EGY HIBRID ELJÁRÁS A TÖBB MEGVALÓSÍTÁSI MÓDÚ ERŐFORRÁS-KORLÁTOS PROJEKTÜTEMEZESI PROBLÉMA MEGOLDÁSÁRA[1]

SZENDRŐI ETELKA
PTE Pollack Mihály Műszaki Kar

A tanulmány egy hibrid algoritmust mutat be a több megvalósítási módú erőforrás-korlátos projektek ütemezési problémájának megoldására. Az ismertetett megközelítésben egy harmóniakereső algoritmust kombináltunk egy új és hatásos ,,head-tail" lokális kereső eljárással, amely egy vegyes egészértékű lineáris programozási modellen (MILP) alapszik. Az új megközelítés lényegének és életképességének igazolására a jól ismert PSPLIB tesztkönyvtár J30MM halmazára vonatkozó számítási eredményeket tesszük közzé. A ,,head-tail" javítás generálására egy jól ismert MILP megoldó szoftvert (CPLEX) alkalmaztunk.

Kulcsszavak: több megvalósítási módú erőforrás-korlátos projektütemezés, heurisztikus és metaheurisztikus technikák, harmóniakereső optimalizálás, hibrid eljárások, projektmenedzselés, számítási eredmények

1 Bevezetés

A tanulmány egy hibrid algoritmust ismertet a több megvalósítási módú erőforrás-korlátos projektütemezési probléma megoldására (MRCPSP, Multimode Resource-Constrained Project Scheduling Problem). Az algoritmus a ,,Sounds of Silence" (,,Csend hangjai") harmóniakereső metaheurisztikán alapul, amelyet Csébfalvi [7] fejlesztett ki az erőforrás-korlátos ütemezési feladatok megoldására és Csébfalvi at al. [5] fejlesztette tovább az MRCPSP feladatok esetére. Magát a harmóniakereső (HS) metaheurisztikát eredetileg Lee és Geem [16] dolgozta ki a zenei improvizációs folyamatok analógiájaként, ahol a zenészek improvizálásának célja a minél tökéletesebb harmónia elérése. A bemutatott továbbfejlesztett algoritmus az eredeti ,,Sounds of Silence" harmóniakereső metaheurisztika és egy új, hatékony, a MILP formulán alapuló ,,head-tail" lokális kereső eljárás kombinációja. A módosított modell váza a Szendrői [23] tanulmányban került ismertetésre, részletes számítási, futtatási eredmények nélkül. Jelen tanulmány mélyebben ismerteti az algoritmust és részletes futtatási eredményekkel támasztja alá annak hatékonyságát. A módosítás lényege, hogy az eredetileg teljesen véletlenszerű kezdő repertoárt helyettesítjük egy előoptimalizált melódiahalmazzal, amely a relaxált megoldásból véletlen perturbációval keletkezik. A ,,head-tail" eljárás megpróbálja csökkenteni a projekt teljes időszükségletét azoknak az erőforrásoknak

[1]Beérkezett: 2010. július 11. E-mail: szendroi@pmmk.pte.hu.

az újraelosztásával, amelyeket a HS eljárás a kezdő és befejező tevékenységekhez rendelt. A lokális kereső eljárás kihasználja azt a tényt, hogy egy gyors és hatékony megoldó szoftver használata a kisméretű MILP problémákra elfogadható időn belül képes megoldást adni. Az ajánlott metaheurisztika hatékonyságának és életképességének bizonyítására, közöljük a számítási eredményeket, amelyeket a jól ismert és népszerű PSPLIB tesztkönyvtár J30MM részhalmazán végzett futtatások [14] során kaptunk. A ,,head-tail" megoldás generálásához egy korszerű MILP szoftvert (CPLEX 8.1) alkalmaztunk.

Az erőforrás-korlátos több megvalósítási módú projektütemezési problémák (MRCPSP) irodalma rendkívül szerteágazó, számos megközelítést, megoldási módszert dolgoztak ki a probléma megoldására. Jelen dolgozat terjedelmi korlátai nem teszik lehetővé, hogy a teljes irodalmat részletesen áttekintse, csak néhány jelentős munka megemlítésére vállalkozhat. Az MRCPSP problémák célja a projekt időtartamának (makespan) minimalizálása, az erőforráskorlátok és a több megvalósítási mód figyelembevételével. A több megvalósítási módú projektekben a tevékenységek erőforrás szükséglete és végrehajtásának időtartama függ a megvalósítási mód megválasztásától. Kisméretű problémák megoldására jól használhatók az optimális megoldást adó egzakt módszerek, míg nagyméretű problémák esetén a gyakorlatban heurisztikus módszereket és metaheurisztikákat alkalmaznak.

Az egzakt eljárásokat kidolgozó kutatók közül Talbot [24] volt az első, aki egy leszámlálási(enumeration) sémát publikált a feladat megoldására. Patterson at al. [20] egy leszámoláson alapuló branch and bound algoritmust tett közzé, amelyben egy keresési fa generálásával jutnak el az optimális megoldáshoz. Sprecher et al. [22], Hartmann és Drexl [11] valamint Sprecher és Drexl [21] branch and bound algoritmusokat közöltek, melyben különböző elsőbbségi szabályokat alkalmaznak a keresési fa metszéséhez. Demeulemeester [8] egy mélységi keresést kombinál a branch and bound eljárással.

Heurisztikus eljárásokat publikált Drexl és Grünewald [9], Özdamar és Ulusoy [19], Boctor [2,3], Kolish és Drexl [15]. Drexl és Grünewald egy sztochasztikus ütemezési modellt tett közzé, míg Özdamar és Ulusoy egy lokális korlátokon alapuló elemzési módszert alkalmaz. Boctor első munkájában prioritási szabályokon alapuló heurisztikákat javasol, míg a második munkájában egy kritikus út módszeren alapuló heurisztikus algoritmust publikál. Kolish és Drexl egy lokális kereső heurisztikát mutat be, amely három fázisból áll.

A metaheurisztikus eljárásokat közlő munkák közül evolúciós algoritmusokat alkalmaz Özdamar [18], Hartmann [10], Alcaraz et al. [1], Lova at al. [17]. Józefowska et al. [13], Bouleimen és Lecocq [4] a szimulált hűtés elvét alkalmazza. Zhang et al. [26] a részecske rajzás (particle swarm) optimalizáló eljárást használja az MRCPSP probléma megoldására.

N	A valóságos tevékenységek száma
M	A tevékenységek megvalósítási módjainak száma
m	Megvalósítási módok indexe $m \in \{1, 2, \ldots, M\}$
i	az i-edik tevékenység indexe $i \in \{1, 2, \ldots, N\}$
D_{im}	az i-edik tevékenység időtartama az m-edik megvalósítási módban
$i \rightarrow j$	i tevékenység a j tevékenység előzménye
PS	az $i \rightarrow j$ elsőbbségi (megelőző-rákövetkező) kapcsolatok halmaza
R	a megújuló erőforrás típusok száma
C	a nem-megújuló erőforrások száma
R_{imr}	az i-edik tevékenység erőforrás-szükséglete az r-edik megújuló erőforrásból az m-edik megvalósítási módban
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 időtartamának összege)
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.
T	a projekt időperiódusainak száma $t \in \{1, 2, \ldots, T\}$
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 módja
W_{imst}	Bináris változó, értéke 1, ha a tevékenység aktív a t időperiódusban, egyébként 0.

1. táblázat. A modellben használt jelölések listája

2 A modell

A vizsgált modellünkben a következő állításokból indulunk ki: A projekt N darab valós tevékenységből áll, amelyeket 1-től N-ig sorszámozunk. Minden i tevékenység, $i \in \{1, 2, \ldots, N\}$ 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 tevékenységek be nem fejeződnek. Jelölje PS halmaz,

$$PS = \{\, i \rightarrow j \mid i \neq j,\ i \in \{0, 1, \ldots, N\},\ j \in \{1, 2, \ldots, N + 1\} \,\}$$

a tevékenységek közötti elsőbbségi feltételek halmazát, ahol a nyíl szimbólum azt jelzi, hogy a j tevékenység csak az i tevékenység befejezése után kezdődhet el. Az erőforrások két nagy csoportba sorolhatók, a megújuló és a nem-megújuló erőforrások csoportjába. A megújuló erőforrásfajták számát jelölje R, a nem-megújuló erőforrásfajták számát jelölje C. Az i-edik tevékenység végrehajtása, $i \in \{1, 2, \ldots, N\}$ az m-edik megvalósítási módban, $m \in \{1, 2, \ldots, M\}$, D_{im} időegységet igényel. Jelölje R_r az időegység alatt rendelkezésre

álló r-edik megújuló erőforrás korlátját, ahol $r \in \{1, 2, \ldots, R\}$. Legyen C_c a projekt teljes erőforráskorlátja a c-edik, $c \in \{1, 2, \ldots, C\}$ 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:

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

Jelölje X_i az i-edik, $i \in \{1, 2, \ldots, N\}$ tevékenység kezdetét, és legyen $[\underline{X}_{im}, \overline{X}_{im}]$ az az időintervallum, amelyben az i-edik tevékenység az m-edik megvalósítási módban elkezdődhet $m \in \{1, 2, \ldots, M\}$. 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. 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. Jelöljük a modell döntési változóit az X_{ims} bináris változókkal, amelyek értéke 1, ha az i-edik tevékenység az m-edik $m \in \{1, 2, \ldots, M\}$ megvalósítási módban kerül ütemezésre s kezdési idővel, egyébként az értéke legyen 0. A modell a következő formulákkal írható le:

$$\min[X_{N+1}] = X^*_{N+1} \tag{2}$$

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

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

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

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

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

$$W_{imst} = \begin{cases} 1 & \text{ha } s \leq t \;\; \wedge \;\; t < s + D_{im} \\ 0 & \text{ha } t < s \;\; \vee \;\; t \geq s + D_{im} \end{cases} \tag{8}$$

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

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

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

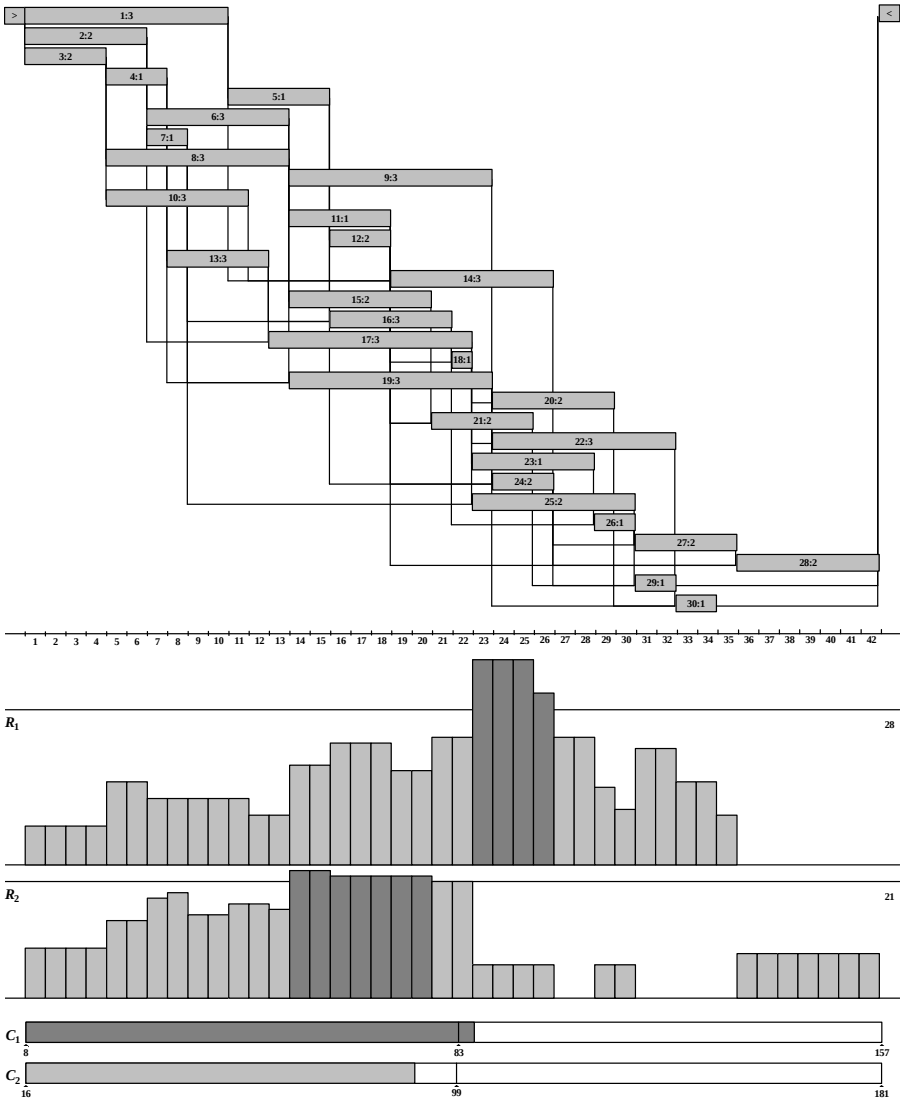
$$X = \{X_1, X_2, \ldots, X_N\} \tag{12}$$

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

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

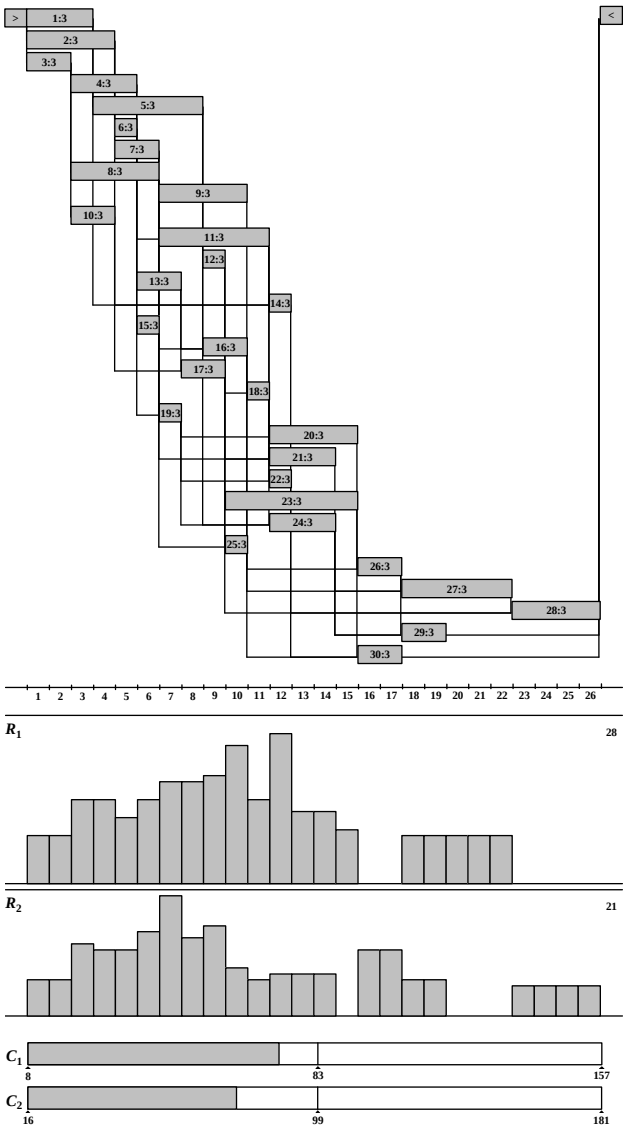
A (4) feltétel biztosítja, hogy minden i-edik tevékenység végrehajtása pontosan egyszer, egyetlen megvalósítási módban és a megvalósítási módjának megfelelő időintervallumban kezdődjék el. A (5-6) feltételek az elsőbbségi (megelőzési-rákövetkezési) relációkat írják le. Az időperiódusokban rendelkezésre 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 [14] hozott létre. Az *1. 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. Az egységnyi látszólagos forrás (nyelő) tevékenységeket a $>$ ($<$) szimbólumok jelzik. A megújuló (nem-megújuló) erőforrás felhasználási hisztogramokban sötétebb szürke szín jelöli azokat az időperiódusokat, amelyekben az erőforrás szükséglet az erőforráskorlátokat meghaladja.



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

A *2. ábra* az első ábrán bemutatott modell egy optimális (minimális időtartamú erőforrás korlátokat kielégítő) ütemezését, szimplex módszerrel kapott egzakt megoldását mutatja. Meg kell jegyeznünk, hogy több optimális megoldás is lehetséges.



2. ábra. A J30MM-10-1 projekt egy optimális ütemezése

3 Az algoritmus

A harmóniakereső (HS) algoritmust Lee és Geem [16] dolgozta ki a zenei improvizáció analógiájára, ahol a zenészek egy jobb harmónia elérésére törekednek. A harmóniakereső eljárásban az optimalizálási feladat a következőképpen

írható le:

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

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

- 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;

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

Az algoritmus egy teljesen véletlenszerű ,,repertoár betöltő" 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. A HS algoritmus egy ,,explicit" algoritmus, mivel közvetlenül a hangokon fejti ki hatását.

A Csébfalvi féle SoS algoritmus és ebben a tanulmányban szereplő algoritmus is implicit módon kezeli a hangokat, 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 világába áttranszformálni. A zene világában az erőforrásprofilok egy ,,többszólamú dallamot" alkotnak. Tegyük fel, hogy minden szólamban csak a ,,magas hangok" hallhatóak, így a transzformált probléma 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:

Zenei analógia	MRCPSP modell
A zenekar N zenészből áll	A projekt N tevékenységből áll
Zenész	Tevékenység
Minden i zenész jellemezhető egy diszjunktív többszólamú hanghalmazzal és a hang lejátszásához szükséges energiával	Minden i tevékenység több megvalósítási móddal és erőforrás igénnyel rendelkezik
Polifonikus dallam	Erőforrásprofil
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

Az MRCPSP esetben minden i zenész, $i \in \{1, 2, \ldots, N\}$ 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 előadás minőségének ellenőrzéséhez 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.

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 *3-4. ábrákon* látható. A 3. ábrán szereplő π_{im} súlyérték, amely annak a valószínűsége, hogy a kiválasztott (x) érték az $m \pm 0.5$ környezetébe esik, a következőképpen definiálható:

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

ahol IS_i a várható érték, σ a szórás. Az eredeti változatban a repertoár feltöltési fázisban IS_i értékét normális eloszlásból generáljuk

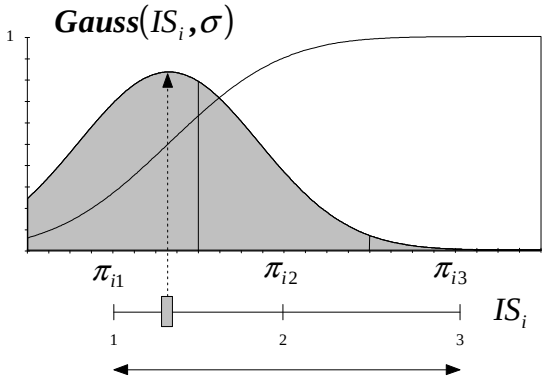
$$IS_i = RandomGauss(\mu, \sigma), 1 \leq IS_i \leq M\ , \quad \mu = 1, \sigma = 1 \text{ paraméterekkel.} \tag{17}$$

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

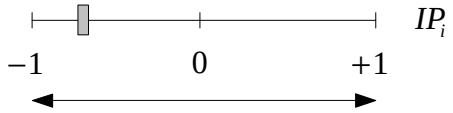
$$IS_i = RandomGauss(\tilde{M}_i, \sigma), \quad 1 \leq IS_i \leq M\ , \tag{18}$$

ahol $\tilde{M}_i$, $1 \leq \tilde{M}_i \leq M$, $i \in \{1, 2, \ldots, N\}$ a relaxált mód érték a relaxált megoldásból. Az előoptimalizáláskor a repertoárt az adott multimódos szituációnak megfelelően állítjuk elő, kikeverjük azokat a valószínűségeket, amelyek diszjunktív dallamokhoz tartozhatnak. A módosított változatban σ az algoritmus egy „bűvös száma" mivel a kezdő repertoár változatosságára nagy hatással van ez az érték. Minden heurisztikának vannak állítható paraméterei,

a mi esetünkben ez a ,,bűvös szám" az állítható paraméter. Az előzetes eredményeinknek megfelelően egy jó beállítás a következő lehet: $0.1 \leq \sigma \leq 0.2$. Kezdetben a választás szabadsága nagy, majd az improvizációk során, az idővel csökken a választás szabadsága, azaz σ értéke csökken.

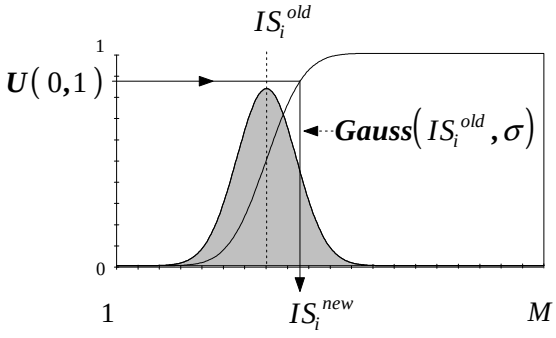


3. ábra. Egy IS_i elképzelés a ,,legjobb" megvalósítási módról ($M = 3$)

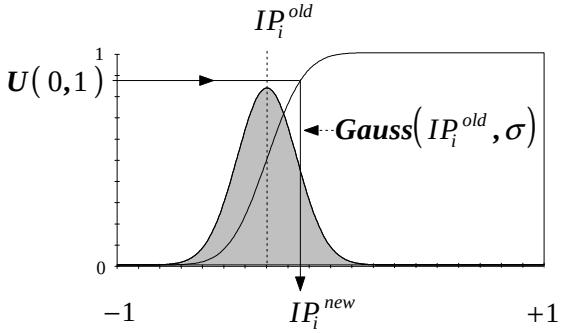


4. ábra. Egy IP_i elképzelés a ,,legjobb" pozícióról

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 folyamatosan csökken (lásd *5-6. ábra*).



5. ábra. IS_i perturbációja



6. ábra. IP_i perturbációja

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. A mi megközelítésünkben az improvizáció a többé vagy kevésbé harmonikus melódia véletlenszerű perturbációját jelenti, ezért az improvizáció valóban közelebb van a valósághoz.

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 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 következőképpen írható le:

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

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

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

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

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 több-dimenziós hátizsák probléma

(MMKP), számos gyors és hatékony problémamegoldó lehetőséggel. G. Csébfalvi és A. Csébfalvi [5] 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, a következő:

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

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

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

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 ,,hangnélküli", 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 (FBI) eljárással (lásd Tormos és Lova [25]) és az új ,,head-tail" lokális kereső heurisztikával, a karmester megpróbálja javítani a létrehozott dallam minőségét. Természetesen a karmester megjegyzi az addigi legrövidebb lehetséges melódiát, azaz ütemezést.

A ,,head-tail" lokális kereső 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 drámaian megnövekedhetnek a ,,head-tail" méretének függvényében. Így tehát egyensúlyt kell teremtenünk a költségek és a minőség között.

4 Számítási eredmények

Az algoritmust a jól ismert PSPLIB (http://129.187.106.231/psplib/) tesztkönyvtár J30MM alkönyvtárának projektjein teszteltük. A J30 halmazban a valóságos tevékenységek száma 30, két megújuló és két nem-megújuló

erőforrás szükséglettel. Mindegyik valódi tevékenységet a három megvalósítási mód valamelyikében lehet végrehajtani. A J30 halmaz 640 projektből áll, de néhány projektre nem létezik elsőbbségi és erőforrás korlátokat kielégítő megoldás, ezért ezeket nem vettük figyelembe. Erre a halmazra nem ismert az összes optimális megoldás, így először az optimális megoldásokat hoztuk létre.

Az egzakt megoldások generálására a jól ismert MILP megoldó szoftvert (CPLEX 8.1) használtuk, alapértelmezett beállításokkal, 900 sec-os időkorláttal. A megadott időkorláton belül 382 esetben sikerült az optimális megoldást elérni a CPLEX használatával, amely nagyon jól jelzi az MRCPSP nehézségi fokát. Az itt bemutatott algoritmus Visual C++$^{(R)}$ 6.0 és Visual Basic$^{(R)}$ 6.0 nyelven íródott.

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$^{(R)}$ operációs rendszer futott rajta. A futtatási eredményeket a *2-5. táblázatokban* foglaltuk össze. A megoldás minőségét a projekt időszükségletének és az optimális időszükséglet értékének százalékos eltérésével mértük (QM). Minden projekt példányra vonatkozóan 30 egymástól független futtatást végeztünk, hogy az eredményeink statisztikailag is szignifikánsak legyenek.

Átlag	Szórás	Minimális idő	Maximális idő	Megoldott esetek
25.44	98.82	0.02	665.00	382

2. táblázat. Megoldási idő CPLEX 8.1 (sec)

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

3. táblázat. „Sounds of Silence" eredmények

A *4. táblázatban* 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 jobb megoldást talált a bemutatott algoritmus az egzakt CPLEX megoldásnál.

Példány neve	Minimum QM (%)	Maximum QM (%)	Átlag QM (%)	Szórás
J30MM-14-2	0.00	6.25	3.12	1.1606
J30MM-14-10	0.00	3.45	1.73	1.7545
J30MM-22-2	-5.13	-5.13	-5.13	0.0000
J30MM-22-8	3.03	9.09	8.69	1.3155
J30MM-25-1	2.94	14.71	8.82	3.3684
J30MM-34-5	2.22	6.67	5.56	1.5192
J30MM-35-9	-2.70	-2.70	-2.70	0.0000
J30MM-38-10	-7.50	-7.50	-7.50	0.0000
J30MM-39-3	2.00	4.00	2.07	0.3651
J30MM-39-5	-10.81	-10.81	- 10.81	0.0000
J30MM-39-8	-2.22	-2.22	-2.22	0.0000
J30MM-40-1	-7.32	-4.88	5.45	1.0496
J30MM-40-3	0.00	5.56	1.39	2.1596
J30MM-41-5	7.50	7.50	7.50	0.0000
J30MM-42-2	-6.90	-3.45	-4.37	1.5517
J30MM-42-5	5.71	8.57	6.00	0.8727
J30MM-42-7	-6.06	3.03	-1.41	2.6069
J30MM-43-6	0.00	10.34	6.32	2.7281
J30MM-46-1	0.00	5.71	2.10	1.4887
J30MM-46-3	-15.79	-5.26	-9.65	3.1945
J30MM-46-5	2.94	11.76	7.15	2.5236
J30MM-46-9	-2.27	0.00	-0.08	0.4144
J30MM-47-3	-3.23	6.45	3.01	2.3862
J30MM-47-5	0.00	0.00	0.00	0.0000
J30MM-48-3	-5.41	2,70	-3.06	2.7255
J30MM-54-3	-12.00	0.00	-6.93	2.7660
J30MM-62-2	0.00	0.00	0.00	0.0000
J30MM-62-3	0.00	3.85	2.70	1.7945

4. táblázat. Minőségi mutató (QM) értékek 30 független futtatásra

A táblázatban látható értékek azt mutatják, 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 jobb megoldást talált, mint az egzakt CPLEX által adott optimum. 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 mutatja, 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. Ez a bemutatott hibrid algoritmus robusztus és hatékony tulajdonságát bizonyítja.

Az *5. táblázatban* a vizsgált esetek CPU felhasználási idejét gyűjtöttük össze, szintén 30 egymástól független futásokra vonatkozóan.

Példány neve	Átlagos idő sec	Min idő sec	Max idő sec	Szórás
J30MM-14-2	3.1921	2.3910	4.0930	0.4990
J30MM-14-10	1.2883	0.7490	3.5540	0.6468
J30MM-22-2	0.4259	0.2520	0.6060	0.0828
J30MM-22-8	0.4336	0.1860	0.6420	0.0995
J30MM-25-1	0.3976	0.2050	0.5270	0.0772
J30MM-34-5	4.4782	3.4280	5.8310	0.6140
J30MM-35-9	7.1133	5.4790	9.4320	0.9328
J30MM-38-10	3.7818	2.8890	4.5220	0.3704
J30MM-39-3	8.2220	6.3240	11.4020	1.2037
J30MM-39-5	3.0148	2.3430	4.0290	0.3526
J30MM-39-8	1.8616	1.4050	2.6730	0.2755
J30MM-40-1	8.1606	5.8620	10.9770	1.2722
J30MM-40-3	5.6851	4.0190	7.8760	0.8806
J30MM-41-5	5.2309	3.4060	7.8730	0.8387
J30MM-42-2	8.9140	5.7620	15.2710	2.1945
J30MM-42-5	16.8075	8.9740	44.1380	6.8167
J30MM-42-7	3.9662	3.1460	6.4170	0.6649
J30MM-43-6	6.6160	3.3730	10.7850	1.96654
J30MM-46-1	4.9304	3.7170	6.4900	0.6255
J30MM-46-3	5.4374	3.3970	7.6860	1.0986
J30MM-46-5	5.8117	4.0500	8.7030	1.1988
J30MM-46-9	4.9696	3.8870	5.7170	0.3703
J30MM-47-3	6.2602	4.2940	9.7590	1.2650
J30MM-47-5	6.9102	5.0200	11.2050	1.4175
J30MM-48-3	4.4953	3.4510	5.5500	0.5164
J30MM-54-3	9.0400	6.2750	16.8730	2.2133
J30MM-62-2	0.4501	0.2940	0.6100	0.0740
J30MM-62-3	0.4831	0.3560	0.6870	0.0676
Átlag, szélsőérték	4.9421	0.1860	44.1380	1.2922

5. táblázat. Felhasznált CPU idő 30 egymástól független futtatásra vonatkozóan

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 a maximális futási idő 10 és 17 másodperc közé esett. A J30MM-42-5 eset maximális CPU idő felhasználása jóval magasabb volt, mint a többi eseté. A már említett J30MM-42-5 eset kivételével, az átlagos futási idők 10 másodpercnél kisebbek voltak. Az átlagos futási idő az összes példány 30 független futtatására vonatkozóan 4.9421 másodperc volt.

Az eredmények alapján állítható, hogy a bemutatott algoritmus a J30MM halmazon versenyképes a jelenleg legjobb J30MM-re alkalmazott populáció alapú heurisztikával, 5000 iterációval (egy szimulált hűtés algoritmus, amelyet Bouleimen és Lecocq [4] fejlesztett ki), amely biztató előzetes eredmény. Ebben az esetben az átlagos eltérés az optimális megoldástól 2.61%. A legjobb megoldást a J30MM halmazra Jarboui et al. [12] publikálták (2.35%), de a megoldási időt vagy egyéb más minőségi paramétert nem közöltek.

5 Összefoglalás

Ebben a tanulmányban bemutattunk egy javított harmóniakereső metaheurisztikát az MRCPSP-re. A bemutatott algoritmusban az eredeti „Sounds

of Silence" harmóniakereső algoritmust kombináltuk egy új, hatékony ,,head-tail" lokális kereső eljárással, amely a vegyes egészértékű lineáris (MILP) formulán alapul és az eredetileg teljesen véletlenszerű kezdő repertoárt (véletlenszerű megvalósítási módok és tevékenység kezdő időpontok), helyettesítettük egy elő-optimalizált dallamhalmazzal, amelyet egy relaxált megoldásból állítottunk elő véletlen perturbációval. Az improvizációs lépések során a MILP és LP formulák megoldásával az egyes tevékenységek megvalósítási módjára vonatkozó listát és egy, a tevékenységek sorrendjére vonatkozó listát kapunk. A ,,karmester" ezekhez a megvalósítási módokhoz és tevékenységekhez próbál megfelelő ütemezést találni. A további lépések során mindig azt az ütemezést cseréljük, amelynél van jobb, rövidebb ütemezés. Ezután az algoritmus a head-tail eljárással megpróbálja tovább csökkenteni a projekt teljes időszükségletét, a kezdő és befejező tevékenységekhez rendelt erőforrások újraelosztásával. Az optimalizálás eredménye egy optimális, legrövidebb erőforrás-korlátos több megvalósítási módú projektütemezés. A számítási eredmények azt mutatták, hogy a ,,Sounds of Silence" algoritmus javított több megvalósítási módú verziója egy gyors és magas minőségű algoritmus.

Irodalom

1. Alcaraz, J., C. Maroto, R. Ruiz, "Solving the multi-mode resource constrained project scheduling problem with genetic algorithms", *Journal of the Operational Research Society*, 54, 614–626, 2003.
2. Boctor, F. F., "A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes", *European Journal of Operational Research*, 90, 349–361, 1996.
3. Boctor, F. F., "Heuristics for scheduling projects with resource restrictions and several resource-duration modes", *International Journal Production Research*, 31, 11, 2547–2558, 1993.
4. Bouleimen, K., H. Lecocq, "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
5. Csébfalvi, G., A. Csébfalvi, "A new metaheuristic for the multidimensional 0-1 knapsack problem", in *Computational Management Science Conference 2008*, Imperial College, London, UK, 38–39, 2008.
6. Csébfalvi, G., A. Csébfalvi, E. Szendrői, "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, 56–59, 2008.
7. Csébfalvi, G., Sounds of Silence: a harmony search metaheurictic for the resource-constrained project scheduling problem. *European Journal of Operational Research*, (under reviewing process), 2007.
8. Demeulemeester, E., B. DeReyck, W. Herroelen, "The discrete time/resource trade-off problem in project networks: a branch and bound approach", *IEE Transactions*, 32, 1059–1069, 2000.
9. Drexl, A., J. Grünewald, "Nonpreemptive Multi-mode resource constrained project scheduling, *IEE Transaction*, 25, 5, 74–81, 1993.

10. Hartmann, S., "Project scheduling with multiple modes: a genetic algorithm", *Annals of Operation Research*, 102, 111–135, 2001.
11. Hartmann, S., A. Drexl, "Project scheduling with multiple modes: a comparison of exact algorithms ", *Networks*, 32, 283–297, 1998.
12. Jarboui, B., N. Damak, P. Siarry, A. Rebai,"A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems", *Applied Mathematics and Computation*, 195, 299–308, 2008.
13. Józefowska, J., M. Mika, R. Rozycki, G. Waligora, J. Weglarz, "Simulated annealing for multi-mode resource- constrained project scheduling", *Annals of Operations Research*, 102, 137–155, 2001.
14. Kolisch, R., A. Sprecher, „PSPLIB – a project scheduling library", *European Journal of Operational Research*, 96, 205–216, 1996.
15. Kolish, R., A. Drexl, "Local search for nonpreemptive multi-mode resource-constrained project scheduling", *IEE Transactions*, 29, 987–99, 1997.
16. Lee, K. S., Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice", *Computer Methods in Applied Mechanics and Engineering*, 194, 3902–3933, 2005.
17. Lova, A., P. Tormos, M. Cervantes, F. Barber, "An efficient hybrid genetic algorithm for scheduling projects with resource-constrained and multiple execution modes", *International Journal of Production Economics*, 117, 302–316, 2009.
18. Özdamar, L., "A genetic algorithm approach to a general category project scheduling problem", *IEEE Transactions on Systems, Man and Cybernetics*, Part C: 29, 44–59, 1999.
19. Özdamar, L., G. Ulusoy, "A local constrained based analysis approach to project scheduling under general resource constraints", *European Journal of Operational Research*, 79, 287–298, 1994.
20. Patterson, J. H., F. B. Talbot, R. Slowinski, J. Weglarz, "Computational experience with a backtracking algorithm for solving a general class of precedence and resource-constrained scheduling problems", *European Journal of Operational Research*, 49, 68–79, 1990.
21. Sprecher, A., A. Drexl, "Solving multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm", *European Journal of Operational Research*, 107, 431–450, 1998.
22. Sprecher, A., S. Hartmann, A. Drexl, "An exact algorithm for the project scheduling with multiple modes", *OR Spektrum*, 19, 195–203, 1997
23. Szendrői, E., "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
24. Talbot, F. B., "Resource-constrained project scheduling with time-resource tradeoffs: the nonpreemptive case", *Management Sciense*, 28, 10, 1197–1210, 1982
25. Tormos, P., A. Lova, "A competitive heuristic solution technique for resource-constrained project scheduling", *Annals of Operations Research*, 102, 65–81, 2001.
26. Zhang, H., C. Tam, H. Li, "Multi-mode project scheduling based on particle swarm optimization", *Computer-Aided Civil and Infrastructure Engineering*, 21, 93–103, 2006.

A HYBRID OPTIMIZATION ALGORITHM FOR SOLVING MULTI-MODE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEMS

This paper presents a hybrid algorithm for the multi-mode resource-constrained project scheduling problem (MRCPSP). In the presented approach a harmony search algorithm is combined with a new and effective „head-tail" local search procedure based on a mixed integer linear programming (MILP) formulation. In order to illustrate the essence and viability of the proposed new approach, we present computational results for the J30MM set from PSPLIB. To generate the „head-tail" improvements a state-of-the-art callable MILP solver (CPLEX) was used.