Alkalmazott Matematikai Lapok **33** (2016), 27–56.

KARBANTARTÁSI PROJEKTEK MÁTRIX ALAPÚ TERVEZÉSE

KOSZTYÁN ZSOLT TIBOR, PRIBOJSZKI-NÉMETH ANIKÓ, KOVÁCS ZOLTÁN

Az ütemezési feladatok közül az egyik legnehezebb probléma, amikor egy projekt során a tevékenységek idő-, költségigényei mellett a projekt minőségi paramétereit harmonizálnunk kell egymással. Feltételezzük, hogy az idő-költség és a karbantartás esetében a rendszerelemek javító-megelőző tevékenységek hatásaként fellépő megbízhatóság növekedése között valamilyen függ-vénykapcsolat létesíthető. Ha pl. egy projektet rövidebb idő alatt kell elvégezni, akkor az többletköltséget igényel. Ugyanígy többletköltséggel járhat egy magasabb rendszer szintű megbízhatóság elérése. A szakirodalom ezeket a problémákat time-cost-quality trade-off problem (TCQTP) problémaosztályba sorolja, ahol ebben a példában a minőségi paraméter a rendszer megbízhatóságának növekedése lesz. Amennyiben az idő, a költség és a minőség között az ún. átváltási függvény diszkrét, akkor ez a probléma bizonyítottan NP-nehéz feladat. Cikkünkben bemutatjuk, hogy a megelőző karbantartási projekt is kezelhető és visszavezethető ilyen problémává, ugyanakkor ehhez ki kell terjesztenünk az eredeti problémaosztályra kifejlesztett módszerek alkal-mazási kereteit. Egy karbantartási projekt ugyanis tartalmazhat, sőt legtöbbször tartalmaz is körfolyamatokat. Egy megelőző karbantartási projekt célja, hogy egy rendszer berendezési elemeit javítva a rendszer megbízhatóságát növelje, ugyanakkor egy meghatározott minimális megbízhatósági, vagy rendelkezésre állásbeli javuláshoz általában nem fogjuk valamennyi berendezéselemet javítani, hanem ebből ki kell válogatni azokat a javító megelőző tevékenységeket, melyek az elvárt minimális megbízhatóság-javulás érdekében elengedhetetlenek. Éppen ezért a hagyományos hálótervezési módszerekkel szakítva olyan mátrix alapú módszerek alkalmazása felé fordulunk, melyek képesek kezelni a bizonytalan tevékenység-előfordulásokat és a bizonytalan tevékenységkap-csolatokat is, ezáltal kiterjesztve az eredeti problémát sztochasztikus hálóstruktúrák kezelésére is.

1. Bevezetés

A berendezések komplexitása folyamatosan nő, ezzel párhuzamosan a karbantartás tárgyköre mára már nemcsak a berendezések állapotának megőrzésével és helyreállításával foglalkozik, hanem kiterjedt a berendezések egységeire is. A gazdasági kényszer és a megbízhatósággal szembeni követelmények arra ösztönzik a

vállalatokat, hogy növeljék a termelő berendezéseik megbízhatóságát, ugyanakkor ésszerűsítsék a karbantartási és javítási költségeket és az üzemfenntartás hiányosságaira visszavezethető hibákat. A nagyobb karbantartási feladatok ún. karbantartási projektekbe szervezhetők. Ezek a karbantartási projektek (idő-, költség- és erőforrás)korlátok közé szorítják a terület szakembereit. *A lehető legrövidebb idő alatt kell a lehető legnagyobb mértékben javítani a rendszer megbízhatóságát vagy a rendelkezésre állását úgy, hogy a felhasznált költségeink minimálisak legyenek.*

A feladat komplexitását mutatja, hogy egyszerre kell megoldanunk egy projektkiválasztási (project screening) és egy, a tevékenységek idő-költség-minőség paraméterei közötti átváltási problémát (time-quality-cost trade-off problem): bár valamennyi berendezéselem megbízhatóságának javítására meghatározható egy vagy több javító-megelőző tevékenység, még egy ún. nagyleállás esetén – amikor szinte valamennyi berendezéselemet felülvizsgáljuk – sem fogjuk az összes lehetséges javító megelőző tevékenységet végrehajtani. Az első kérdés, amit ilyenkor meg kell válaszolnunk, hogy egy adott költség- és időkeret esetén vajon mely tevékenységeket kell/lehet majd végrehajtani?

A tevékenységeket általában többféleképpen is meg lehet valósítani, melyekhez különböző költség, idő és minőségi paraméterek rendelhetők. A projektmenedzsernek e paraméterek figyelembevételével kell egyensúlyoznia, hogy valamennyi javító-megelőző tevékenységet a(z idő-, költség)korlátokat nem túllépve végre tudja hajtani/hajtatni.

A feladat specialitása, hogy itt az ún. minőségi (quality) paramétert a megbízhatósági értékek javulásából fogjuk számolni, ami nem triviális feladat. A rendszert leíró ún. megbízhatósági diagram (angolul: Reliability Block Diagram, rövidítve: RBD) akár teljesen más struktúrát is követhet, mint amilyen struktúrát maga a karbantartási projekt követ. Egy berendezés elemhez általában több javító-megelőző tevékenység is rendelhető, melyek hatására növekedhet a berendezés elem megbízhatósága és ezáltal a rendszer megbízhatósága is, vagy egy másik számításnál a rendszer rendelkezésre állása. Egy berendezéselem minden önálló karbantartási egységet képező géprész, melynek/melyeknek meghibásodása során a karbantartás helyszínén azt/azokat tovább nem bontják.

Ebben a tanulmányban olyan, a kutatásunk során kifejlesztett mátrix alapú karbantartás-tervezési módszert (Matrix-based Maintenance Management Method = M^4) mutatunk be, amelyet sikerrel lehet alkalmazni berendezések karbantartásának tervezésére. Célunk, hogy a módszer alkalmazásával átláthatóbbá, egyszerűbbé tegyük az egyébként is bonyolultnak tűnő karbantartás-tervezést. Tesszük mindezt úgy, hogy a maximalizált rendszermegbízhatóság elérése érdekében törekszünk a berendezésegységek minél magasabb megbízhatóságára is amellett, hogy a vállalat által támasztott költség- és időterven belül maradjon az eredményként kapott karbantartási projektváltozat/-struktúra.

2. Szakirodalmi áttekintés

Mivel a javasolt módszer kombinálja az ún. átváltási módszereket a projektkiválasztási eljárásokkal, így e fejezetben elsősorban e területeket tekintjük át. A legtöbb költség-idő átváltási probléma már önmagában is általában kombinatorikus, ún. NP-nehéz feladat (lásd: 2.1. alfejezetet). A feladatot tovább bonyolítja a megbízhatósági paraméterek meghatározása, illetve ezek tervezés során történő figyelembevétele, így ezzel a területtel is külön foglalkozunk a 2.2. alfejezetben. A karbantartás- és projekttervezésnek egy mátrix alapú modell szolgáltat keretet. A 2.3. alfejezet bemutatja, hogy hogyan lehet a projekttervezési eljárásokat egy mátrixos modellben kombinálni.

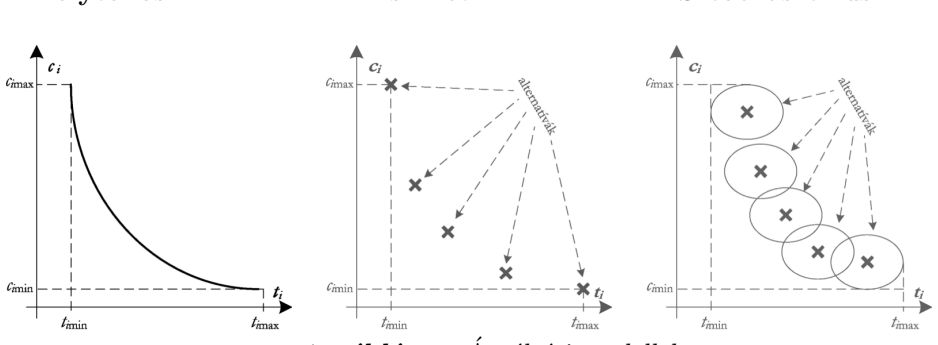
2.1. Idő-költség átváltási módszerek

Az idő- és költségcsökkentési eljárások már több mint ötven évre tekintenek vissza. Az elsők között még Fulkerson [13] foglalkozott olyan feladatokkal, ahol a tevékenységek időtartama és a költségigénye között feleltetett meg egy folytonos függvényt. Feltételezte, hogy amennyiben egy tevékenységet rövidebb idő alatt kell végrehajtani, akkor ahhoz több emberi erőforrás, drágább technológia, vagyis több (közvetlen) költségigény fog társulni. A probléma e változatát folytonos költség-idő átváltási problémának, angolul Continuous Time-Cost Trade-off Problem (CTCTP) nevezik. A nyolcvanas évekig nagyon élénk kutatás folyt ezen a területen. Az idő- és költségigények között nemcsak lineáris [18], hanem konvex [2], [26], sőt konkáv [11] függvénykapcsolatokat is tudtak kezelni.

A folytonos átváltási problémák általában nagyon gyorsan megoldhatók. A lineáris idő-költség átváltási problémákat visszavezetve minimális költségű folyamokra közel lineáris futásidőben végrehajtható algoritmusokat kaphatunk (lásd pl. [15], [16], [29]).

A valóságot azonban sokkal jobban modellezi az átváltási probléma diszkrét változata. Nehezen elképzelhető ugyanis folytonos függvény a tevékenységek időtartama és pl. az emberi erőforrásszükséglet között. Ugyanígy nehézkes folytonos függvényekkel jellemezni egy költségesebb, de időtakarékosabb technológia alkalmazásának hatását.

A diszkrét modellben úgynevezett végrehajtási módokat határozunk meg. Ehhez tartoznak idő- és költségadatok. Általában itt is feltételezzük, hogy a végrehajtási időtartam csökkentése többletköltséggel jár (lásd az 1. táblázatot). Mindkét változat esetén számos célfüggvényt határoztak meg a legegyszerűbbektől (pl. legrövidebb, legkisebb költségigényű projektterv megadása) egészen a komplex, adott költségigényt nem túllépő, lehető legrövidebb, vagy éppen az adott időszükségletet nem túllépő, lehető legkisebb költségű projektterv meghatározásáig [10]. Szemben azonban a folytonos eset gyors megoldási lehetőségeihez képest, ez a probléma ilyen komplex célfüggvények esetén néhány speciális típusú projekthálót leszámítva NP-nehéz [9].



1. táblázat. Átváltási modellek

A problémát tovább bonyolítja, ha az idő- és költségigényeket a pontos értékek helyett csak intervallumon tudjuk becsülni [12]. Ekkor az időtartamok és a költségigények közötti összefüggéseket egy pont helyett pl. egy szórásellipszissel jellemezhetjük (lásd az 1. táblázatot).

A karbantartási projekteknél a tevékenység-időtartamokat és a költségigényeket előre meg kell becsülnünk, ugyanakkor ilyen esetekben nehezen értelmezhető a tevékenységek és az időtartamok közötti folytonos függvénykapcsolat; éppen ezért az átváltási probléma diszkrét változatával foglalkozunk. Ugyanakkor az idő- és költségadatok mellett a rendszermegbízhatóság növekedésével, mint a projekt egyfajta minőségi paraméterével foglalkozunk, de mint azt láthatjuk, e „minőségi" paraméter tevékenységekhez való rendelése korántsem triviális feladat.

Egy karbantartási projekt esetén a minőségi paraméter a berendezések vagy berendezéselemek megbízhatóságának, vagy más számítások esetén rendelkezésre állásának (várható) javulásaként értelmezhető. Ugyanakkor ezeket a tevékenységeket berendezéselemekhez kell rendelnünk. A rendszer megbízhatóságának javulását pedig a megbízhatóságot leíró megbízhatósági diagram segítségével jellemezhetjük, aminek a struktúrája a karbantartási projekt struktúrájától jelentős mértékben eltérhet. Elképzelhető, hogy egy alacsony megbízhatóságú rendszerelem jelentős mértékű javítása sem fogja a rendszer megbízhatóságát számottevően emelni, hiszen, ha pl. egy alacsony megbízhatóságú elem meghibásodásakor egy tartalékrendszer a feladatokat átveszi, akkor ez kisebb mértékű zavart okoz, mintha egy jóval magasabb megbízhatóságú, de tartalék rendszerrel nem rendelkező berendezés elem esik ki, és veszélyezteti a teljes rendszer működését. A tartalékbeépítési tevékenység még az egyszerűbb eset, mert mind a költségek, mind pedig a megbízhatóság várható javulása jól jelezhető előre. Az egyéb, például meghibásodást megelőző beavatkozásoknál a költségek még viszonylag jó közelítéssel (bár sokszor csak a berendezés megbontása után), a várható megbízhatóság azonban nehezebben adható meg. Éppen ezért a javasolt modellben a minőségjavulás

kiszámítása során a megbízhatósági diagramot is figyelembe kell vennünk, ahogyan arra a 2.2. fejezetben részletesen is kitérünk.

1996-ban Babu és Suresh [1] voltak az elsők, akik azt javasolták, hogy az időtartam és költségigény közötti kapcsolatok mellett a minőség-költség és minőség-idő relációkat is értelmezzék. A problémát idő-költség-minőség átváltási problémaként határozták meg (angolul: Time-Cost-Quality Trade-off Problems, rövidítve: TCQTP). A modelleknél nemcsak azt feltételezik, hogy az időtartam rövidítése közvetlen költségnövekménnyel jár, hanem azt is, hogy a magasabb minőség elérése is általában más, drágább technológiát igényel.

E problémakör leggyakrabban alkalmazott diszkrét változata (DTCQTP) is NP-nehéz feladat, hiszen maga a diszkrét idő-költség átváltási feladat is NP-nehéz, ezért a legtöbb kutató ([35],[31]) igyekezett valamilyen heurisztikus közelítő megoldást adni a probléma kezelésére.

Valamennyi itt bemutatott modell abból indul ki, hogy a projekttervek, illetve a projektet megadó logikai hálóstruktúrák változatlanok. Ugyanakkor a karbantartási projektek esetén még az ún. nagyleállások esetén sem fogjuk valamennyi lehetséges javító megelőző tevékenységet elvégezni. Ki kell ezek közül választani azokat tevékenységeket, amelyek végrehajtása után egy kívánt rendelkezésreállást, vagy egy rendszermegbízhatósági szintet el tudunk érni, de emellett nem lépünk túl egy adott költségkeretet, ráadásul mindezeket a javításokat a folyamatos termelés fenntartása érdekében a lehető leghamarabb végre is tudjuk hajtani.

A feladat megoldásához először tehát el kell tudnunk dönteni, hogy mely tevékenységeket milyen sorrendben hajtjuk végre, végül pedig választanunk kell a megvalósítási alternatívák közül.

Cikkünkben egy új módszert javaslunk, amely segíti a projektmenedzserek munkáját abban, hogy mely tevékenységek megvalósíthatók egy adott szűkös költség- és időkeret között. Kombináljuk a tevékenységek kiválasztását a hagyományos átváltási modellekkel, létrehozva egy új feladatosztályt, melyet hibrid átváltási problémáknak neveztünk el. Ezek közül most a diszkrét változatra mutatunk egy példát, melynek neve angolul: Hybrid Discrete Time-Cost-Quality Trade-off Problem (HDTCQTP). Maga a feladatosztály sokkal bővebb, mint amit most cikkünkben körül tudunk járni. Lehetőségünk most csak a megelőző karbantartási projektek vizsgálatára szorítkozik, melynek angol neve: Preventive Maintenance Project Scheduling Problem (PMPSP).

2.2. Komplex rendszerek megbízhatósága

A megbízhatósági blokkdiagrammal (angolul: Reliability Block Diagram, rövidítve: RBD) általában a rendszer megbízhatóságát (angolul Total System Reliability, rövidítve: TSR) határozhatjuk meg, amit a továbbiakban úgy értelmezünk, mint a helyes működés valószínűsége egy adott időintervallumban. Hasonlóan, a megbízhatósági blokkdiagram segítségével számítható ki a rendelkezésreállás, amely megmutatja, hogy egy adott időintervallum mekkora hányadában működő-

képes a rendszerünk. A rendszerelemeket kétállapotúnak tekintjük, de ez a módszer szempontjából nem jelent megkötést, mert a valószínűség helyett kiterjesztett megbízhatóságkoncepció, például kapacitáskihasználás is alkalmazható lenne.

Az RBD megmutatja, hogy milyen logikai kapcsolat van a rendszer működéséhez szükséges elemek között. A blokkdiagramnak is számos változata ismert (lásd pl. Gertsbackh [14], illetve Idhammar [17] monográfiáit). A modellünkben minden berendezéselemről feltesszük, hogy bármely másiktól függetlenül működik. Egy rendszerelem lehet akár egy egész részrendszer, egy részegység, komponens vagy bármilyen más része a rendszernek.

Az egyszerű RBD-k soros vagy párhuzamos elemekből, vagy ezek kombinációiból épülnek fel [28].

A blokkdiagram módszer során az i-edik blokk: $r_i : \mathcal{R}_0^+ \to [0,1]$ megbízhatósági függvényének ismeretében határozzuk meg a műszaki rendszerek megbízhatóságát, modellünkben mindvégig feltételezve, hogy az egyes rendszerelemek meghibásodása egymástól független.

A teljes rendszer $TSR : \mathcal{R}_0^+ \to [0,1]$, eredő megbízhatósági függvénye, amely adott $t \in \mathcal{R}_0^+$ időpontban értelmezhető. Az eredő megbízhatóság számításának gyorsasága függ a rendszer komplexitásától. A legegyszerűbb eset, ahol csak ÉS, illetve VAGY kapcsolatok fodrulnak elő, mert ekkor a szorzási szabály alkalmazható. ÉS kapcsolat esetén a rendszer működéséhez valamennyi berendezéselemének működőképes állapotban kell lennie. Ekkor

$$TSR(t) = \prod_{i:=1}^{n} r_i(t).$$

A VAGY kapcsolat esetén a rendszer működéséhez elég, ha egy részrendszer át tudja venni a működés feladatait. Ekkor n párhuzamos blokk elrendezése esetén:

$$TSR(t) = 1 - \prod_{i:=1}^{n} (1 - r_i(t))$$

összefüggéssel határozható meg a rendszer megbízhatósága.

Természetesen nem minden rendszert tudunk ilyen egyszerű alapelemekkel leírni. Gyakran előfordul, hogy olyan rendszereket kell modellezni, amikor is nem lehet a rendszert ÉS-VAGY alrendszerekre szétbontani, ekkor segíthet az ún. *igazságtáblával [25]* vagy a *működési útvonalak* módszerével történő rendszermegbízhatóság-számolás [32], vagy a még számolásigényesebb szimulációs eljárások [24].

Az igazságtábla (angolul: Event Space Method, rövidítve: ESM) szerinti eredőszámítás [36] azon alapul, hogy kétállapotú – működő és meghibásodott – elemeket feltételezve, meghatározzuk a rendszer működését eredményező állapotkombinációk valószínűségének összegét. A módszer nagy előnye, hogy gyakorlatilag bármilyen rendszerstruktúra esetén meghatározható a rendszer eredő megbízhatósága. Még a rendszerelemek megbízhatóság szempontjából való függetlenségét sem használjuk ki. Ugyanakkor nagy hátránya az eljárásnak, hogy valamennyi

rendszerelem működés/nem működés kombinációját számba kell venni. Ez pedig n elem esetén 2^n lehetséges (működési- vagy hiba)állapotot jelent.

A működési útvonalak módszere (angolul: Path-Tracing Method, rövidítve: PTM) az igazságtáblához hasonlító eljárás [32]. A PTM során a „működési" utak valószínűségeinek unióját vesszük alapul. Ebben az esetben az utak metszeteinek levonására is szükség van, hogy a teljes rendszerre számolt megbízhatóság ne tartalmazzon redundáns adatokat. Ebből adódóan legrosszabb esetben itt is 2^n lehetséges kombinációt kell számba vennünk.

A dekompozíciós eljárás (angolul: Decomposition Method, rövidítve: DCM, lásd az 5. ábrán látható pszeudo kódot) a fentiekkel ellentétben egy gyors eljárás, amely a teljes valószínűség elvét alkalmazza. Első lépésként kiválaszt egy ún. kulcselemet. Mivel a kulcselem megválasztásától függ a módszer számításigénye, ezért célszerű olyan kulcselemet választani, amelyik legna-gyobb fokszámú berendezéselem a megbízhatósági diagramban. Jelölje rögzített $t > 0$ esetén $P(S) = TSR(t)$ az S (teljes) rendszer működési valószínűségét. Jelölje $P(K)$ egy $K \subset S$ kulcs elem működési valószínűségét ugyanebben $t > 0$ rögzített időpontban. Ekkor a teljes valószínűség elve szerint $TSR(t) = P(S) = P(S|K) \cdot P(K) + P(S|\overline{K}) \cdot P(\overline{K})$, ahol $P(\overline{K})$ jelöli a K kulcselem hibás működési valószínűségét, $P(S|K)$ pedig az S rendszer működési valószínűségét, ha feltételezzük, hogy $K \subset S$ kulcselemünk működőképes állapotban van. A kulcselem kiválasztása után, ha a hálózat ÉS-VAGY kapcsolódású elemekre bontható, akkor ezeket az elemeket összevonjuk, így határozva meg a rendszer megbízhatóságát; míg ha nem egyszerűsíthető a hálózat, akkor újabb kulcselemet választunk. A módszert a teljes megbízhatóság meghatározásáig ismételjük. A módszer pszeudo kódja megtalálható a Mellékletben (5. ábra).

A bemutatott módszerek nem csak abban segíthetnek, hogy meghatározzuk a rendszer megbízhatóságát, hanem abban is, hogy egy-egy berendezéselem javítása után mennyivel nő a rendszer megbízhatósága. Modellünkben az egyszerűség kedvéért egy statikus mo-dellt alkalmazunk, ahol tehát $t > 0$ értékét rögzítjük, ugyanakkor egy következő cikkben bemutatjuk, hogy a megbízhatóságcsökkenést hogyan lehet felhasználni az ún. prediktív karbantartás során, ahol már a megbízhatóság időbeli változását is figyelembe vesszük.

2.3. Mátrix alapú projekttervezés

A berendezések megbízhatóságának, rendelkezésre állásának növelésére javító-megelőző tevékenységeket határoznak meg. A tevékenységek meghatározásának alapja lehet egy ún. hibamód és -hatás elemzés (angolul: Failure Mode and Effect Analysis, rövidítve: FMEA), ahol az egyes hibamódokhoz javító megelőző tevékenységeket rendelhetünk. Az eredeti FMEA modellben még egy hibamódhoz egy javító-megelőző tevékenység társult, ugyanakkor ezt a megkötést számos továbbfejlesztés feloldotta (részletes áttekintést olvashat az olvasó Sutrisno és Lee [34] dolgozatában).

A javító-megelőző tevékenységekből állíthatjuk össze a karbantartási projekteket. Ezek a projektek azonban speciális szerkezetűek, merőben eltérhetnek a hagyományos, pl. építési, beruházási projektektől.

A legfontosabb eltérés, hogy itt a legritkább esetben fog megvalósulni, hogy valamennyi javító-megelőző tevékenység megvalósul. Ez alól nem kivétel sem a nagyleállás, sem az időszakosan elvégzett ún. fővizsgálat sem.

A másik fontos eltérés, hogy a karbantartási projektek esetén nagyon sokszor fordul elő, hogy bizonyos tevékenységeket újra és újra el kell végeznünk egészen addig, ameddig a kívánt berendezésmegbízhatósági szintet el nem érjük.

A lehetséges visszatérések, mint körfolyamatok kezelése még nem igényelné feltétlen a mátrixos tervezést, hiszen nagyon korán, a hatvanas évek végén Pritsker [30] tanulmányában már kezelte ezt a problémát, de bizonyos javító-megelőző tevékenységek elhagyhatósága már kikényszeríti a hálós tervezési eljárások meghaladását.

A hálós tervezési eljárásoknál már a kifejlesztésük során gondoltak arra, hogy a hálós terveket majd mátrixok tárolják [6]. Ekkor a szomszédsági mátrix segítségével jelölhetők a tevékenységek. A mátrix celláiban lévő tevékenységekhez pedig rendelhetők idő- és költségadatok is. Kifejezetten mátrixos projekttervezési módszer kidolgozása Steward [33] nevéhez köthető. Itt már nem megjelenési forma a mátrix, hanem a tervezés eszköze. A mátrix cellái nem a tevékenységeket, hanem a tevékenységek közötti kapcsolatokat jelenítették meg. Az így kapott négyzetes szomszédsági mátrixot Dependency Structure Matrix (röviden DSM) módszernek nevezték el.

Amíg a DSM-mátrix egy determinisztikus rendszert, illetve projektek esetén háló struktúrát feltételezett, ahol a tevékenységek közötti kapcsolatokat szigorú rákövetkezési kapcs-olatokként tekintettük, a továbbfejlesztéseként javasolt Numerikus DSM (NDSM) mátrix [39] már képes volt modellezni a rákövetkezési kapcsolatok fontosságát, illetve valószínűségét is.

Elsőként Kosztyán és mtsai [20] mutattak rá arra, hogy ha valószínűségekként kezeljük a rákövetkezési relációkat, akkor attól függően, hogy egy rákövetkezést betartunk vagy elhagyunk, különböző hálóstruktúrát, illetve ezeket jellemző DSM-mátrixokat fogunk kapni. Az így kapott struktúrákat projektstruktúráknak nevezték el. A módszert, amellyel ezeket meg lehet határozni, pedig sztochasztikus hálótervezési eljárásnak (angolul: Stochastic Network Planning Method, rövidítve: SNPM) [22].

Azokat a kapcsolatokat, amelyek valószínűsége kisebb, mint egy, de nagyobb, mint nulla, *bizonytalan kapcsolatok*nak nevezzük. Ha megadjuk ezek valószínűségeit, akkor meg tudjuk határozni a lehetséges projektstruktúrák bekövetkezési valószínűségeit is [22]. A modell nagy előnye a hagyományos hálótervezési eljárásokhoz képest, hogy itt nem szükséges valamennyi lehetséges alternatíva hálóstruktúráját meghatározni, hanem egyetlen, ún. SNPM-mátrixban jelölni lehet a biztos és a bizonytalan kapcsolatokat is.

PDM — **Determinisztikus** — **Nem determinisztikus**

Számszerűsített

PDM	A	B	C	D	E	t	c	r_1	r_2
A	0.8	1	0.8	0.2	0.1	4	2.4	2	1
B		1		0.4	0.9	2	1.8	3	3
C			0.1		0.1	4	9.6	8	5
D				0.2	0.3	10	42	12	2
E					0.7	3	0.9	1	2
Logic Domain (LD)						TD	CD	Resource Domain	

PDM	A	B	C	D	E	t_{min}	t_{max}	c_{min}	c_{max}	r_{1min}	r_{1max}	r_{2min}	r_{2max}
A	0.8	1	0.8	0.2	0.1	4	5	2.4	3.2	2	3	1	2
B		1		0.4	0.9	2	3	1.8	2.0	3	4	3	3
C			0.1		0.1	4	6	9.6	9.6	8	10	5	6
D				0.2	0.3	10	12	42	44	12	15	2	4
E					0.7	3	6	0.9	1.1	1	4	2	3
Logic Domain (LD)						Time Domain		Cost Domain		Resource Domain (RD)			

Nem számszerűsített

PDM	A	B	C	D	E	t	c	r_1	r_2
A	?	X	?	?	?	4	2.4	2	1
B		X		?	?	2	1.8	3	3
C			?		?	4	9.6	8	5
D				?	?	10	42	12	2
E					?	3	0.9	1	2
Logic Domain (LD)						TD	CD	Resource Domain	

PDM	A	B	C	D	E	t_{min}	t_{max}	c_{min}	c_{max}	r_{1min}	r_{1max}	r_{2min}	r_{2max}
A	?	X	?	?	?	4	5	2.4	3.2	2	3	1	2
B		X		?	?	2	3	1.8	2.0	3	4	3	3
C			?		?	4	6	9.6	9.6	8	10	5	6
D				?	?	10	12	42	44	12	15	2	4
E					?	3	6	0.9	1.1	1	4	2	3
Logic Domain (LD)						Time Domain		Cost Domain		Resource Domain (RD)			

2. táblázat. Project Domain Matrix

A módszer továbbfejlesztéseként egy ún. projekt szakértői mátrix segítségével [21] (angolul: Project Expert Matrix, rövidítve: PEM) már a bizonytalan tevékenység-előfordulások is modellezhetők. Az ilyen bizonytalan tevékenység-előfordulásokat és bizonytalan kapcsolatokat tartalmazó projekttervek esetén először arról kell döntenünk, hogy mely tevékenységet fogjuk végrehajtani [19]. Eredményül SNPM-mátrixszal jellemezhető ún. *projektváltozat*okat kapunk. A második fázisban a korábban ismertetett módszerekkel kell arról döntenünk, hogy a tevékenységeket milyen sorrendben hajtjuk végre.

A mátrixalapú projekttervezési eljárásokat nem csak logikai tervezésre, hanem ütemezésre [7], [27], valamint költség- és erőforrástervezésre is alkalmazták [3], [38], [4], [23].

A modellezéshez az $n \times n$ -es logikai struktúrát leíró mátrix mellé további oszlopokat, ún. *domain*-eket határoztak meg az idő-, költség- és erőforrásadatok jelölésére. Az ilyen mátrixokat angulul Domain Mapping Matrixoknak (DMM) nevezték el a kutatók [8]. A PEM mátrix idő-, költség- és erőforrásadatokkal való kiterjesztése az ún. Project Domain Matrix (PDM) [23].

A PDM-mátrixból négy változatot határoztak meg attól függően, hogy a bizonytalan tevékenység-előfordulásokat, illetve tevékenységrelációkat számszerűsítjük, vagy sem (specified PDM/ semi-specified PDM); illetve hogy az idő-, költség- és erőforrásadatoknak több alternatíváját is meghatározzuk-e, vagy sem (deterministic PDM/ non-deterministic PDM) [23].

A javasolt PDM-mátrix minden esetben négy részmátrixot, ún. *domain*-t tartalmaz. Az első $n \times n$ -es részmátrix a logikai kapcsolatokat (Logic Domain, LD) írja le egy PEM-mátrix segítségével. A módszer alkalmazásához nem szükséges a logikai kapcsolatokat és a tevékenységeket számszerűsíteni. Elegendő csak azt meghatározni, hogy a tevékenység-előfordulások, illetve a kapcsolatok biztosak („X"-szel jelöljük) vagy bizonytalanok („?"-lel jelöljük). Az üres cellák felelnek meg annak, ha két tevékenység között nem értelmezünk rákövetkezési relációt.

A tevékenység-előfordulásokhoz, illetve a kapcsolaterősségekhez különböző számszerűsített adatokat társítunk. Ezek lehetnek pl. a tevékenység/kapcsolat-előfordulások valószínűségei pl. hasonló projekteket alapul véve. Lehetnek fontossági vagy prioritási értékek is. Mi a most javasolt modellünkben eltekintünk attól, hogy ezeket az értékeket számszerűsítsük, így a PDM nem számszerűsített változatát használjuk fel az általunk javasolt modellben.

A következő részmátrix (Time Domain, TD) a tevékenységek időtartamát mutatja. Ha minden tevékenység időtartamát egyetlen számmal jellemezzük, akkor az időadatokat determinisztikusnak tekintjük. Lehetőség van azonban különböző megvalósítási alternatívákhoz tartozó időadatokat is megadni. A 2. táblázat utolsó oszlopában ezek közül csak a minimális, illetve a maximális időtartamot jelöltük.

A harmadik részmátrix (Cost Domain, CD) a tevékenységek közvetlen költségét jellemzi. A költségek is lehetnek determinisztikusak, ekkor egy tevékenységhez csak egyetlen költségalternatívát rendelünk. Hasonlóan a tevékenységekhez, itt is lehet akár több költségigényt is rendelni egyetlen tevékenységhez, modellezve, hogy a tevékenységek különbözőképpen, ebből adódóan pedig különböző költségigénnyel hajthatók végre. A költségigényeket itt tágabban, nem megújuló erőforrásként is lehet értelmezni.

A PDM-modell utolsó részmátrixa a megújuló erőforrásokat tartalmazó részmátrix (Resource Domain, RD). Ha r db erőforrással rendelkezünk, akkor determinisztikus esetben r oszlopból áll ez a részmátrix. Itt is lehetőség van azonban egy-egy alternatívához különböző erőforrás-igényt rendelni.

Kosztyán [23] a javasolt mátrixmodellen túl egy polinomiális rendű, gyors algoritmust is javasolt számszerűsített determinisztikus PDM-mátrixok kiértékelésére. A módszer kihasználta, hogy minden bizonytalan tevékenység-előfordulás esetén két lehetséges alternatíva között kell döntenünk, nevezetesen: vagy megvalósítjuk, vagy elhagyjuk a tevékenységet a projektből. Minden lépésnél ki lehet számítani, hogy mi a legkisebb költségű projektterv (a kötelezőkön kívül minden még bizonytalan tevékenység-előfordulás elhagyása), mi a lehető legrövidebb projektterv (minden (még) bizonytalan kapcsolat feloldása és a tevékenységek legkorábbi időpontra való ütemezése) [19]. Ha a két lehetséges alternatíva közül bármelyiknél teljesül, hogy a korlátként szabott minimális költségigényt a lehető legkisebb költségigényű projektváltozat is túllépi, akkor azt a döntési ágat nem érdemes továbbértékelni, mert a kitűzött korlátokon belül nem valósítható meg a projekt. A módszerről részletesen olvashat Kosztyán [23] tanulmányában. Ebben

az esetben a tevékenység-előfordulásokhoz és kapcsolaterősségekhez rendelt pontértékekkel lehetett meghatározni a projektváltozatok és projektstruktúrák pontértékeit, költség- és erőforrásigényeit. Polinomiális rendben lehetett meghatározni az első N legvalószínűbb, legfontosabb, legrövidebb, vagy éppen legkisebb költséggel rendelkező projektterveket anélkül, hogy szükség lett volna valamennyi projektterv meghatározására. Jelen cikkünkben azonban olyan projekttervekkel foglalkozunk, amelyekben a projekttervben szereplő tevékenység-előfordulásokhoz nem feltétlenül tudunk pontértéket rendelni. Ezek alapján a projektváltozatok pontértékeit sem tudjuk meghatározni. A karbantartási projektterv összeállításánál javító-megelőző tevékenységeket hajtunk végre, amelyek különböző technológiával, különböző költség- és időigényekkel járhatnak, tehát a 2. táblázatban bemutatott PDM-mátrixok közül a nem számszerűsített nem determinisztikus változatot kell alkalmaznunk, illetve megbízhatósági blokkdiagramot, a berendezéselemekhez rendelhető javító-megelőző tevékenységet és a be-csült megbízhatóságnövekedést beépítve továbbfejlesztenünk.

Ha többfajta logikai struktúrát kell összekapcsolnunk, akkor ki kell lépnünk a korábban tárgyalt $n \times m$ -es több részmátrixot tartalmazó összetett mátrixok adta lehetőségek köréből. Az ilyen típusú modellezési problémák kezelésére fejlesztették ki az ún. *Multi-Domain Matrix*-ot (MDM). Itt lehetőség van pl. egy projekt esetén a szervezeti hierarchia, a feladatstruktúra és az ütemterv összekapcsolására. Danilovic és Browning [8] tanulmánya azonban még mindig fix kapcsolatokat és tevékenység-előfordulásokat feltételezett. Ahogyan azt a korábbiakban már említettük, a karbantartási projekt a legritkább esetben tartalmaz minden javító-megelőző tevékenységet. Ebből adódóan a Danilovic és Browning által javasolt megoldások nem alkalmazhatók a karbantartási projekt korrekt leírására.

3. Karbantartási projektek mátrixos tervezése

Mielőtt a javasolt mátrix alapú eljárást ismertetnénk, először a 3.1. fejezetben a feladat matematikai leírását adjuk meg. A probléma modellezését egy mátrixmodell segítségével valósítottuk meg, melynek részletes ismertetését a 3.2. fejezetben tárgyaljuk. A probléma megoldására javasolt algoritmus három fázisból áll, melynek első két fázisa polinomiális rendben vezeti vissza a megelőző karbantartási feladatot a diszkrét idő-minőség-költség átváltási problémára, mely már megoldható egyrészt a korábban már kifejlesztett eszközökkel (lásd: 2.1. fejezetet), másrészt az általunk javasolt megoldással is.

3.1. A karbantartási probléma meghatározása

A probléma egy ún. hibrid idő-költség-minőség átváltási probléma (Hybrid Time-Cost-Quailty Trade-off Problem, HTCQTP) diszkrét változatának egy spe-

ciális aleseteként tekinthető. Jelen cikkben ezt a megelőző karbantartás-tervezési feladatot (angolul: Preventive Maintenance Project Scheduling Problem, rövidítve: PMPSP) formalizáljuk, mely mint látni fogjuk, a diszkrét idő-minőség-költség átváltási probléma (DTCQTP) általánosításaként tekinthető. A formalizmus megadásánál kihasználjuk, hogy a megvalósítás mátrixok segítségével történik, így az absztrakt megadáson kívül a mátrixos leírást is ismertetjük.

3.1. Definíció. Jelölje $K := \{k_1, k_2, \ldots, k_z\}$ a berendezések véges halmazát. A megbízhatósági diagram szomszédsági mátrixát megadó $z \times z$ mátrixot jelölje $\mathbf{K} \in \{0,1\}^{z \times z}$.

A megbízhatósági diagramról feltételezzük, hogy egyszerű gráffal jellemezhető (többszörös élt és hurokélt nem tartalmaz), ebből adódóan a szomszédsági mátrix átlója 0 értékeket tartalmaz, melyet a későbbiekben felhasználunk a kritikussági, megbízhatósági vagy rendelkezésre állási adatok jelölésére (lásd: 3.2. fejezetet).

3.2. Definíció. Jelölje $R : K \to [0,1]$ a berendezés(elem) megbízhatósági függvényét. Jelölje továbbá $TSR(K) \in [0,1]$ a teljes rendszer megbízhatóságát.

A megbízhatósági függvény jelen esetben egy rögzített $t > 0$ időpontban mutatja az $R_i = R(k_i)$ rendszerelem megbízhatóságát. Abban az esetben, ha ez a megbízhatósági érték egy úgynevezett cr_i kritikus megbízhatósági érték alá esik, akkor a berendezéselem javítására mindenképpen szükség lesz (lásd: 1. ábra).

3.3. Definíció. Jelölje $A := \{a_1, \ldots, a_n\}$ n elemű véges halmaz a berendezéselemek megelőző karbantartásához kapcsolható javító-megelőző tevékenységek halmazát. Jelölje a bizonytalan tevékenység-előfordulások halmazát $\widetilde{A} = \{\widetilde{a}_1, \ldots, \widetilde{a}_s\} \subseteq A$, valamint $\overline{A} = A \setminus \widetilde{A}$ a kötelezően végrehajtandó javító-megelőző tevékenységek halmazát.

3.4. Definíció. $\mathcal{A} = (A, \prec, \sim, \bowtie)$ struktúrát **hibrid projektterv**nek nevezzük, ahol $(\prec, \sim, \bowtie)$ bináris relációkat $\forall a_i, a_j \in A, i \neq j$ esetén a következőképpen értelmezzük.

(1) $a_i \prec a_j \Rightarrow a_i$ és a_j **között szigorú rákövetkezési reláció** áll fent. a_j a **követő**, míg a_i a **megelőző tevékenység**
(2) $a_i \sim a_j \Rightarrow a_j$ nem követi a_i tevékenységet
(3) $a_i \bowtie a_j \Rightarrow$ későbbi döntés eredményeként a_j követni fogja a_i-t, vagy nem. A döntés eredményéig az ilyen kapcsolatot **bizonytalan kapcsolatnak** nevezzük.

Amennyiben a bizonytalan kapcsolatok számossága $\left|\widetilde{A}\right|$ akkor a lehetséges projektváltozatok száma: $2^{|\widetilde{A}|}$. A tevékenységeket és kapcsolataikat egy $n \times n$-es PEM mátrix reprezentálhatja (lásd: 2.3. fejezetet).

3.5. Definíció. Jelölje $\Xi(A) := \left\{S \subseteq A : \overline{A} \subseteq S\right\}$ a **projektváltozatok halmazát**. A halmaz egy elemét a **kiválasztott projektváltozat**nak nevezzük: $S \in \Xi(A)$.

Egy kiválasztott projektváltozat már nem tartalmaz bizonytalan tevékenység-előfordulásokat. Egy $\mathcal{S} = (S, \prec, \sim, \bowtie)$ mátrix-reprezentációját egy SNPM mátrix írja le (lásd: 2.3. fejezetet). Mivel sem a bizonytalan tevékenység-előfordulásokat, sem pedig a bizonytalan kapcsolatokat nem számszerűsítjük, így a mátrixreprezentációkban a biztos kapcsolatokat jelölheti „X" illetve 1-es, a bizonytalan kapcsolatokat, illetve bizonytalan tevékenység-előfordulásokat „?" illetve $0,5$. A mátrixreprezentációkban azokat a kapcsolatokat/tevékenységeket, melyeket elhagyunk a projektből, jelölje üres cella, „$\emptyset$" vagy 0.

3.6. Definíció. $\mathcal{X} = (S, \prec, \sim)$ struktúrát **projektstruktúrá**nak nevezünk, ahol $S \in \Xi(A)$ egy kiválasztott projektváltozat.

Egy projektstruktúra már nem tartalmaz bizonytalan kapcsolatokat. A projektstruktúra logikai tervének mátrixreprezentációja egy szomszédsági vagy DSM-mátrix.

Az ütemezési és különösen az átváltási problémáknál nagyon gyakran felteszik, hogy a tevékenységgráf, amit itt a projektstruktúra jellemez, nem tartalmaz kört. Ezt itt úgy írhatnánk le, hogy $\prec$ reláció részben rendezés S halmazon. Ugyanakkor már a kezdeti mátrixtervezési módszerek (lásd: Steward [33]) is modellezték, detektálták (lásd: Xiao és mtsai [37]), illetve egynél kisebb valószínűségű visszacsatolásoknál fel is oldották (lásd: Kosztyán [23]) az ilyen visszacsatolásokat. Éppen ezért mi most csak az egyszerűség kedvéért tesszük fel, hogy egy projektstruktúra már nem tartalmaz köröket.

A javasolt algoritmus során minden bizonytalan tevékenység-előfordulásról döntünk, hogy megvalósítjuk vagy sem (1. fázis, lásd: 3.3. fejezetet). Ezután pedig minden bizonytalan kapcsolatról határozunk, hogy előírjuk (soros megvalósítás), vagy sem (párhuzamos megvalósítás). A módszert egészen addig folytatjuk, ameddig egyetlen bizonytalan kapcsolat sem marad a modellünkben (2. fázis, lásd: 3.3. fejezetet). Vagyis az eredményül kapott, a projekttervet leíró mátrixreprezentáció már egyetlen „?" szimbólumot sem fog tartalmazni.

3.7. Definíció. Tegyük fel, hogy minden $a \in A := \{a_1, a_2, \ldots a_n\}$ tevékenységet $m \in \mathbb{Z}^+$ módon tudunk végrehajtani. Tegyük fel továbbá, hogy minden tevékenységhez összesen r-féle erőforrást rendelhetünk, akkor minden $a \in A$ tevékenységre vonatkozó végrehajtási módok $r + 3$-asokkal írhatók le: $M_a := \{(t_a, c_a, \Delta R_a, r_{a_1}, r_{a_2}, \ldots, r_{a_r}) : 1, 2, \ldots, m\}$, melyek tartalmazzák a végrehajtási mód idő- (t_a), illetve költségigényét (c_a), valamint a javító-megelőző tevékenység adott berendezés elemre gyakorolt megbízhatóság-növekedését (ΔR_a), ezen kívül pedig az erőforrás-szükségleteket $(r_{a_1}, r_{a_2}, \ldots, r_{a_r})$.

Megjegyzés. Jelölje egy $a_j \in A$ tevékenység $v, w \in \{1, 2, \ldots, m\}$ végrehajtási módjainak időigényét: t_{jv}, t_{jw}, költségigényét: c_{jv}, c_{jw}, erőforrásigényeit leíró vektort: $\mathbf{r}_{jv}, \mathbf{r}_{jw}$, valamint a tevékenység hatására jelentkező megbízhatóságnövekedést: $\Delta R_{jv}, \Delta R_{jw}$. Az átváltási feladatoknál általában feltételezik, hogy $t_{jv} < t_{jw}$ esetén $c_{jv} \geq c_{jw}$, $\mathbf{r}_{jv} \geq \mathbf{r}_{jw}$ valamint $\Delta R_{jv} \leq \Delta R_{jw}$ teljesül, vagyis az időbeli rövidítés, pótlólagos (közvetlen) költségnövekménnyel (lásd: 1. táblázatot), illetve pótlólagos erőforrás-növekménnyel jár, miközben a rövidebb idő alatt kisebb mértékben lehet növelni a berendezéselemek és ezáltal a rendszer megbízhatóságát. Ugyanakkor látni fogjuk, hogy ezt az összefüggést a javasolt algoritmusunk során sehol nem használjuk ki. Csak annyit várunk el, hogy a különböző módokat tekintve az idő-, erőforrás- és költségigények korlátosak legyenek. Továbbá lehessen meghatározni a minimális és maximális szükségleteket, valamint a minimális és maximális megbízhatóságnövekményeket.

Miután meghatároztuk, mely tevékenységeket hajtjuk végre (1. fázis), valamint azt is megállapítottuk, hogy milyen sorrendben hajtjuk ezeket végre (2. fázis), ki kell választanunk hogy a tevékenységeket milyen módon hajtjuk végre (3. fázis). Eredményül egy úgynevezett projektütemtervet kapunk, amely tartalmazza a végrehajtandó tevékenységeket és a végrehajtás módját.

3.8. Definíció. Tekintsünk egy $\mathcal{X} = (S, \prec, \sim)$ projektsttuktúrát, ahol $S \in \Xi(A)$ egy kiválasztott projektváltozat. A **projektütemterv** azon $a \in S$ tevékenységekre vonatkozó idő-, költég-, erőforrásigények, illetve a javító-megelőző tevékenység okozta megbízhatóságjavulások halmaza, $\overrightarrow{s}_{\mathcal{X}} = \{s_a : a \in S\}$ ahol $s_a \in M_a$.

3.9. Definíció. Egy rögzített $\mathcal{X} = (S, \prec, \sim)$ projektstruktúrára vonatkozó $\overrightarrow{s}_{\mathcal{X}}$ projektütemtervre a **projekt teljes költsége (Total Project Cost, TPC)** a projektütemtervben szereplő tevékenységek költségigényeinek összegeként tekinthető: $c\left(\overrightarrow{s}_{\mathcal{X}}\right) := \sum_{(c,t,\Delta R,r_1,\ldots,r_r)=s_a \ ,\ a\in S, s_a \in M_a} c$.

Hasonlóan kiszámítható az $\overrightarrow{s}_{\mathcal{X}}$ átfutási ideje (Total Project Time, TPT), melyet jelöljön $t\left(\overrightarrow{s}_{\mathcal{X}}\right)$, valamint a rendszer megbízhatóságának (Total System Reliability, TSR) növekménye, a rendszer K berendezéseire, melyet jelöljön $\Delta TSR\left(K, \overrightarrow{s}_{\mathcal{X}}\right)$. Jelölje továbbá $\mathbf{r}_{\max}\left(\overrightarrow{s}_{\mathcal{X}}\right)$ az erőforrás-maximumokat tartalmazó r elemű vektort. (A függvények számításait az 5. és 7. pszeudo kódok tartalmazzák.)

A bevezetett jelölésekkel már megfogalmazható a megelőző karbantartási projekttervezési probléma.

1. Probléma. (a) Megelőző karbantartástervezési probléma (**Preventive Maintenance Project Scheduling Problem**, PMPSP), **legrövidebb átfutási idejű projektütemterv keresése**: Legyen $K := \{k_1, k_2, \ldots, k_z\}$ egy véges, berendezéselemeket tartalmazó halmaz. Legyen továbbá $A := \{a_1, a_2, \ldots, a_n\}$ véges tevékenységeket tartalmazó halmaz. Jelölje $S \in \Xi(A)$ egy kiválasztott

projektváltozatot $S \subseteq A$, valamint jelöljön $\mathcal{X} = (S, \prec, \sim)$ egy projektstruktúrát. A projektstruktúra egy lehetséges projektütemtervét jelölje $\overrightarrow{s}_{\mathcal{X}}$. Legyen $C_c \geq 0$ a költség, $C_t \geq 0$ az idő, $\mathbf{C_r} \geq \mathbf{0}$ pedig az erőforráskorlát vektora. Jelölje továbbá $1 \geq C_{\Delta TSR} \geq 0$ az előírásként tekinthető minimális rendszermegbízhatóság-növekményt.

$$\begin{aligned} &\arg\min t(\overrightarrow{s}_{\mathcal{X}}) \\ &\text{feltéve, hogy} \\ c(\overrightarrow{s}_{\mathcal{X}}) &\leq C_c \\ \mathbf{r}_{\max}(\overrightarrow{s}_{\mathcal{X}}) &\leq \mathbf{C_r} \\ 1 \geq \Delta TSR(K, \overrightarrow{s}_{\mathcal{X}}) &\geq C_{\Delta TSR} \end{aligned} \tag{1}$$

A fenti feladat során azt az $\overrightarrow{s}_{\mathcal{X}}$ projektütemtervet kell meghatároznunk, ahol a költség- és erőforráskorlátokat figyelembe véve, egy minimális rendszermegbízhatóság-növekményt elérve a javító-megelőző tevékenységeket a lehető legrövidebb idő alatt tudjuk elvégezni. Mivel a gyakorlatban ez a feladat szokott a leggyakrabban előfordulni, hiszen a folyamatos működéshez a legfontosabb, hogy minél rövidebb idő alatt sikerüljön a karbantartási projektet végrehajtani, ezért a továbbiakban mi is ezzel a feladattal foglalkozunk. Ugyanakkor a bemutatott módszerünk alkalmas az adott korlátokat betartó legkisebb költséggel rendelkező, vagy éppen a legnagyobb rendszermegbízhatóság-növekményt elérő projektterv meghatározására is. Ekkor a feladatok a következőképpen írhatók le.

2. Probléma. (b) Megelőző karbantartás-tervezési probléma (Preventive Maintenance Project Scheduling Problem, PMPSP), **legkisebb költségű projektütemterv keresése**:

$$\begin{aligned} &\arg\min c(\overrightarrow{s}_{\mathcal{X}}) \\ &\text{feltéve, hogy} \\ t(\overrightarrow{s}_{\mathcal{X}}) &\leq C_t \\ \mathbf{r}_{\max}(\overrightarrow{s}_{\mathcal{X}}) &\leq \mathbf{C_r} \\ 1 \geq \Delta TSR(K, \overrightarrow{s}_{\mathcal{X}}) &\geq C_{\Delta TSR} \end{aligned} \tag{2}$$

3. Probléma. (c) Megelőző karbantartás-tervezési probléma (Preventive Maintenance Project Scheduling Problem, PMPSP), **legnagyobb rendszermegbízhatóság-növekménnyel járó projektütemterv keresése**:

$$\begin{aligned} &\arg\max \Delta TSR(K, \overrightarrow{s}_{\mathcal{X}}) \\ &\text{feltéve, hogy} \\ t(\overrightarrow{s}_{\mathcal{X}}) &\leq C_t \\ c(\overrightarrow{s}_{\mathcal{X}}) &\leq C_c \\ \mathbf{r}_{\max}(\overrightarrow{s}_{\mathcal{X}}) &\leq \mathbf{C_r} \end{aligned} \tag{3}$$

Lehetőség van továbbá ún. több célfüggvény szerint is megfelelő ún. Pareto-optimális megoldást találni. Ezzel azonban csak egy későbbi tanulmányunkban foglalkozunk részletesebben.

3.2. A mátrix alapú modell felépítése

A javasolt M^4 (Multi-domain Maintenance Management Matrix) mátrix modell összesen 7 részmátrixot (domaint) tartalmaz.

1. **Block domain (BD)**: egy $z \times z$ mátrix, ahol z a berendezéselemek számát adja meg. A részmátrix a megbízhatósági blokkdiagram (RBD) mátrixreprezentációja, ahol az átlók tartalmazzák a kritikussági, megbízhatósági vagy rendelkezésre állás értékeit. Legyen β_{ij} egy cellája **BD** részmátrixnak, ahol $i \neq j$ esetén $\beta_{ij} = 1$ jelenti a megbízhatósági diagramban két berendezéselem közötti kapcsolatot. $\beta_{ij} = 0$ pedig azt jelenti, hogy a két berendezéselem között megbízhatósági szempontból nincs kapcsolat. A diagonális értékek $0 \leq R(k_i) = r_i = \beta_{ii} < 1$ a berendezéselemek megbízhatóság-értékei lesznek.

2. **Equipment-task mapping domain (ED):** egy $z \times n$-es (átváltási) mátrix, ahol n a javító-megelőző tevékenységek, z pedig a berendezéselemek számát jelöli. **ED** (rész)mátrix egy elemét jelölje: ε_{ij} $(i = 1, 2, \ldots, z;$ $j = 1, 2, \ldots, n)$. Az $1 \geq \varepsilon_{ij} \geq 0$ egy a_j javító-megelőző tevékenység relatív hatását mutatja egy k_i berendezéselem megbízhatóságára. Egy berendezéselemhez több javító-megelőző tevékenységet is rendelhetünk, de egy javító-megelőző tevékenység csak egy berendezés-elemhez tartozhat.

3. **Increase of reliability domain (ID):** egy $m \times n$-es mátrix, ahol n a tevékenységek száma, míg m a lehetséges megvalósítási módok száma. Legyen $1 \geq \eta_{jw} \geq 0$ $(j = 1, 2, \ldots, n;\ w = 1, 2, \ldots, m)$ az **ID** mátrix egy cellája. Ekkor η_{jw} azt mutatja, hogy ha egy a_j javító-megelőző tevékenységet jw módon valósítunk meg, akkor az mennyivel növelheti a berendezéselemek megbízhatóságát. Ekkor egy k_i berendezéselemre vonatkozó megbízhatóságnövelés a következőképpen számítható: $\Delta R(k_i) = \sum_{j:=1}^{n} \varepsilon_{ij}\eta_{jw}$. Egy a_j tevékenység végrehajtásából eredő minimális, illetve maximális megbízhatóság növekedést jelölje: $\eta_j^{\max} = \max_w \eta_{jw}$, illetve $\eta_j^{\min} = \min_w \eta_{jw}$. Jelölje $\mathbf{\Delta R}_{\max} := \{\eta_1^{\max}, \eta_2^{\max}, \ldots, \eta_n^{\max}\}$, illetve $\mathbf{\Delta R}_{\min} := \{\eta_1^{\min}, \eta_2^{\min}, \ldots, \eta_n^{\min}\}$ a maximális megbízhatóság-növekmény vektorokat.

4. **Logic domain (LD):** egy $n \times n$ -es mátrix, ahol n a javító-megelőző tevékenységek száma; λ_{ij} $(i, j = 1, 2, \ldots, n)$ az **LD** (rész)mátrix egy cellája. A diagonális $(\lambda_{ii},\ i = 1, 2, \ldots, n)$ elemek reprezentálják tevékenység-előfordulásokat. 0 jelenti, ha nem valósítjuk meg a tevékenységet, $0,5$ jelöli a bizonytalan tevékenység-előfordulást, 1 pedig a biztos tevékenységmegvaló-

sítást. A diagonálison kívüli cellák ($\lambda_{ij}, i \neq j$) reprezentálják a tevékenységek közötti kapcsolatokat, ahol 0, jelöli a rákövetkezési kapcsolat hiányát, 1 a rákövetkezési kapcsolat meglétét. $0,5$ pedig a bizonytalan kapcsolatot mutatja. Ebben a modellben nem adunk további pontértékeket sem a tevékenység-előfordulásoknak, sem a kapcsolaterősségeknek.

5. **Time domain (TD):** egy $n \times m$ -es mátrix, ahol n a javító-megelőző tevékenységek számát, m pedig a végrehajtási módok számát jelöli. $\tau_{jw} \geq 0$ cellaérték jelöli a w módon végrehajtandó a_j tevékenység időigényét. Jelölje egy a_j tevékenység maximális, illetve minimális időigényét $\tau_j^{\max} = \max_w \tau_{jw}$, illetve $\tau_j^{\min} = \min_w \tau_{jw}$. Jelölje továbbá $\mathbf{t}_{\max} := [\tau_1^{\max}, \tau_2^{\max}, \ldots, \tau_n^{\max}]$, és $\mathbf{t}_{\min} := [\tau_1^{\min}, \tau_2^{\min}, \ldots, \tau_n^{\min}]$ a maximális/minimális időigényeket tartalmazó vektort.

6. **Cost domain (CD):** egy $n \times m$ -es mátrix, ahol n a javító-megelőző tevékenységek számát, m pedig a végrehajtási módok számát jelöli. $\zeta_{jw} \geq 0$ cellaérték jelöli a w módon végrehajtandó a_j tevékenység költségigényét. Jelölje egy a_j tevékenység maximális, illetve minimális költségigényét $\zeta_j^{\max} = \max_w \zeta_{jw}$, illetve $\zeta_j^{\min} = \min_w \zeta_{jw}$ Jelölje továbbá

$$\mathbf{c}_{\max} := [\zeta_1^{\max}, \zeta_2^{\max}, \ldots, \zeta_n^{\max}], \text{ és } \mathbf{c}_{\min} := [\zeta_1^{\min}, \zeta_2^{\min}, \ldots, \zeta_n^{\min}]$$

a maximális/minimális erőforrás-igényeket tartalmazó vektort.

7. **Resource domain (RD):** egy $n \times rm$ -es mátrix, ahol n a javító-megelőző tevékenységek, r a (megújuló) erőforrás-igények, m pedig a végrehajtási módok számát jelöli. A mátrix első m oszlopában az első, a második m oszlopában a második, az r-edik m oszlopában az r-edik erőforrásoknak a tevékenységek egyes végrehajtási módjaihoz tartozó igényeit jelöljük.

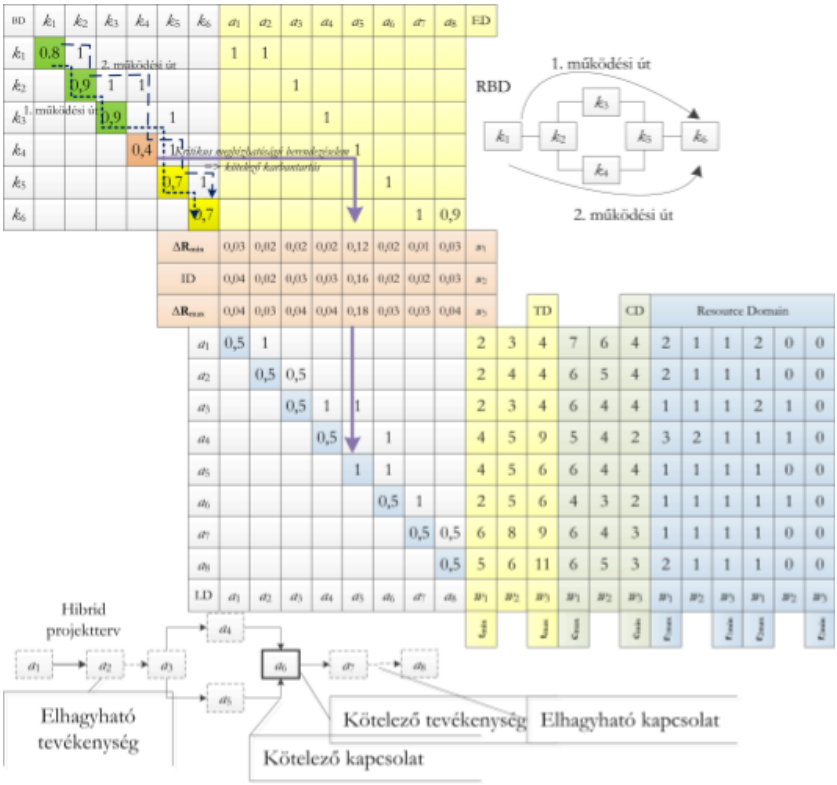
A hét részmátrix elhelyezkedését mutatja az 1. ábra egy példán keresztül feltételezve, hogy a kritikus beavatkozási érték valamennyi berendezéselemre $cr = 0,5$. Mivel az 1. ábrán a k_4 berendezés-elemre $R(k_4) = 0,4 < cr$, ezért a k_4 megbízhatóságát mindenképpen javítani kell. Ebből adódóan az a_5 tevékenységet mindenképpen el kell végeznünk, hogy a minimális $0,5$-ös megbízhatósági értéket elérjük.

Megbízhatósági szempontból a k_3 és k_4 berendezéselem párhuzamosan van csatolva egymással, e két elemmel pedig sorosan az összes többi rendszerelem. Az ábrán bejelölt 2 működési út adható meg. Ebből adódóan a rendszer megbízhatósága:

$$TSR = R(k_1)R(k_2)(1 - (1 - R(k_3))(1 - R(k_4)))R(k_5)R(k_6).$$

3.3. A javasolt algoritmus bemutatása

A javasolt algoritmus (melynek angol neve Preventive Maintenance Project Scheduling Algorithm, PMPSA, lásd: 6. ábrán látható pszeudo kódot) valamennyi



1. ábra. Megelőző karbantartás-tervezés mátrix alapú modellje

fázisában kihasználjuk, hogy az elérhető, idő-, költség-, erőforrás- és maximális megbízhatósági értéket anélkül kiszámíthatók, hogy az összes lehetséges projektváltozatot meg kellene határoznunk.

Legtöbb költségünk akkor keletkezik, ha valamennyi javító-megelőző tevékenységet végrehajtjuk, és ezek közül is a legköltségesebb alternatívát választjuk ($C_{\max}$). Legkevesebb költségű projektváltozatot az a projektterv adja, ahol csak a kötelező tevékenységek szerepelnek a projekttervben és itt is a legkevésbé költséges alternatívát választjuk ($C_{\min}$).

Nagyon hasonlóan a költségekhez, amennyiben a tevékenységek várható hatását meg tudjuk határozni, akkor becsülhető a maximális megbízhatóság-javulás ($\Delta TSR_{\max}$). Ezt az értéket itt is akkor érhetjük el, ha valamennyi javító-megelőző tevékenységet végrehajtjuk. Ha csak a kötelezőkre szorítkozunk, akkor a minimális megbízhatóság-javulást ($\Delta TSR_{\min}$) fogja jelölni.

Az időszükséglet számításához a fentieken kívül a tevékenységek közötti kapcsolatokat is figyelembe kell vennünk. A legrövidebb projekttervet akkor kapjuk, ha valamennyi bizonytalan tevékenységet későbbi projektbe ütemezzük át, vagyis ebből a projektből elhagyjuk és valamennyi, technológiailag nem kötelező kapcsolatot feloldunk és a tevékenységek végrehajtását párhuzamosítjuk ($T_{\min}$). Ezzel szemben a leghosszabb átfutási időt ($T_{\max}$) valamennyi tevékenység végrehajtása és valamennyi tevékenységkapcsolat betartása fogja eredményezni.

Ha a tevékenységeket a legkorábbi időpontra ütemezzük, akkor az erőforrásigények maxi-mumát ($\mathbf{r}_{\max}$) akkor kapjuk, amikor valamennyi bizonytalan tevékenységet végrehajtjuk, de valamennyi bizonytalan kapcsolatot feloldjuk (párhuzamos végrehajtás) és a megvalósítási módok közül azt választjuk, ahol az erőforrás-igény maximális. Ugyanígy a legkevesebb erőforrás-igény ($\mathbf{r}_{\min}$) akkor keletkezik, ha valamennyi bizonytalan tevékenység-előfordulást elhagyjuk ebből a projektből (pontosabban átütemezzük egy későbbi, másik projektbe), a kötelező tevékenység-előfordulások kapcsolatait azonban meghagyjuk (soros végrehajtás), és az erőforrás-igények közül a minimálisakkal számolunk.

Az első és a második fázisban mindig két lehetséges alternatíva közül választunk, nevezetesen az első fázisban: megvalósítjuk, vagy elhagyjuk (átütemezzük) a tevékenységet, illetve a második fázisban: előírjuk, vagy feloldjuk két tevékenység között a kapcsolatot. Ebből adódóan a döntési fánk, amit be kell járnunk, egy speciális bináris fa lesz. Valamennyi döntési ág esetén ki tudjuk számítani a lehetséges legkisebb, illetve legnagyobb idő-, költség-, erőforrásigényeket, illetve a minimális, maximális rendszermegbízhatóság-növekményt, feltéve, hogy a tevékenységeket megvalósítjuk, vagy elhagyjuk. Ekkor tehát a döntési fánk egy ún. bináris kupac lesz (binary heap). A fa tetején meg tudjuk mondani, hogy mi az a legkisebb idő-, költség-, erőforrásigény, amelyet biztosan igényel bármely projektváltozat. Egy tevékenység megvalósításáról, vagy elhagyásáról döntve szintén ki tudjuk számolni a legkisebb idő-, költség-, illetve erőforrásigényt, vagy éppen a maximális rendszermegbízhatóság-növekményt.

Nagyon fontos, hogy minden döntés után is egy M4 mátrixreprezentációt kapunk, viszont az első fázisban minden lépésben eggyel csökken a bizonytalan tevékenységek, a másodikban a bizonytalan kapcsolatok száma. Őket a döntésünknek megfelelően vagy elhagyjuk, vagy előírjuk.

A javasolt módszer a projektváltozatok kiértékelése során azokat az alternatívákat részesíti előnyben, amelyek célfüggvény értéke kedvezőbb.

Mivel minden döntés után meg tudjuk mondani az elérhető legnagyobb rendszermegbízhatóság-növekményt, illetve minimális idő- és költségigényt, a következő vágási szabályokat határozhatjuk meg az első két fázisra.

1. Szabály. [TSR_CUT] Ha egy projektváltozatra számolt maximális rendszermegbízhatóság-növekmény kisebb, mint a korlátként támasztott előírt megbízhatóság növekmény: $\Delta TSR_{\max} < C_{\Delta TSR}$, akkor sem az adott,sem az ebből

származtatható projektváltozatok, sem pedig a projektváltozatból származtatható projektstruktúrák nem megengedettek.

2. Szabály. [TPC_CUT] Ha egy projektváltozat minimális költségigénye magasabb, mint a költségkorlát: $C_{\min} > C_c$, akkor sem az adott, sem az ebből származtatható projektváltozatok, sem pedig a projektváltozatból származtatható projektstruktúrák nem megengedettek.

3. Szabály. [TPT_CUT] Ha egy projektváltozat vagy egy projektstruktúra minimális időigénye magasabb, mint az időkorlát: $T_{\min} > C_t$, akkor sem az adott projektváltozat/projektstruktúra, sem az ebből származtatható projekttervek nem megengedettek.

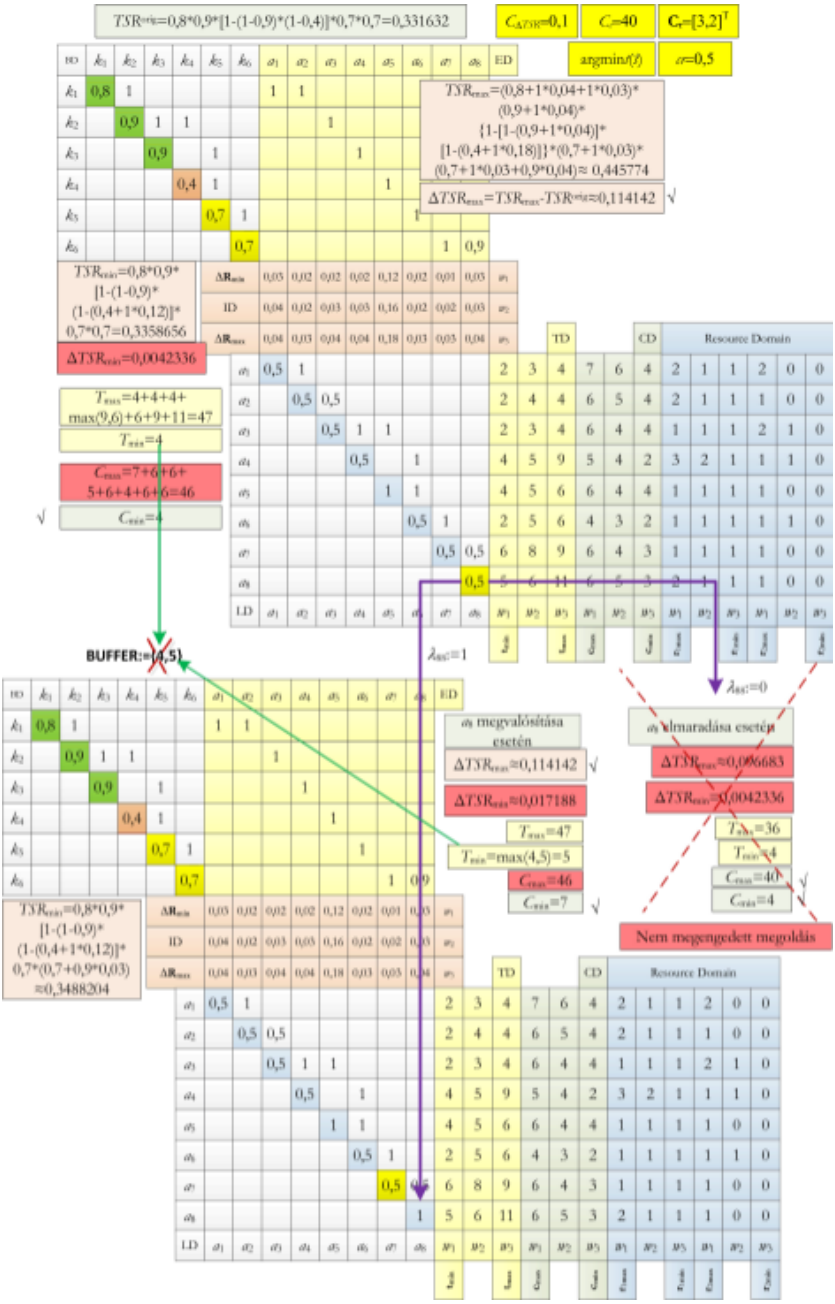
Erőforrásokra azért nem határozunk meg ilyen vágási szabályt, mert itt erőforrás-simító eljárást a harmadik fázisban fogunk végezni. A legkorábbi időpontra ütemezett tevékenységek esetében a maximális erőforrásigényen mind egy kiegyenlítési, mind pedig erőforrás-simítási algoritmussal csökkenteni lehet. Ennek számítási igénye ugyanakkor sokkal magasabb, mint pl. az ütemezésé. A módszer első fázisának szemléltetésére tekintsük az alábbi példát.

3.1. Példa. Legyen adott a megelőző karbantartási terv mátrixos ütemterve (2. ábra). A feladat, hogy a karbantartási projektet a legkorábbi időpontban végezzük el úgy, hogy a rendszer megbízhatósága minimálisan 10%-ot növekedjen ($C_{\Delta TSR} = 0,1$), de a költségigényként támasztott 40 e $-t ne lépjük túl ($C_c = 40$). A feladatot 3 karbantartóval és 2 gépbeállítóval kell elvégeztetnünk ($\mathbf{C_r} = [3,2]^T$). Feltétel. hogy a megbízhatósági diagramtól függetlenül minden berendezéselemnek legalább 0,5-ös megbízhatósági értéket el kell érnie ($cr = 0,5$).

Mivel a célfüggvény a lehető legrövidebb projektátfutási idő megtalálása, így az első fázisban azokat a projektváltozatokat részesítjük előnyben, amelyek kevesebb tevékenységet tartalmaznak. Tehát minden lépésben, ha a tevékenység a kritikus úton van, akkor igyekszünk a tevékenységet elhagyni a projektből, mert ekkor kapunk rövidebb átfutási időt (lásd: 2. ábra); ugyanakkor, ha az így kapott mátrixra számolt legnagyobb rendszermegbízhatóság-növekménnyel járó projektváltozat megvalósítása sem tudja garantálni a 10%-os megbízhatóság-növekményt, akkor ezt az ágat, illetve valamennyi alágát ki kell vennünk a döntési fából. Vagyis a 2. ábra példáján a_8 tevékenység megvalósítása mellett döntünk.

Bár az algoritmus során a legjobb megoldás megtalálására törekszünk, szükség lehet a második, vagy épp harmadik legjobb megoldás megtalálására. Éppen ezért a lehetséges legrövidebb átfutási időket egy rendezett **BUFFER** halmazban tároljuk.

A második fázisban arról döntünk, hogy a tevékenységeket milyen sorrendben hajtjuk majd végre. Fontos megjegyezni, hogy mivel már döntöttünk arról,



2. ábra. Az első fázis első lépése

hogy egy tevékenységet végrehajtunk-e vagy sem, ezért azok minimális és maximális költségvonzatát, illetve a rendszer megbízhatóságának minimális és maximális növekményét a tevékenységek végrehajtási sorrendje (a javasolt modellünkben) nem befolyásolja. Éppen ezért elegendő csak a minimális és maximális lehetséges tevékenységidőket kiszámolni. Ehhez pedig elegendő a logikai tervet és az időigényeket tartalmazó részmátrix (lásd: 3. ábra) megadása.

A célfüggvénynek legmegfelelőbb projektstruktúra meghatározásakor csak a 3. vágási szabályt alkalmazhatjuk. A második fázis végére már egy projektstruktúrát kapunk, ahol a cél a célfüggvénynek leginkább megfelelő, a korlátokat nem túllépő projektütemterv meghatározása. Ebben a fázisban már az ismert diszkrét idő-minőség-költség átváltási problémát kell megoldanunk, mely sajnos NP-nehéz feladat [9]. Ugyanakkor, ha nincs túl sok lehetséges alternatívánk, akkor az első két fázisban ismertetett gondolatmenetet továbbvihetjük, nevezetesen: minden lépésben döntünk arról, hogy egy adott tevékenységet a lehetséges módok közül melyikkel valósítjuk meg. A döntési fában minden csúcsnak (kivéve az utolsó (n-edik szinten lévőket)) m gyermeke van, hiszen minden tevékenységet m-féleképpen oldhatunk meg.

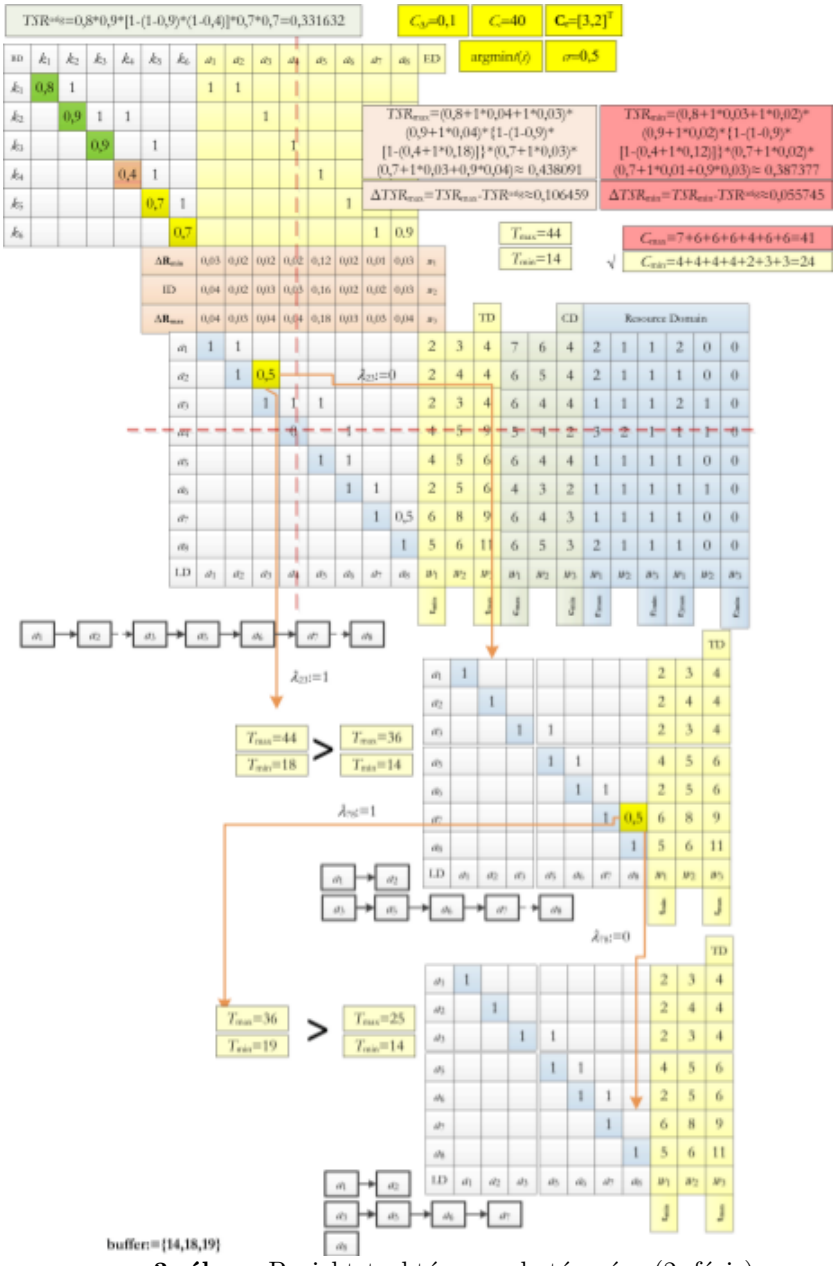
Minden pontban kiszámítjuk az elérhető legkisebb/legnagyobb átfutási idejű, költségigényű, valamint a legkisebb/legnagyobb rendszermegbízhatóság-növekményt eredményező projektterveket.

Az 1., 2., 3. vágási szabályokat alkalmazva egy adott célfüggvénynek leginkább megfelelő projektütemtervet kapunk.

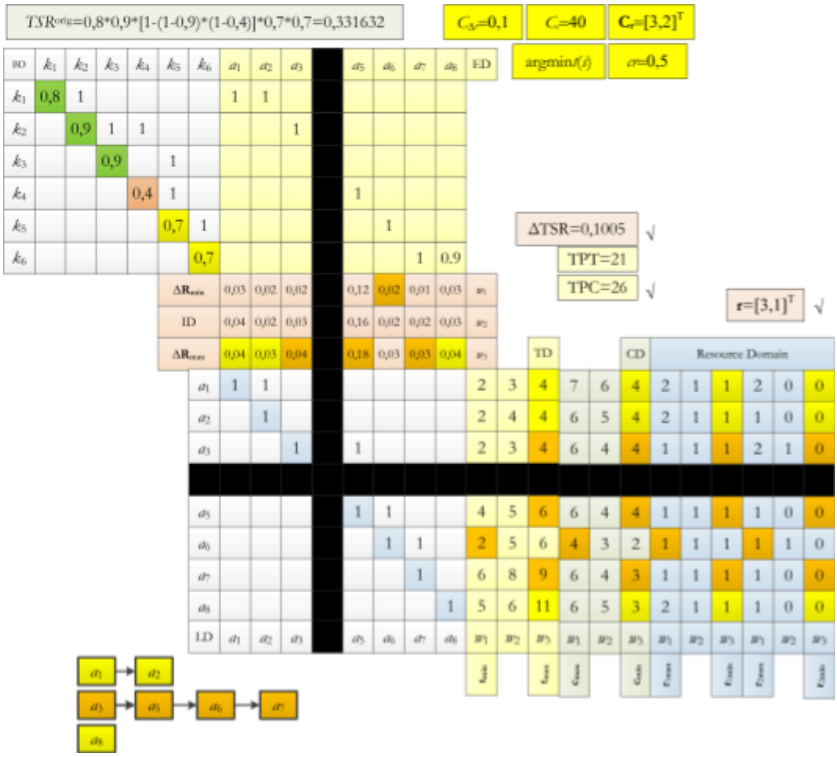
Az eredményül kapott ütemtervre végzünk el egy erőforrás-simítást. Ha eredményül egy, a korlátokat nem túllépő megoldást kapunk (jelen esetben ez $\mathbf{r} = [3, 1]^T$, amely nem lépi túl a $\mathbf{C_r} = [3, 2]^T$ korlátot), akkor ezt a projekttervet javasolhatjuk a karbantartók számára (lásd: 4. ábra). Ha az erőforrás-tervezés nem vezet eredményre, akkor vissza kell lépnünk a következő legjobb megoldásra és ott kell elvégeznünk újra az erőforrás-simítást.

4. A javasolt algoritmus komplexitása

Az első két fázisban kihasználjuk, hogy mindig csak két lehetséges alternatíva közül kell választanunk, nevezetesen megvalósítjuk vagy elhagyjuk a bizonytalan tevékenységeket, illetve a második fázisban előírjuk vagy feloldjuk a bizonytalan kapcsolatokat. A döntési fa minden ágán meg tudjuk mondani, hogy mi a leghosszabb/legrövidebb átfutási idő; legkisebb/legnagyobb projektköltség; legkisebb/legnagyobb megbízhatóság-növekmény, anélkül, hogy valamennyi projektváltozatot, illetve az ezekből eredeztethető projektstruktúrát meghatároznánk. A vágási szabályok segítségével pedig azokat a projektváltozatokat, illetve az azokból származtatható projektstruktúrákat ki sem kell értékelni, amelyek egy minimális (idő-, költség-, megbízhatóság-növekmény-) korlátot nem elégítenek ki.



3. ábra. Projektstruktúra meghatározása (2. fázis)



4. ábra. Optimális ütemterv meghatározása (3. fázis)

Az első két fázisban anélkül tudunk eljutni egy projektstruktúráig, hogy a döntési fában vissza kellene lépni. Ha a bizonytalan tevékenységek száma u, a bizonytalan kapcsolatok száma pedig v, akkor $O(u+v)$ lépésben kaphatunk egy projektstruktúrát. A harmadik fázis diszkrét átváltási probléma (lásd: De és mtsai [9]) és a legtöbb erőforrás-tervezési feladat (lásd: Brucker és mtsai [5]) már igazoltan NP-nehéz feladatok, ahol a javasolt módszerek mellett számos heurisztikus módszert is alkalmazhatunk [5].

5. Összefoglalás

A javasolt modell általánosítja a hagyományos idő-minőség-költség átváltási problémát, kiterjesztve azt bizonytalan tevékenység-előfordulások és bizonytalan kapcsolatok kezelésének lehetőségével. Egy $O(u+v)$ algoritmus (ahol u a bizony-

talan tevékenységek, v pedig a bizonytalan kapcsolatok száma) vezeti vissza a javasolt karbantartás-tervezési problémát egy hagyományos átváltási problémára. Ugyanakkor meg kell jegyezni, hogy a karbantartás-tervezés a rendszer megbízhatóságáról egy statikus állapotot feltételez, holott az akár a végrehajtás ideje alatt is változik, változhat. Egy előrejelző rendszer és a javasolt módszer kombinálása már nem csak a megelőző (preventive), hanem az ún. prediktív karbantartás-tervezés eszköztárát is gazdagíthatná. Ennek bemutatására ugyanakkor majd csak egy következő tanulmányban kerülhet sor.

6. Köszönetnyilvánítás

A kutatás a Új Nemzeti Kiválóság Program támogatásával készült.

Köszönetünket fejezzük ki továbbá az MTA-PE Regionális Innovációs és Fejlődéstani Hálózati Kutatócsoport kollégáinak cikkünkhöz adott jobbító javaslataikért.

A. Melléklet

```
function [BD,R]:=sp(BD,R)
  i:=1 //Soros elemek összevonása
  while i<|R| :
   l:=false
   if numel(find(BD(i,:)=1))=1 then: //Soros elemek keresése
    j:=find(BD(i,:}=1} //j lesz i elemmel sorosan összekötött berendezéselem
    if numel(find(BD[:,j)=1)]=1 //Ha i és j elem között csak egyreláció van...
     l:=true          //... akkor a soros elemeket összevonjuk.
     R(i]:=R(i)*R(j]; //Az összevont elem megbizhatósága a soros elemek megb. szorzata
     [BD,R]:=sp(BD({1..|R|}\{i},{1..|R|}\{j}),R({1..|R|}\{j}] //i-edik sor és j-edik
    end                                                       //oszlop elhagyása a mátrixból
   end
   if l=false then:
    i:=i+1      //Ellenkező esetben nincs összevonás
   end
  end
  i:=1 //Párhuzamos elemek összevonása
  while i<|R| :
   l:=false
   if numel(find(BD(i,:)=1))>=2 then: //Párhuzamos elemek keresése
    IJ:=find(BD(i,:)=1} : IJ=IJ{1..2) //Csak két párhuzamos elemet vonunk össze
    if numel(find[BD{IJ,:]=1}]=2 then:
     l:=true
     R(IJ{1}]=1-(1-R(IJ{1}])*(1-R(IJ{2}]) //Megbizhatóság kiszámitása
     [BD,R]:=sp(BD{{1..|R|}\{IJ{1}},{1..|R|}\{IJ{2}}},R{{1..|R|}\{IJ{2}}}
    end
   end
   if l=false
    i:=i+1
   end
  end
return [BD,R]

function [BD,R]:=DCM(BD,R)
 for i:=2 to |R| : //Üres oszlop eltávolitása
  if BD{:,i}=0 then: [BD,R]:=rekeysp_decomp(BD({1..|R|}\i,{1..|R|}\i),R({1..|R|}\i))
 for i:=2 to |R| : //Üres sor eltávolitása
  if BD{i,:}=0 then: [BD,R]:=rekeysp_decomp(BD({1..|R|}\i,{1..|R|}\i),R({1..|R|}\i))
 [BD,R]:=sp(BD,R)
 key:=argmax(sum(BD,1),sum(BD,2)} //Maximális fokszámú kulcselem megtalálása
 [BD,R]:=rekeysp_decomp(BD({1..|R|}\key,{1..|R|}\key},R({1..|R|}\key}])//~Eltávolitása

function tsr:=TSR(BD,R)
 [BD,tsr]:=DCM(BD,R)
end
```

5. ábra. Pszeudo kód. Rendszermegbízhatóság kiszámítása soros-párhuzamos összevonással és dekompozíciós módszerekkel

```
function M4s:=minTPT_ProjStruct(M4,M4s,duration,C_Δr,C_t,C_c,objDTCQTF):
  global buffer //minimális átfutási időket tartalmazó rendezett halmaz
  [i,j]:=impact(LD,TD)//Lásd 3. a pszeudokódot
   if i≠0 and j≠0 then:
      LD(i,j):=0 //Kapcsolat elhagyása
      mins_0:=TPT(|LD|,min(TD)) //t_min:=min(TD)
      maxs_0:=TPT(|LD|,max(TD)) //t_max:=max(TD)
      if maxs_0>=duration and mins_0<=duration then:
         M4s:=minTPT_ProjStruct M4,M4s,duration,C_Δr,C_t,C_c,objDTCQTF)
      else: //Átfutási idők eltárolása
         if mins_0∉buffer then: buffer:=buffer∪mins_0
         LD(i,j):=1 //Kapcsolat előírása
         mins_1:=TPT(|LD|,min(TD)) //t_min:=min(TD)
         maxs_1:=TPT(|LD|,max(TD)) //t_max:=max(TD)
         if maxs_1>=duration and mins_1<=duration then:
            M4s:=minTPT_ProjStruct M4,M4s,duration,C_Δr,C_t,C_c,objDTCQTF)
         else: //Átfutási idők eltárolása
            if mins_1∉buffer then: buffer:=buffer∪mins_1
   else: if [TD,CD,ID]=DTCQTF(M4,C_Δr,C_t,C_c,objDTCQTF) is feasible than : M4s:=M4s∪M4
return M4s //objTCTP a DTCQTF célfüggvénye

function M4s:=ProjScen(M4,M4s,SCORE,C_Δr,C_t,C_c):
  global BUFFER
  i:=argmax(max(TD)) //t_max:=max(TD)
  if i≠0 then:
    LD(i,i):=0 //Tevékenység elhagyása
    ΔR_min0:=TSR(BD,diag(BD)+ED*([diag(LD)].*Δr_max)')-TSR(BD,diag(BD))
    C_min0:=TPC([diag(LD)],min(CD)) //c_min:=min(CD)
    T_min0:=TPT(|LD|,min(TD)) //t_min:=min(TD)
    T_max0:=TPT(|LD|,max(TD)) //t_max:=max(TD)
    if T_max0>=SCORE and T_min0<=SCORE and C_min0<=C_c and T_min0<=C_t and ΔR_max0>=C_Δr then:
      M4s:=ProjScen(M4,M4s,SCORE,C_Δr,C_t,C_c)
    else: //Átfutási idők eltárolása
     if T_min0∉BUFFER then: BUFFER:=BUFFER∪T_min0
     LD(i,i):=1 //Tevékenység előírása
     ΔR_min1:= TSR(BD,diag(BD)+ED*([diag(LD)].*Δr_max)')-TSR(BD,diag(BD))
     C_min1:=TPC([diag(LD)],min(CD)) //c_min:=min(CD)
     T_min1:=TPT(|LD|,min(TD)) //t_min:=min(TD)
     T_max1:=TPT(|LD|,max(TD)) //t_max:=max(TD)
     if T_max1>=SCORE and T_min1<=SCORE and C_min1<=C_c and T_min1<=C_t and ΔR_max1>=C_Δr then:
         M4s:=ProjScen(M4,M4s,SCORE,C_Δr,C_t,C_c)
     else: // Átfutási idők eltárolása
         if T_min1∉BUFFER then: BUFFER:=BUFFER∪T_min1
   else: M4s:=M4s∪M4
return M4s

function M4s:=PMPSP(M4,C_Δr,C_t,C_c,objTCTP,n):
   global BUFFER,buffer
   BUFFER:=TPT(|LD|,min(TD)) //t_min:=min(TD)
   SCENARIOs:=∅ //Projektváltozatok halmaza
   M4s:=∅ //Projektstruktúrák halmaza
   i:=1
   structures:=0
   do
      SCENARIOs= SCENARIOs∪ProjScen(M4,SCENARIOs, BUFFER(i),C_Δr,C_t,C_c)
      SCENARIO=SCENARIOs(i)                    //Projekt változatok
      buffer:=TPT(|LD|, min(TD)) //t_min:=min(TD)
      j:=1
      do
         PDMs=PDMs∪minTPT_ProjStruct(SCENARIO,M4s,buffer(j),C_Δr,C_t,C_c,objDTCQTF)
         j:=j+1
         structures:=structures+1       //Projektstruktúra
      while j<=|buffer| and structures<=n:
      i:=i+1
   while i<=|BUFFER|:
return PDMs
```

6. ábra. Pszeudo kód. Javasolt algoritmus a karbantartás-tervezési probléma (PMPSP) megoldására

```
function tpt:=TPT(LD,t):
    n:=|t|
    EST:=0 //Earliest Start Time = Legkorábbi kezdés
    EFT:=t //Earliest Finish Time = Legkorábbi befejezés
    if LD(i,i)>0 then: //if task may be completed
      for i:=1 to (n-1):
         for j:=i+1 to n:
           if LD(i,j)>0 then: //Ha a tevékenységek között van kapcsolat
              if EFT(i)>EST(j) then:
                 EST(j):=EFT(i)
                 EFT(j):=EST(j)+t(j)
              end
           end
         end
      end
    end
return max(EFT)

function tpc:=TPC(TASKS,c): //TASKS: LD részmátrix diagonálisa c: költségvektor
    n:=|c|
    tpc:=0
    for i:=1 to n:
         if TASKS(i)==1 then:
            tpc:=tpc+c(i)
         end
    end
return tpc

function rMAX:=maxres(DSM,t,R)
 n:=|T|
 EST:=0 //EST: n elemű vektor null vector
 EFT:=t
 for i:=1 to (n-1) :
   for j:=(i+1) to n :
     if DSM(i,j)>0
          if EST(j)<EFT(i) then:
             EST(j):=EFT(i)
             EFT(j):=EST(j)+t(j)
          end
     end
   end
 end
 BP:=sort(union(EST,EFT)) //Az erőforrás-függvény töréspontjainak meghatározása
 b:=|BP|                  //Töréspontok száma
 r:=|R(1,:)|              //Erőforrások száma
 RESFUNC:=zeros(b,r)      //Erőforrásigények meghatározása
 for i:=1 to b :
  RESFUNC(i,:)=sum(R(find((EST<=BP(i))&(EFT>BP(i))),:),1)
 end
return max(RESFUNC)'

function [i,j]:=Impact(LD,TD)
  IMPACT:=0 : i:=0 : j=0 : n:=|LD(:,i)|
  for I:=1 to n-1:
    for J:=i+1 to n:
        dependency:=LD(I,J)
        if LD(I,J)<1 and LD(I,J)>0 then: //bizonytalan kapcsolat
           LD(I,J):=1
           t_maxi:=TPT(LD,max(TD)) //t_max:=max(TD)
           t_mini:=TPT(LD,min(TD)) //t_min:=min(TD)
           LD(I,J):=0
           t_max0:=TPT(LD,max(TD)) //t_max:=max(TD)
           t_min0:=TPT(LD,min(TD)) //t_min:=min(TD)
           if IMPACT>(t_maxi-t_max0) then:
             i:=I : j:=J : IMPACT:=(t_maxi-t_max0)
           end
           if IMPACT>(t_mini-t_min0) then:
             i:=I : j:=J : IMPACT:=(t_mini-t_min0)
           end
        end
    end
  end
return [i,j]
```

7. ábra. Pszeudo kód. Segédfüggvények: átfutási idő (TPT), költségigény (TPC), maximális erőforrásigény (maxres) számítása, csúcs kiválasztása (impact)

Hivatkozások

[1] Babu, A. and Suresh, N.: *Project management with time, cost, and quality considerations.* European Journal of Operational Research **88(2)** (1996): 320–327.

[2] Berman, E. B.: *Resource allocation in a pert network under continuous activity time-cost functions.* Management Science **10(4)** (1964): 734–745.

[3] Browning, T. and Eppinger, S.: *Modeling impacts of process architecture on cost and schedule risk in product development.* Engineering Management, IEEE Transactions on **49(4)** (2002): 428–442.

[4] Browning, T. R.: *Managing complex project process models with a process architecture framework.* International Journal of Project Management **32(2)** (2014): 229–241.

[5] Brucker, P., Drexl, A., Mohring, R., Neumann, K., and Pesch, E.: *Resource-constrained project scheduling: Notation, classification, models, and methods.* European Journal of Operational Research **112(1)** (1999): 3–41.

[6] Camara, A. W.: *An aumtomated pert/cpm production scheduling appilcation on the univac iii.* Technical report, DTIC Document (1968).

[7] Chen, C.-H., Ling, S. F., and Chen, W.: *Project scheduling for collaborative product development using {DSM}.* International Journal of Project Management **21(4)** (2003): 291–299.

[8] Danilovic, M. and Browning, T. R.: *Managing complex product development projects with design structure matrices and domain mapping matrices.* International Journal of Project Management **25(3)** (2007): 300–314.

[9] De, P., Dunne, E. J., Ghosh, J. B., and Wells, C. E.: *The discrete time-cost tradeoff problem revisited.* European Journal of Operational Research **81(2)** (1995): 225–238.

[10] De, P., Dunne, E. J., Ghosh, J. B., and Wells, C. E.: *Complexity of the discrete time-cost tradeoff problem for project networks.* Operations Research **45(2)** (1997): 302–306.

[11] Falk, J. E. and Horowitz, J. L.: *Critical path problems with concave cost-time curves.* Management Science **19**(4-part-1) (1972): 446–455.

[12] Feng, C., Liu, L., and Burns, S.: *Stochastic construction time-cost trade-off analysis.* Journal of Computing in Civil Engineering **14(2)** (2000): 117–126.

[13] Fulkerson, D. R.: *A network flow computation for project cost curves.* Management science **7(2)** (1961): 167–178.

[14] Gertsbakh, I.: *Reliability Theory - With Applications to Preventive Maintenance.* Springer Berlin (2000).

[15] Goldberg, A. and Tarjan, R.: *Solving minimum-cost flow problems by successive approximation.* In Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC '87,(1987), pages 7–18, New York, NY, USA. ACM.

[16] Goldberg, A. V.: *An efficient implementation of a scaling minimum-cost flow algorithm.* Journal of Algorithms **22(1)** (1997): 1–29.

[17] Idhammar, C.: *Preventive Maintenance, Essential Care and Condition Monitoring Book.* IDCON inc. (1999)

[18] Kelley, J. E.: *Critical-path planning and scheduling: Mathematical basis.* Operations Research **9(3)** (1961): 296–320.

[19] Kosztyán, Z.: *Mátrix alapú stratégiai projekttervezési eljárások.* SZIGMA **44(1–2)** (2013): 65–94.

[20] Kosztyán, Z., Fejes, J., and Kiss, J.: *Sztochasztikus hálóstruktúrák kezelése projektütemezési feladatokban.* SZIGMA **39(1–2)** (2008): 85–103.

[21] Kosztyán, Z. and Kiss, J.: *Pem–a new matrix method for supporting the logic planning of software development projects.* In DSM 2010: Proceedings of the 12th International DSM Conference, Cambridge, UK, 22–23.07. (2010)

[22] Kosztyán, Z. and Kiss, J.: *Stochastic network planning method.* In Elleithy, K., editor, Advanced Techniques in Computing Sciences and Software Engineering (2010), pages 263–268. Springer Netherlands.

[23] Kosztyán, Z. T.: *Exact algorithm for matrix-based project planning problems.* Expert Systems with Applications **42(9)** (2015): 4460–4473.

[24] Kovács, Z.: *Karbantartási stratégiák monte carlo optimalizálása.* Szigma **XXXIX.** (2008): 185–198.

[25] Kovács, Z. and Vítek, M.: *Rendszer-megbízhatóság számítása igazságtáblázat alkalmazásával.* Minőség és Megbízhatóság **4** (1991): 43–44.

[26] Lamberson, L. R. and Hocking, R. R.: *Optimum time compression in project scheduling.* Management Science **16(10)** (1970): B–597–B–606.

[27] Minogue, P. et al.: *"gantt-like" dsms.* In DSM 2011: Proceedings of the 13th International DSM Conference.(2011)

[28] Moubray, J.: *Reliability-Centered Maintenance.* Second edition. Industrial Press, Inc.; 2 Revised edition. (1997)

[29] Orlin, J. B.: *A faster strongly polynomial minimum cost flow algorithm.* Operations Research **41(2)** (1993): 338–350.

[30] Pritsker, A. A. B.: *GERT: Graphical evaluation and review technique.* Rand Corporation. (1966)

[31] Rahimi, M. and Iranmanesh, H.: *Multi objective particle swarm optimization for a discrete time, cost and quality trade-off problem.* World Applied Sciences Journal **4(2)** (2008): 270–276.

[32] Shiker, M.: *Some methods of calculating the reliability of mixed models.* Journal of Babylon University **21(3)** (2013): 770–774.

[33] Steward, D. V.: *The design structure system- a method for managing the design of complex systems.* IEEE transactions on Engineering Management **28(3)** (1981): 71–74.

[34] Sutrisno, A. and Lee, T.: *Service reliability assessment using failure mode and effect analysis (fmea): survey and opportunity roadmap.* International Journal of Engineering, Science and Technology **3(7)** (2011): 25–38.

[35] Tareghian, H. R. and Taheri, S. H.: *On the discrete time, cost and quality trade-off problem.* Applied Mathematics and Computation **181(2)** (2006): 1305–1312.

[36] Wang, Y.-M. and Elhag, T.: *Evidential reasoning approach for bridge condition assessment.* Expert Systems with Applications **34(1)** (2008): 689–699.

[37] Xiao, R., Chen, T., and Tao, Z.: *Information modeling and reengineering for product development process.* International Journal of Management Science and Engineering Management **2(1)** (2007): 64–74.

[38] Yan, H.-S., Wang, Z., and Jiang, M.: *A quantitative approach to the process modeling and planning in concurrent engineering.* Concurrent Engineering **10(2)** (2002): 97–111.

[39] Yassine, A., Falkenburg, D., and Chelst, K.: *Engineering design management: An information structure approach.* International Journal of Production Research **37(13)** (1999): 2957–2975.

(Beérkezett: 2016. január 28.)

KOSZTYÁN ZSOLT TIBOR
PRIBOJSZKI-NÉMETH ANIKÓ
KOVÁCS ZOLTÁN
Pannon Egyetem
Gazdaságtudományi Kar
Menedzsment Intézet
Kvantitatív Módszerek Intézeti Tanszék és Ellátási Lánc Menedzsment Intézeti Tanszék
8200 Veszprém, Egyetem utca 10.
{kzst,nemethani,kovacsz}@gtk.uni-pannon.hu

MATRIX-BASED MAINTENANCE MANAGEMENT

Zsolt Tibor Kosztyán, Anikó Pribojszki-Németh, Zoltán Kovács

In this paper a new problem, namely: *preventive maintenance project scheduling problem* is specified. This problem integrates the structures of system reliability and the sequences of the preventive maintenance tasks. Since in a preventive maintenance project scheduling problem usually not all equipment will be maintained, the problem specify a flexible project, which consist not only mandatory but also supplementary tasks. The proposed algorithm reduce the preventive maintenance project scheduling problem to a traditional discrete time-quality-cost trade-off problem within a polynomial time. The proposed algorithm is able to rank all the feasible project plans according to the predefined preferences of scores like time and cost. The developed matrix based methods and proposed exact algorithm may be important and essential components of a project expert system supporting strategic decision makings particularly in case of large, complex flexible maintenance projects.