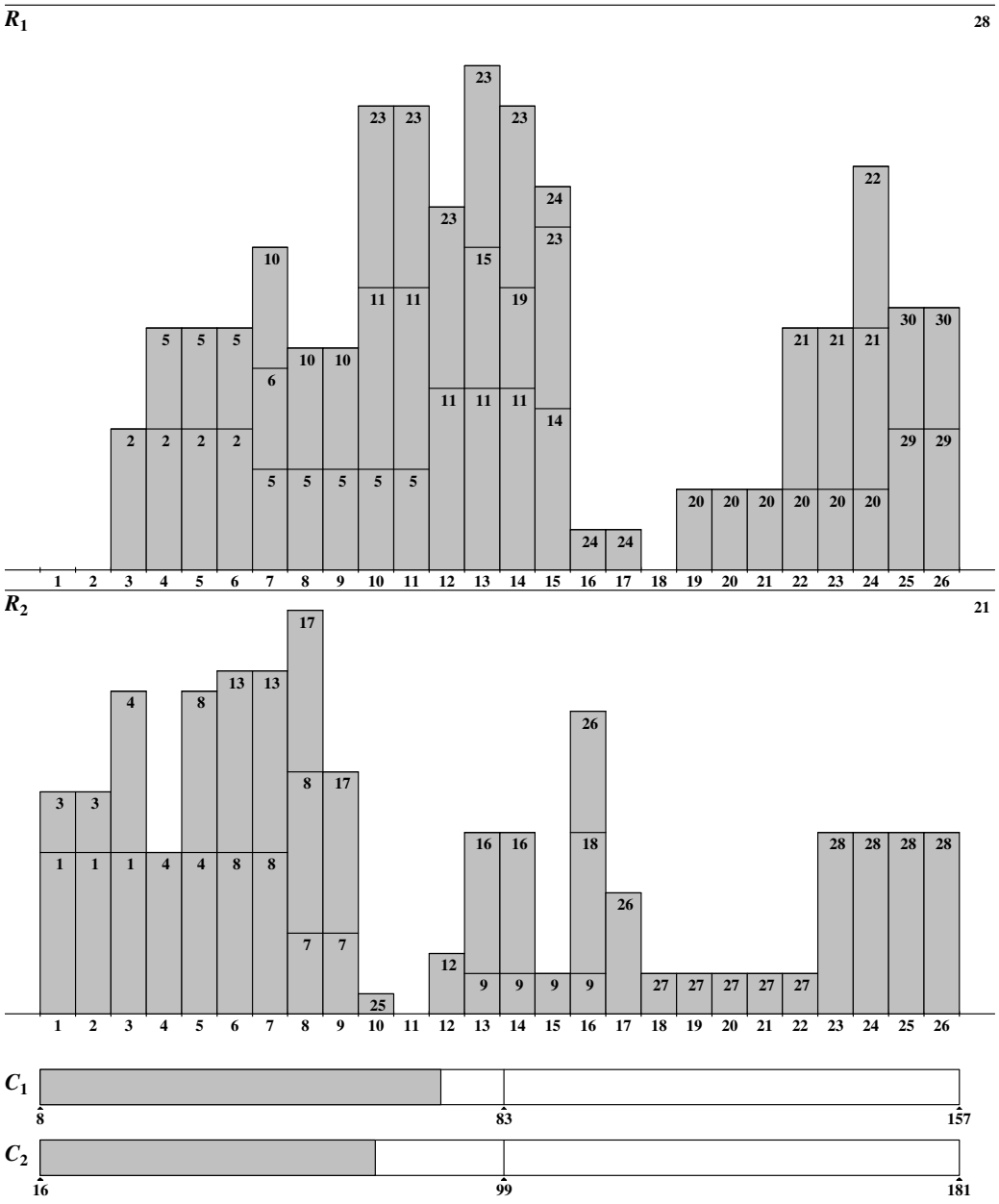


17. ábra A legkorábbi ütemezés erőforrás profilja a J30MM-10-1 esetre

A következő *18.* és *19. ábrákon* a „legjobb” optimális ütemezés és erőforrás felhasználás látható az erőforrás kiegyenlítés előtt. A „legjobb” ütemezés alatt természetesen az elsőbbségi feltételeknek megfelelő és az erőforráskorlátokat kielégítő minimális időtartamú ütemezést értjük.



18. ábra A „legjobb” ütemezés a kiegyenlítés előtt



19. ábra A legjobb ütemezés erőforrásprofilja a kiegyenlítés előtt.

Az erőforrás kiegyenlítés hatását a *14. táblázatban* mutatjuk be. A táblázat az erőforrásegységek újraindítási eseményeinek számát tartalmazza a kiegyenlítés előtt és után.

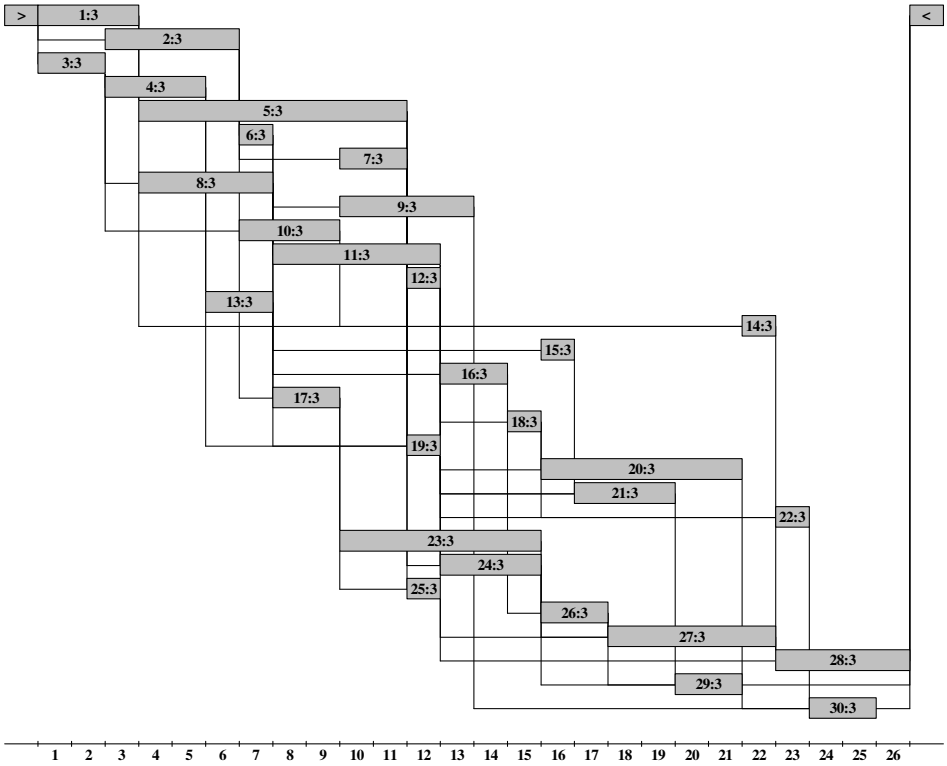
14. táblázat

Az újraindítási események száma az erőforrás kiegyenlítés tükrében

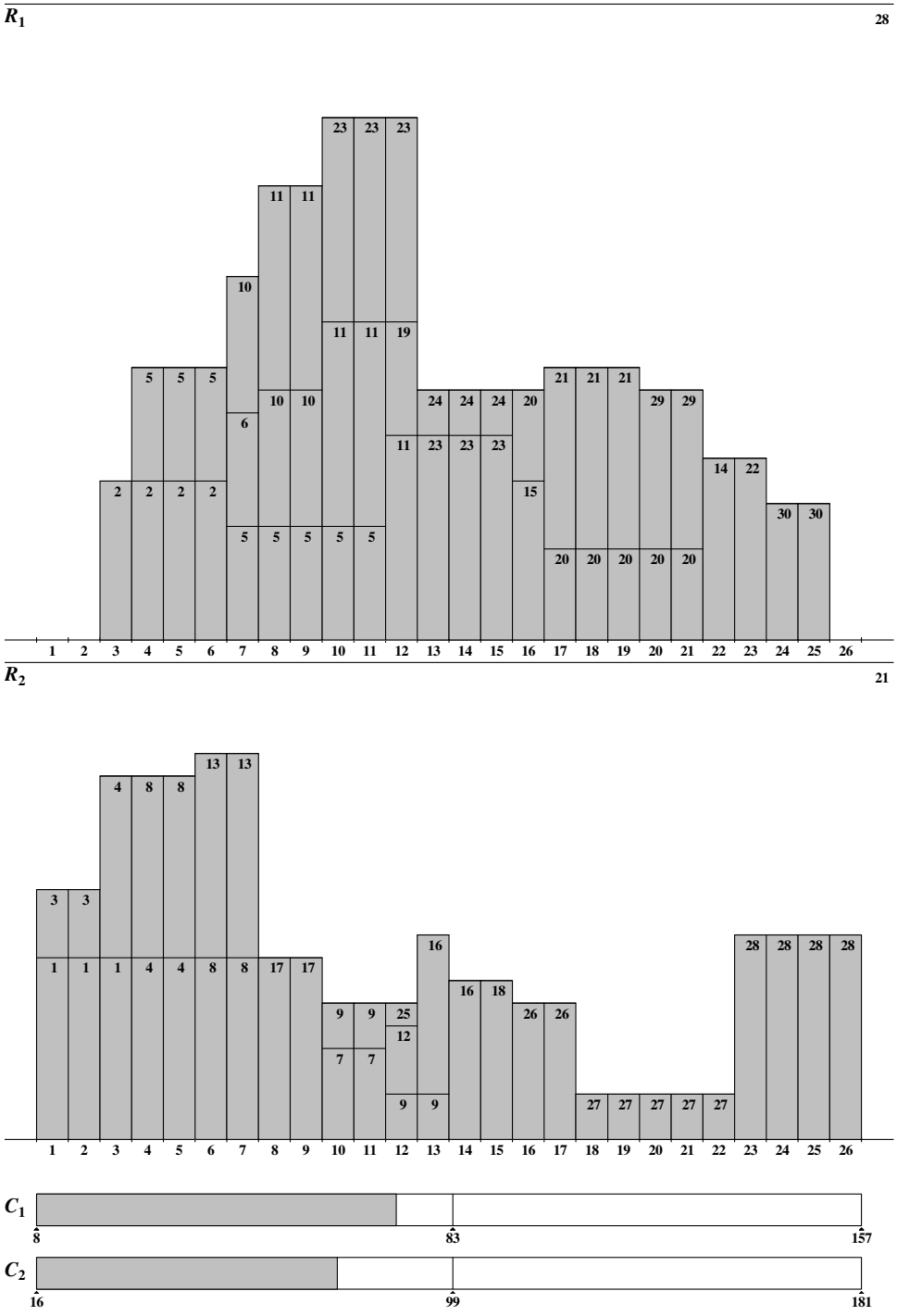
Erőforrás	Kiegyenlítés előtt	Kiegyenlítés után
1	55	24
2	57	27
	112	**51**

A táblázatból kitűnik, hogy a simító eljárás több mint 50%-os csökkenést eredményezett az induló-újrainduló erőforrásegységek számában.

A *20.-21. ábrán* a legjobb ütemezést és a legjobb ütemezés erőforrásprofilját mutatjuk be a kiegyenlítés után.



20. ábra: Legjobb ütemezés a kiegyenlítés után.



21. ábra A legjobb ütemezés erőforrásprofilja a kiegyenlítés után.

Összegzés

Disszertációnkban egy új harmóniakereső metaherusztikát ismertettünk, amely a több-megvalósítási módú projektütemezési problémák esetén képes optimális, minimális erőforrás korlátos ütemezést adni.

Először részletes ismertetést adtunk a projektütemezési problémák alapjairól, a projektek osztályozási szempontjairól. A további fejezetekben a nemzetközi szakirodalom eredményeit, a több-megvalósítási módú erőforrás-korlátos projektek optimalizáló módszereit, egzakt és heurisztikus megoldásait ismertettük. Áttekintettük az erőforrás kiegyenlítési problémák témakörében készült tanulmányokat.

Kiemelten foglalkoztunk a heurisztikus, illetve metaheurisztikus megoldásokkal, mivel a problémakör természetéből adódóan a heurisztikus megoldások bizonyulnak hatékonynak. Bemutattuk a tématerület egyik leghatékonyabb metaheurisztikáját a harmóniakereső metaheurisztikát.

A dolgozat további fejezeteiben részletesen ismertettük az új harmóniakereső algoritmusunkat, amely a több-megvalósítási módú erőforrás-korlátos projektek időtartamát minimalizálja megújuló és nem-megújuló erőforrások felhasználásával.

A hibrid algoritmusunk az egy-megvalósítási módú erőforrás korlátos projektek időtartamát minimalizáló, Csébfalvi által kifejlesztett Sounds of Silence algoritmus továbbfejlesztése, bővítése. Munkánkkal bebizonyítottuk, hogy a harmóniakereső heurisztika Sounds of Silence algoritmusa megfelelő továbbfejlesztéssel, adaptálható a több-megvalósítási módú projektek esetére.

Az előoptimalizáló eljárással lényeges minőségi javulást értünk el. Az előoptimalizáló eljárás egy több-korlátos 0-1 hátizsák problémát old meg. Az előoptimalizálással egy jó kiinduló megoldást sikerül előállítani, amely kizárja azokat a megvalósítási módokat, amelyek esetében a nem-megújuló erőforrásokra vonatkozó korlátok nem teljesülnének. Az előoptimalizálás nem csak az algoritmus sebességében eredményezett lényeges javulást, hanem a megoldás minőségében is.

Alkalmassá tettük az algoritmust, a forward-backward improvement, a head-tail eljárásokkal arra, hogy még pontosabb és gyorsabb eredményt adjon. Részletes, reprodukálható számítási eredményeket közöltünk a nemzetközileg, a kutatók körében ismert és elfogadott teszthalmazon. Ezek az eredmények mind azt mutatták, hogy algoritmusunk hatékony, robosztus algoritmus.

Részletesen ismertettük modellünket, az algoritmus fő lépéseit, s megadtuk a legfontosabb eljárások struktogramját.

A konfliktusjavító modell adaptálásával alkalmassá tettük az algoritmust arra, hogy másodlagos kritériumokat is megfogalmazhassunk célfüggvényként. Másodlagos célként az erőforrás kiegyenlítési és hozzárendelési probléma megoldásának képességével növeltük az algoritmus tudását. (Meg kell jegyezni, hogy más típusú másodlagos kritérium is megadható az optimalizálás céljaként, a másodlagos kritérium rugalmasan változhat.) Ezzel a gyakorlathoz még közelebb álló probléma megoldására is alkalmassá vált hibrid algoritmusunk. A továbbfejlesztett metaheurisztika hatékonyságának szemléltetésére részletes számítási eredményeket mutattunk be a PSPLIB tesztkönyvtár J30MM részhalmazán végzett futtatásokról. Jelenleg ez a legnehezebb teszthalmaz, amely a kutatók számára rendelkezésre áll. A számítási eredmények ismertetésekor

említett Gentben folyó kutatások új kihívásokat jelentenek, amelyek egyben további kutatási irányt is megszabnak számunkra. A valódi kihívást jelentő, 50 és 100 tevékenységet tartalmazó halmazok esetén is vizsgálhatjuk az algoritmusunk hatékonyságát, versenyképességét. A kapott, versenyképes eredményeinket egy, a közeljövőben, Dubrovnikban sorra kerülő konferencián kívánjuk bemutatni. Sikerült felvenni a kapcsolatot Vanhoucke és Peteghen professzorokkal. A teszthalmaz kapcsán közös kutatás, együttműködés lehetőségei körvonalazódnak. Az EJOR folyóiratban a szerzők cikke referálás alatt áll, s eredményeik további vizsgálatainkhoz referencia pontként használhatók.

A témakör lényegéből fakad, hogy további kutatási célok is megfogalmazhatóak. Egy másik továbbfejlesztési irány lehet a több-megvalósítási módú erőforrás-korlátos projektütemezés időtartamának minimalizálása, valamint az erőforrásprofil kisimítása mellett a nettó jelenérték maximalizálási kritérium beépítése is a modellbe, ami már harmadik optimalizálási kritériumként jelenik meg. A probléma jelentőségét jelzi, hogy ennek a témakörnek is folyamatosan nő az irodalma.

Irodalomjegyzék

- Ahuja, H.N., (1976) Construction Performance Control by Networks. Wiley, New York.
- Akkan, C., Drexl, A., Kimms, A., (2005) Network decomposition-based benchmark results for the discrete time-cost tradeoff problem, *European Journal of Operational Research* 165, 339–358.
- Alcaraz, J., Maroto, C., Ruiz R., (2003). Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms, *Journal of theOperational Research Society* 54, 614–626.
- Al-Fawzan, M., Haouari, M. (2005) A bi-objective model for robust resourceconstrained project scheduling, *International Journal of Production Economics* 96 175–187.
- Alvarez-Valdés R.; Tamarit, J.M , (1989) Heuristic algorithms for Resource-Constrained Project Scheduling: a review and an empirical analysis, *Slowinski R. és Weglarz J., Eds., Advances in Project Scheduling*, p. 113–134, Elsevier
- Anagnostopoulos K.P., Koulinas G.K., (2010) A simulated annealing hyperheuristic for construction resource leveling, *Construction Management and Economics,* 28, 163-175.
- Artigues, C., (2005.) The resource-constrained project scheduling problem. In: Artigues, C., Demassey, S., Neron, E. (Eds.), Resource-constrained project scheduling: models, algorithms, extensions and applications. ISTE-Wiley, London, pp. 21–35.
- Ballestin F., Valls V., Quintanilla S., (2008), Pre-emption in resource-constrained project scheduling, *European Journal of Operation Research* 189 (3) 1136-1152.

- Bandelloni, M., Tucci, M., Rinaldi, R., (1994) Optimal resource leveling using non-serial dynamic programming. *European Journal of Operational Research* 78, 162-177.
- Bartels, J.-H., Zimmermann, J., (2009) Scheduling tests in automotive R&D projects, *European Journal of Operational Research* 193 (3) 805–819.
- Bellenguez, O., E. Neron, E., (2005) Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills, *Lecture Notes in Computer Science* 3616 229–243.
- Bianco, L., P. Dell'Olmo, P., Speranza, M.G., (1998) Heuristics for multimode scheduling problems with dedicated resources, *European Journal of Operational Research* 107 260–271.
- Böttcher J., Drexl A., Kolisch R., Salewski F., (1999) Project scheduling under partially renewable resource constraints. *Management Science* 45 (4), 543-559.
- Bouleimen K., Lecocq H., (2003) A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, *European Journal of Operational Research*, 149, 268-281, 2003.
- Boctor, F.F., (1993.) Heuristics for scheduling projects with resource restrictions and several resource-duration modes. *International Journal of Production Research* 31 (11), 2547–2558.
- Boctor, F.F., (1996a). A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. *European Journal of Operational Research* 90 (2), 349–361.
- Boctor, F.F., (1996b). Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research* 34 (8), 2335–2351.
- de Boer, R., Resource-constrained multi-project management, a hierarchical decision support system. Ph.D. Thesis, University of Twente, 1998.

- Bomsdorf, F., Derigs, U., (2008) A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem, *OR Spectrum* 30 (4) 751–772.
- Brinkman, K., Neumann, K., (1996.) Heuristic procedures for resource-constrained project scheduling with minimal and maximal time lags: the minimum project duration and resource leveling problem. *Journal of Decision Systems* 5, 129–156.
- Brucker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E., (1999) Resource-constrained project scheduling: notation, classification, models and methods. *European Journal of Operational Research* 112 (1), pp 3-41.
- Brucker P., Kunst S., (2001) Resource-constrained project scheduling and timetabling, Lecture Notes in Computer Science 2079, 277-293
- Brucker, P., Knust, S., (2003). Lower bounds for resource-constrained project scheduling problems. *European Journal of Operational Research* 149 (2), 302–313.
- Burgess, A.R., Killebrew, J.B., (1962.) Variation in activity level on a cyclic arrow diagram. *Journal of Industrial Engineering* 13 (2), 76–83.
- Csébfalvi G., Csébfalvi A., Szendrői E., (2008a), A harmony search metaheuristic for the resource-constrained project scheduling problem and its multi-mode version, in *"Project Management and Scheduling 2008"*, F. S. Serifoglu, Ü. Bilge (Editors), Istanbul, Turkey, 2008, pp 56-59.
- Csébfalvi G., Eliezer O., Láng B., Levi R.,(2008b) "A conflict repairing harmony search metaheuristic and its application for bi-objective resource-constrained project scheduling problems", in *Project Management and Scheduling 2008*, F. S. Serifoglu, Ü. Bilge, (Editors), Istanbul, Turkey, 60-63, 2008
- Csébfalvi G., Konstantinidis P. (1998), "A new exact resource balancing procedure for the multiple resource-constrained project scheduling problem", in *Proceedings of APMOD '98*, Limasol, Cyprus, 11-13 March, 1998

- Csébfalvi, G., Csébfalvi,A., (2008) A new metaheuristic for the multidimensional 0-1 knapsack problem, in "*Computational Management Science Conference 2008*", Imperial College, London, UK, 38-39, 2008
- Chiang, C.W., Huang, Y.Q., Wang, W.Y., 2008. Ant colony optimization with parameter adaptation for multi-mode resource-constrained project scheduling. *Journal of Intelligent and Fuzzy Systems* 29 (4–5), 345–358.
- Chyu, C.C., Chen, A.H.L., Lin, X.H., 2005. A hybrid ant colony approach to multi-mode resource-constrained project scheduling problems with nonrenewable types. In: Proceedings of First International Conference on Operations and Supply Chain Management, Bali, 2005.
- Davis, K.R., Stam, A., Grzybowski, R.A., (1992) Resource constrained project scheduling with multiple objectives: A decision support approach, *Computers and Operations Research* 19 (7) 657–669.
- Debels D., Vanhoucke M., (2008), The impact of various activity assumptions ont he lead time and resource utilization of resource-constrained projects, *Computers and Industrial Engineering* 54, 140-154.
- Demeulemeester, E.L., Herroelen, W.S., (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science* 38 (12), 1803–1818.
- Demeulemeester E.L., Herroelen W.S., (1996), An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem. *European Journal of Operational Research* 90 (2) 334-348.
- Demeulemeester, E.L., de Reyck, B., Foubert, B., Herroelen, W.S., Vanhoucke, M., (1998) New computational results on the discrete time/cost trade-off problem in project networks, *Journal of the Operational Research Society* 49 614– 626.
- De Reyck, B., Demeulemeester, E., Herroelen, W., 1998. Local search methods for the discrete time/resource trade-off problem in project networks. Naval Research Logistic Quarterly 45 (6), 553–578.

- Demeulemeester, E.L., Herroelen, W.S., (2002). Project Scheduling – A Research Handbook. Kluwer Academic Publishers, Boston.
- Drexl, A., Grunewald, J., (1993). Nonpreemptive multi-mode resource-constrained project scheduling. IIE Transactions 25 (5), 74–81.
- Drexl, A., Nissen, R., Patterson, J.H., Salewski, F., (2000). Progen/px – An instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions, *European Journal of Operational Research* 125 (1) 59–72.
- Drexl, A., Kimms, A., (2001) Optimization guided lower and upper bounds for the resource investment problem, *Journal of the Operational Research Society* 52 (2001) 340–351.
- Drezet L.E., Billaut J.C., (2008), A project scheduling problem with labour constraints and time-dependent activities requirements, *European Journal of Operational Research* 112 (1) 217-225.
- Duin, C.W., Van Der Sluis, E., (2006). On the complexity of adjacent resource scheduling. *Journal of Scheduling* 9 (1), 49–62.
- Easa, S.M., (1989) Resource leveling in construction by optimization. *Journal of Construction Engineering and Management* 115, 302-316.
- Elmaghraby, S.E., (1977) Activity networks: Project planning and control by network models, Wiley, New York, 1977.
- Erenguc, S.S., Tufekci, S., Zappe, C.J., (1993). Solving time/cost trade-off problems with discounted cash flows using generalized Benders decomposition. Naval Research Logistics 40 (1), 25–50.
- Geng, J., Weng, L., Liu, S., (2011) An improved ant colony optimization algorithm for nonlinear resource leveling problems, *Computers and Mathematics with Applications*, 61, 2300-2305
- Harris, R.B., (1990) Packing method for resource leveling (pack). *Journal of Construction Engineering and Management* 116, 39-43.

- Hartmann S., (1999), Project Scheduling under limited resources: Models, methods and applications, Number 478 in Lecture Notes in Economics and Mathematical Systems, Springer, Berlin, Germany.
- Hartmann, S., (2001) Project scheduling with multiple modes: A genetic algorithm, *Annals of Operations Research* 102 111–135.
- Hartmann, S., Briskorn, D., (2010). A survey of variants and extensions of the resource –constrained project scheduling problem, *European Journal of Operational Research* 207 1-14.
- Hartmann, S., Drexl, A., (1998). Project scheduling with multiple modes: a comparison of exact algorithms. *Networks* 32 (4), 283–297.
- Hartmann, S., Sprecher, A., (1996). A note on ‘‘Hierarchical models for multi-project planning and scheduling”. *European Journal of Operational Research* 94 (2), 377–383.
- Herroelen, W., De Reyck, B., (1999) Phase transition in project scheduling, *Journal of the Operational Research Society,* 50, 148-156.
- Herroelen, W.,(2006) Project scheduling–theory and practice, *Production and Operations Management*, vol., 14, num. 4, p. 413–432.
- Icmeli, O., Erenguc, S.S., (1996). The resource constrained time/cost trade-off project scheduling problem with discounted cash flows. *Journal of Operations Management* 14 (3), 255–275.
- Jarboui B., Damak N., Siarry P., Rebai A., (2008) A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems, *Applied Mathematics and Computation*, 195,299-308,2008
- Jozefowska, J., Mika, M., Rozycki, R., Waligora, G., Weglarz, J., (2001) Simulated annealing for multi-mode resource-constrained project scheduling, *Annals of Operations Research* 102 137–155.
- Kolisch, R., (1995.) Project Scheduling under Resource Constraints: Efficient Heuristics for Several Problem Classes. Physisca-Verlag, Heidelberg.

- Kolisch, R., (1996). Serial and parallel resource-constrained project scheduling methods revisited: theory and computation. *European Journal of Operational Research* 90 (2), 320–333.
- Kolisch, R., Drexl, A., (1997). Local search for nonpreemptive multi-mode resourceconstrained project scheduling. IIE Transactions 29 (11), 987–999.
- Kolisch, R., Hartmann, S., 1999. Heuristic algorithms for the resource-constrained project scheduling problem: classification and computational analysis. In: Weglarz, J. (Ed.), Project Scheduling: Recent Models, Algorithms, and Applications. Kluwer Academic Publishers, Boston, pp. 147–178.
- Kolisch R., Sprecher A.,(1996) "PSPLIB – a project scheduling library," in *European Journal of Operational Research*, vol. 96, pp. 205-216.
- Kolisch, R., Sprecher, A., Drexl, A.,(1995). Characterization and Generation of a general class of resource-constrained project scheduling problems. *Management Science* 41 (10), 1693–1703.
- Kolisch, R., Padman R., (2001) An integrated survey of deterministic project scheduling, *Omega* 29 249–272.
- Lancaster, J., Ozbayrak, M. (2007) Evolutionary algorithms to project scheduling problems – a survey of the state-of-the-art. *International Journal of Production Research*, Vol. 45, No. 2. 425-450.
- Láng Blanka (2010) A nettó jelenérték maximalizálása az erőforrás-korlátos projektütemezés során - egy új harmónia kereső heurisztika, Phd dolgozat, PTE-KTK Gazdálkodástani Doktori Iskola
- Lee K. S., Geem Z. W.(2005), "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice" in *Computer Methods in Applied Mechanics and Engineering*, vol. 194. 2005, pp. 3902-3933.

- Li, H., Womer, K., 2008. Modeling the supply chain configuration problem with resource constraints. *International Journal of Project Management* 26 (6), 646–654.
- Mahdavi, M., Fesanghary, M., Damangir, E., (2007) An improved harmony search algorithm for solving optimization problems, *Applied Mathematics and Computation*, 188 pp 1567-1579.
- Mika M., Waligóra G., Węglarz, J.,(2006), Modelling setup times in project scheduling. In : Józefowska J., Węglarz, J. (Eds.), Perspectives in Modern Project Scheduling. Springer, Berlin, pp. 131-163.
- Mika M., Waligóra G., Węglarz, J.,(2008) Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. European Journal of Operational Research, 187 (3), 1238-1250.
- Möhring, R.H., (1984). Minimizing costs of resource requirements in project networks subject to a fixed completion time. *Operations Research* 32 (1), 89–120.
- Nabrzynski, J., Weglarz, J., (1999) Knowledge-based multiobjective project scheduling problems, in: Weglarz (1999) pp. 383–411.
- Neron, E., (2002). Lower bounds for the multi-skill project scheduling problem. In: Proceedings of the Eighth International Workshop on Project Management and Scheduling (PMS2002), Valencia, Spain, pp. 274–277.
- Neumann, K., Schwindt, C., (2002) Project scheduling with inventory constraints, *Mathematical Methods of Operations Research* 56 513–533.
- Neumann, K., Schwindt, C., Zimmermann, J., (2002). Project Scheduling with Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions. *Springer*, Berlin.
- Neumann, K. Zimmermann J., (1999a) Methods for resource-constrained project scheduling problem with regular and nonregular objective functions and schedule-dependent time windows, in: Weglarz Ed. Project Scheduling:

Recent Models, Algorithms and Applications, Kluwer Academic Publishers, pp. 261–288.

- Neumann, K., Zimmerman, J., (1999b) Resource leveling for projects with chedule-dependent time windows, *European Journal of Operational Research* 117 591-605
- Nonobe, K., Ibaraki, T. (2002), Formulation and tabu search algorithm for the resourceconstrained project scheduling problem, in: C.C. Ribeiro, P. Hansen (Eds.), Essays and Surveys in Metaheuristics, Kluwer Academic Publishers., pp.557–588.
- Nudtasomboon, N., Randhawa, S.U., (1997) Resource-constrained project scheduling with renewable and non-renewable resources and time-resource tradeoffs, *Computers and Industrial Engineering* 32 (1) 227–242.
- Özdamar, L. (1999) A genetic algorithm approach to a general category project scheduling problem, *IEEE Transactions on Systems, Man, and Cybernetics,* Part C: Applications and Reviews 29 44–59.
- Özdamar, L., Ulusoy, G.,(1994). A local constraint based analysis approach to project scheduling under general resource constraints. *European Journal of Operational Research* 79 (2), 287–298.
- Patterson, J. H, (1984). A Comparison of Exact Procedures for Solving the Multiple Constrained Resource Project Scheduling Problem. *Management Science* 30, 7, 854-867.
- Patterson, J.H., Slowinski, R., Talbot, F.B., Weglarz, J., (1989). An algorithm for a general class of precedence and resource constrained scheduling problems. In: Slowinski, R., Weglarz, J. (Eds.), Advances in Project Scheduling. Elsevier Amsterdam, pp. 3–28.
- Patterson, J.H., Talbot, F.B., Słowinski, R., Weglarz, J., (1990). Computational experience with a backtracking algorithm for solving a general class of precedence and resource-constrained scheduling problems. *European Journal of Operational Research* 49 (1), 68–79.

- Pesch, E., (1999) Lower bounds in different problem classes of project schedules with resource constraints, in: Weglarz(1999), pp. 53–76.
- Pritsker, A.A.B., Watters, L.J., Wolfe, P.M., (1969.) Multiproject scheduling with limited resources: a zero-one programming approach. *Management Science* 16 (1), 93–107.
- Raidl, G.R., (1998) An improved genetic algorithm for the multiconstrained 0-1 knapsack problem. In. D.B. Fogel, editor, Proceeding of the 1998 IEEE Internationale Conference on Evolutionary Computation, pp 207-211, IEEE Press
- Słowinski, R., (1980). Two approaches to problems of resource allocation among project activities – a comparative study. *Journal of the Operational Research Society* 31 (8), 711–723.
- Slowinski, R., Soniewiczki, B., Weglarz J., (1994). DSS for Multiobjective Project Scheduling, *European Journal of Operational Research*, 79, 220-229.
- Schwindt, C., (1995) ProGen/max: A new problem generator for different resource-constrained project scheduling problems with minimal and maximal time lags. Research Report, University of Karlsruhe.
- Schwindt, C., Trautmann, N., (2003). Scheduling the production of rolling ingots: industrial context, model, and solution method. *International Transactions in Operational Research* 10 (6), 547–563.
- Speranza, M.G., Vercellis, C., (1993). Hierarchical models for multi-project planning and scheduling. *European Journal of Operational Research* 64 (2), 312–325.
- Sprecher, A., (1994) Resource-constrained project scheduling: Exact methods for the multi-mode case, Number 409 in Lecture Notes in Economics and Mathematical Systems, Springer, Berlin, Germany, 1994.
- Sprecher, A., Drexl, A., (1998). Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research* 107 (2), 431–450.

- Sprecher, A., Hartmann, S., Drexl, A., (1997). An exact algorithm for project scheduling with multiple modes. *OR Spektrum* 19 (3), 195–203.
- Szendrői E., (2009) "A Hybrid Method for the Multi-Mode Resource-Constrained Project Scheduling Problem", in Proceedings of the First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering, B.H.V. Topping, Y. Tsompanakis, (Editors), Civil-Comp Press, Stirlingshire, United Kingdom, paper 3, 2009. doi:10.4203/ccp.92.3
- Szendrői, E., (2010a) A robust hybrid method for the multimode resource-constrained project scheduling problem, *Pollack Periodica*, Vol 5, No. 3, Akadémiai Kiadó
- Szendrői, E., (2010b)A Hybrid Method for the Multi-Mode Resource-Constrained Project Scheduling Problem with Strip Packing like Resource Constraints, in Proceeding of the Seventh International Conference on Engineering Computational Technology, B.H.V. Topping, , J.M. Adam, F. J. Pallarés, R. Bru and M.L.Romero (Editors) Civil-Comp Press, Stirlingshire, United Kingdom paper 93, 2010
- Szendrői, E., (2010c) Egy hibrid eljárás a több megvalósítási módú erőforrás-korlátos projektütemezési probléma megoldására, *SZIGMA*, 2010. XLI. Évfolyam, 3-4. szám, 177-194
- Talbot, F.B., (1982). Resource-constrained project scheduling with time-resource trade-offs: the nonpreemptive case. *Management Science* 28 (10), 1197–1210.
- Tiwari, V., Patterson, J.H., Mabert, V.A., (2009) Scheduling projects with heterogeneous resources to meet time and quality objectives, *European Journal of Operational Research* 193 (3) 780–790.
- Tormos P., Lova A. (2001), A competitive heuristic solution technique for resource-constrained project scheduling, *Annals of Operations Research*, 102, 65–81

- Van Peteghem, V., Vanhoucke, M., (2010.) A genetic algorithm for the preemptive and nonpreemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research* 201 (2), 409–418.
- Vanhoucke, M., Demeulemeester, E.L., Herroelen, W.S., (2001) An exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem, *Annals of Operations Research* 102 179–196.
- Viana, A., de Sousa, J.P., (2000), Using metaheuristics in multiobjective resource constrained project scheduling, *European Journal of Operational Research* 120 (2) 359–374.
- Voß, S., Witt, A., (2007) Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application, *International Journal of Production Economics* 105 (2) 445–458.
- Weglarz, J., (1980). On certain models of resources allocation problems. *Kybernetes* 9 (1), 61–66.
- Weglarz, J., (1981). Project scheduling with continuously-divisible, doubly constrained resources. *Management Science* 27 (9), 1040–1052.
- Weglarz J., (1999) (Ed.), Project Scheduling: Recent Models, Algorithms and Application, Kluwer Academic Publishers
- Weglarz, J., Józefowska J., Mika M., Waligóra G., (2011) Project scheduling with finite or infinite number of activity processing modes –A survey. *European Journal of Operational Research* 208 177-205.
- Zhang, H., Tam, C.M., Li, H., (2006.) Multi-mode project scheduling based on particle swarm optimization. *Computer-Aided Civil and Infrastructure Engineering* 21 (2), 93–103.
- Zhu, G., Bard, J.F., Yu, G., (2006) A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem, *INFORMS Journal on Computing* 18 (2006) 377–390.