Alkalmazott Matematikai Lapok **38** (2021), 105–126.
DOI: 10.37070/AML.2021.38.1.08

SZINERGIÁK HATÁSA A SZOFTVERFEJLESZTÉSI PROJEKTEKBEN

KOSZTYÁN ZSOLT TIBOR, SZALKAI ISTVÁN, KURBUCZ MARCELL TAMÁS

Az emberi erőforrások megfelelő elosztása a szoftverfejlesztési projektek egyik legfontosabb sikertényezője. Bár e kijelentésben jellemzően egyetértés uralkodik a szakirodalomban, a szoftverprojektek tervezését - ezáltal az erőforrás allokációs feladatot - támogató módszerek éppen a probléma „emberi" oldalát hagyják figyelmen kívül, amikor a projektcsapatokat egymástól független munkavállalók összességének tekintik. Jelen tanulmány célja kettős. Egyfelől kiegészíti a szoftverprojekt-ütemezési problémát (angolul: Software Project Scheduling Problem, röviden: SPSP) a munkavállalók közötti - azok teljesítményét befolyásoló - szinergiákkal, majd egy szimulált projektadatbázisból választva, a projektek optimalizálását követően megvizsgálja e szinergiák projektköltségekre gyakorolt hatását. Az eredmények alapján nem csak a munkavállalók közötti kölcsönös függőségek mértéke, hanem a függőségek által meghatározott szinergiahálózat jellemzői - így a hálózat topológiája, mérete és fokszám-központisága - is jelentősen befolyásolják a szoftverfejlesztési projektek költségeit.

1. Bevezetés

A szoftverprojekt-ütemezési probléma (angolul: Software Project Scheduling Problem, röviden: SPSP) [19, 8, 13] a szoftverfejlesztési és karbantartási folyamatokhoz kapcsolódó munkaerő-allokációs [3] és tevékenységütemezési [14] feladatokra vonatkozik. A probléma megoldása során a projekt költségét és időtartamát egyaránt minimalizálni szeretnénk, azonban ezek a célok ellentétben állnak egymással (lásd pl. [2]). Ez a többcélú természet még tovább bonyolítja a tervezést, a szoftverfejlesztési projektek egyre növekvő méretének eredményeként pedig szinte lehetetlenné teszi a kézi ütemezést [26].[1] Bár a témával kapcsolatos kutatások az utóbbi években egyre intenzívebbé váltak, a fent említett okból kifolyólag ezek többnyire a számítógéppel támogatott tervezés technikai fejlesztéseire irányultak (lásd pl. [6, 11, 19]).

[1] Az SPSP megoldása NP-nehéz feladat.

Coello és mtsai. [7] és Myszkowski és mtsai. [23] számos metaheurisztikát javasolt a probléma megoldására, míg [19] összehasonlította ezen algoritmusok pontosságát és skálázhatóságát.

Chicano és mtsai. [6] és Luna és mtsai. [19] eredményei alapján általános esetben a PAES (Pareto archived evolution strategy) [15] algoritmus rendelkezik a legjobb skálázhatósággal és segítségével érhetőek el a legjobb (megoldásokat tartalmazó) Pareto halmazok, a széles körben alkalmazott, ún. NSGA-II (non-dominated sorting genetic algorithm II) [10] és SPEA2 (strength Pareto evolutionary algorithm 2) [33] algoritmusok bizonyulnak a legpontatlanabbnak. Mindazonáltal magas költségvetésű, rövid projektek esetén a PAES-t az NSGA-II, SPEA2 és számos újabb algoritmus, pl. a MOCell (multi-objective cellular genetic algorithm) [24] is felülmúlta [19].

Míg az SPSP szakirodalma jellemzően ennek a többcélú optimalizálási problémának a megoldására fókuszál (lásd pl. [6] és [19]), kevés figyelem irányul a projektcsapatok összeállításának emberi szempontjaira, így a munkavállalók teljesítményének egymástól való kölcsönös függésére [25]. A munkavállalók kölcsönös függőségek vizsgálata a szociometria [22, 27] tudományterületéhez tartozik. Az itt elért eredmények alapján a társadalmi hálózatok központisági mutatói erősebb közvetlen előjelzői a teljesítménynek, mint az egyéni jellemzők (így pl. a funkcionális szerep, státusz vagy a kommunikációs szerep) [1], valamint a decentralizált csoportok jobban teljesítenek az összetett feladatok megoldása esetén, mint a központosított struktúrával rendelkező csoportok [28].

Jelen tanulmány célja kettős. Egyfelől kiegészíti a szoftverprojekt-ütemezési problémát a munkavállalók közötti - azok teljesítményét befolyásoló - szinergiákkal és a képességek mértékeinek (levels) figyelembe vételével, majd egy szimulált projektadatbázisból választva, a projektek optimalizálását követően megvizsgálja e szinergiák projektköltségekre gyakorolt hatását. Az eredmények alapján nem csak a munkavállalók közötti kölcsönös függőségek mértéke, hanem a függőségek által meghatározott szinergiahálózat jellemzői - így a hálózat topológiája, mérete és fokszám-központisága - is jelentősen befolyásolják a szoftverfejlesztési projektek költségeit.

A tanulmány az alábbi felépítést követi. A 2. fejezet bemutatja az SPSP eredeti, illetve a dolgozók szinergiájával kiegészített formáját. A 3. fejezet a probléma megoldására alkalmazott genetikus algoritmust ismerteti. A 4. fejezetben - szimulált projektadatok segítségével - megvizsgáljuk a munkavállalók szinergiájának SPSP-re gyakorolt hatását. Végezetül, tanulmányunk eredményeit az 5. fejezet foglalja össze.

2. A probléma formális leírása

Jelen fejezetben ismertetjük az SPSP-t, valamint annak szinergiákkal való kiterjesztését, az SSPSP-t[2]. A szakirodalomban fellelhető tanulmányokkal ellentétben - a könnyebb áttekinthetőség és rugalmasabb tervezés érdekében - a problémát mátrixos formában tárgyaljuk.

Ennek alapját az ún. multidomain mapping (MDM) módszer [9] nyújtja, melynek lényege a projekttervezési mátrixon belüli egyes tartományok interakciójában rejlik. Jelen tanulmányban a tevékenységek közötti eredetileg csak rugalmatlan kapcsolatokat és biztosan megvalósuló tevékenységeket kezelő módszert (lásd [9, 4]) kiegészítettük a rugalmas kapcsolatok és alacsonyabb prioritású, ún. nem kötelezően végrehajtandó tevékenységek kezelésével (lásd [16, 17, 18]), valamint a tervezési mátrixba egy új, szinergiákra vonatkozó tartományt vezettünk be. Az új projekttervezési mátrixra a továbbiakban **SMM** (synergy mapping model) mátrixként hivatkozunk (lásd 1. ábra). A rugalmas kapcsolatok és a relatív tevékenységprioritások bevezetésére azért van szükség, hogy a javasolt módszertan, a szoftverfejlesztés területén alkalmazott rugalmas - itt agilis, illetve hibrid - megközelítéseket is kezelni tudja, ami megengedi a tevékenységek más projektszakaszba történő átütemezését, illetve a projektstruktúra átrendezhetőségét.

2.1. Jelölések

A jelöléseket az alábbiakban foglaljuk össze:

- $E = \{e_1, \ldots, e_m\}$ a *munkavállalók* (employees) halmaza.
- $\mathbf{Y} \in \mathbb{R}^{m \times m}$ szimmetrikus, pozitív elemekből álló mátrix, a munkavállalók kölcsönös egymásra gyakorolt hatását (*szinergiákat*) tárolja: $\mathbf{Y}_{i,j} > 1$ pozitív (egymást erősítő), $\mathbf{Y}_{i,j} = 1$ semleges, míg $0 < \mathbf{Y}_{i,j} < 1$ negatív (egymást gyengítő) kölcsönhatást jelez e_i és e_j között ($\mathbf{Y}_{i,i} = 1$). $\mathbf{Y}$ az SMM mátrix *szinergia része* (synergy domain).
- Tetszőleges $\varepsilon \subseteq E$ részhalmaz és $|\varepsilon| \leq 1$ esetén:

$$\overline{Y}_{\varepsilon} := \sqrt[\eta]{\prod_{i,j \in \varepsilon} \prod_{i<j} [\mathbf{Y}]_{i,j}}, \qquad \text{ahol } \eta = \frac{|\varepsilon| \cdot (|\varepsilon| - 1)}{2} \tag{1}$$

  az ε munkavállalók szinergiáinak *mértani közepe* (és $\overline{Y}_{\varepsilon} = 1$, ha $|\varepsilon| = 1$).
- $S = \{\sigma_1, \ldots, \sigma_s\}$ a *képességek* (skills) halmaza.
- Az e_i munkavállaló képességeinek halmazát $S(e_i) := \left\{\sigma_1^{(i)}, \ldots, \sigma_{\rho_i}^{(i)}\right\} \subseteq S$ jelöli.

[2]Synergy-based Software Project Scheduling Problem

– Az $\ell(e_i, \sigma_k)$ nemnegatív valós szám jelöli az e_i munkavállaló σ_k képességben meglévő *szintjét* (level), $(1 \leq i \leq m,\ 1 \leq k \leq s)$. Nyilván $\sigma_k \in S(e_i) \iff 0 < \ell(e_i, \sigma_k)$. Ezeket a szinteket *összeadhatónak* tételezzük fel, vagyis e_{i_1} és e_{i_2} együttes munkája esetén a teljesítmény:

$$\mathbf{Y}_{i_1,i_2} \cdot (\ell(e_{i_1}, \sigma_k) + \ell(e_{i_2}, \sigma_k)). \tag{2}$$

Kettőnél több munkavállaló esetén csak közelítő képletet használhatunk[3] $(\varepsilon \subseteq E)$:

$$\ell(\varepsilon, \sigma_k) := \overline{Y}_{\varepsilon} \cdot \sum_{i \in \varepsilon} \ell(e_i, \sigma_k). \tag{3}$$

Az $\ell(e_i, \sigma_k)$ számokat az $\mathbf{S} \in \mathbb{R}^{m \times s}$ mátrixban tároljuk: $\mathbf{S}_{i,k} := \ell(e_i, \sigma_k)$, $\mathbf{S}$ az SMM mátrix *képességeket megjelenítő része* (skill domain).

– $A = \{a_1, \ldots, a_n\}$ az elvégzendő *feladatok* (tevékenységek, tasks) halmaza. Ezen belül $A^c \subseteq A$ a *kötelező* (mandatory), míg $A^- := A \setminus A^c$ a *kiegészítő, nem kötelező* (supplementary) feladatokat jelöli. A projekt tervezésekor az algoritmus automatikusan fogja eldönteni, hogy mely kiegészítő feladatokat teljesítsük.[4] Ezt a halmazt $A^{c(O)}$ jelöli. Nyilván a kötelező feladatok esetén nincs választásunk: $A^c \subseteq A^{c(O)}$ szükséges.

– A feladatok között három-féle *kapcsolat* (reláció, dependency) kerülhet meghatározásra: minden $i, j \leq n$, $i \neq j$ indexre: $a_i \prec a_j$ *szigorú* (strict) kapcsolat: a_j csak a_i befejezése után kezdődhet, $a_i \sim a_j$ *semleges* (no) kapcsolat: a_j és a_i elvégzése nem befolyásolják egymást, $a_i \bowtie a_j$ *rugalmas* (bizonytalan, uncertain or flexible) kapcsolat: az *algoritmus* dönti el, hogy az $a_i \bowtie a_j$ relációt $a_i \prec a_j$ vagy $a_j \prec a_i$ vagy $a_i \sim a_j$ kapcsolattá változtatja. Nyilván $\prec$ egy részbenrendezés, ami köröket[5] nem tartalmaz, $\bowtie$ és $\sim$ szimmetrikusak. Feltehető, hogy $a_i \prec a_j$ esetén $i < j$.

– Az $\mathbf{A} \in \mathbb{R}^{n \times n}$ mátrixban tároljuk a feladatokat és kapcsolataikat: $\mathbf{A}_{i,i} = 1 \iff a_i$ kötelező, $0 < \mathbf{A}_{i,i} < 1 \iff a_i$ kiegészítő ($\mathbf{A}_{i,i}$ *értéke* az a_i feladat pontértékét vagy (relatív) prioritását jelentheti), $\mathbf{A}_{i,j} = 1 \iff a_i \prec a_j$, $\mathbf{A}_{i,j} = 0 \iff a_i \sim a_j$, $0 < \mathbf{A}_{i,j} < 1 \iff a_i \bowtie a_j$ ($\mathbf{A}_{i,j}$ *értéke*, feladattól és a korábbi projekttapasztalatok felhasználásától függően az $a_i \bowtie a_j$ kapcsolat pontértékét, prioritását vagy a valószínűségét jelölheti). $\mathbf{A}$ az SMM mátrix *logikai része* (logical domain). Az *algoritmusnak* az $\mathbf{A}$ mátrix összes 0 és 1 közötti elemét 0-ra vagy 1-re kell változtatnia (többit változatlanul hagyva), a *végső* mátrixot $\mathbf{A}(\mathbf{O})$-val jelöljük ($\mathbf{A}(\mathbf{O}) \in \mathbb{R}^{n \times n}$).

– Az a_j tevékenységhez szükséges képességek halmazát $S(a_j) :=$

[3]Bár ezt a formulát később az algoritmus által előállított $\mathbf{O}$ hozzárendelési mátrix módosítani fogja, lásd alább az $\mathbf{M}$ mátrix után.

[4]Pl. az előírt korlátozó feltételek, vagy a célfüggvény optimalizálása miatt.

[5]Mint pl. $a_1 \prec a_2 \prec \ldots \prec a_1$

$\left\{\sigma_1^{(j)}, \ldots, \sigma_{\rho_j}^{(j)}\right\} \subseteq S$ jelöli, pontosabban: $L(a_j, \sigma_k)$ „egység" kell legalább „a_j-hez σ_k-ból". Nyilván $L(a_j, \sigma_k) \in \mathbb{R}_0^+$, $\sigma_k \in S(a_j) \iff 0 < L(a_j, \sigma_k)$ és $L(a_j, \sigma_k) \leq \ell(\varepsilon_j, \sigma_k)$ szükséges (az $\varepsilon_j \subseteq E$ részhalmazt is az algoritmus választja).

– A $\mathbf{W} \in \mathbb{R}^{s \times n}$ mátrixban tároljuk L értékeit: $\mathbf{W}_{j,k} := L(a_j, \sigma_k)$, $\mathbf{W}$ neve *munkatartalom mátrix* (skilled work domain), $w_{j,k}$ elemei *munkatartalmi elemek* (skilled work elements).

– Az $\mathbf{M} \in \mathbb{R}^{m \times n}$ mátrix $\mathbf{M}_{i,j}$ elemei adják meg, hogy az e_i munkavállaló munkaidejének *legfeljebb* mekkora *hányadát* töltheti az a_j feladat megoldásával, $0 \leq \mathbf{M}_{i,j} \leq 1$, $\mathbf{M}$ neve *párosítási rész* (matching domain).

– Az SSPSP probléma megoldásakor az algoritmus eldönti, hogy az e_i munkavállaló munkaidejének *ténylegesen* mekkora *hányadát* tölti az a_j feladat megoldásával, ezt az $\mathbf{O} \in \mathbb{R}^{m \times n}$ mátrix (output domain) $\mathbf{O}_{j,i}$ eleme adja meg, nyilván $0 \leq \mathbf{O}_{j,i} \leq \mathbf{M}_{i,j}$ és[6] $\sum_{j=1}^{n} \mathbf{O}_{j,i} \leq 1$.

– Az algoritmus által az a_j feladathoz rendelt összes emberi erőforrás:

$$A_j := \sum_{i=1}^{m} \mathbf{O}_{j,i}, \tag{4}$$

melyet az $\varepsilon_j := \{e_i \in E : 0 < \mathbf{O}_{j,i}\}$ munkavállalók alkotnak.

– A munkavállalók által elvégzett munka *időtartamát* (duration) $a_j^{dur}(\mathbf{O})$ illetve $\overline{a}_j^{dur}(\mathbf{O})$ jelöli: az a_j tevékenység elvégzéséhez szükséges idő (ami az erőforrások és a szinergiák függvénye), képletét (9) és (10) adja meg. $a_j^{start}(\mathbf{O})$ a kezdési időpont[7], $a_j^{end}(\mathbf{O}) = a_j^{start}(\mathbf{O}) + a_j^{dur}(\mathbf{O})$ a befejezés ideje (lásd még (11)). p_{dur} vagy $\mathbf{TPT}$ jelöli a *projekt időtartamát* (total project time), p_{cost} vagy $\mathbf{TPC}$ a *költségét* (total project cost).

– Mindegyik e_i munkavállaló *teljes* munkaidejének $e_i^w := \sum_{j=1}^{n} \mathbf{O}_{j,i}$ *részében* dolgozik a projekten (az algoritmus megoldása szerint), ami *legfeljebb* $e_i^{maxw} := \sum_{j=1}^{n} \mathbf{M}_{i,j}$ lehet, nyilván $0 \leq e_i^w \leq \min\{e_i^{maxw}, 1\}$. e_i^{salary} az e_i munkavállaló havi fizetése.

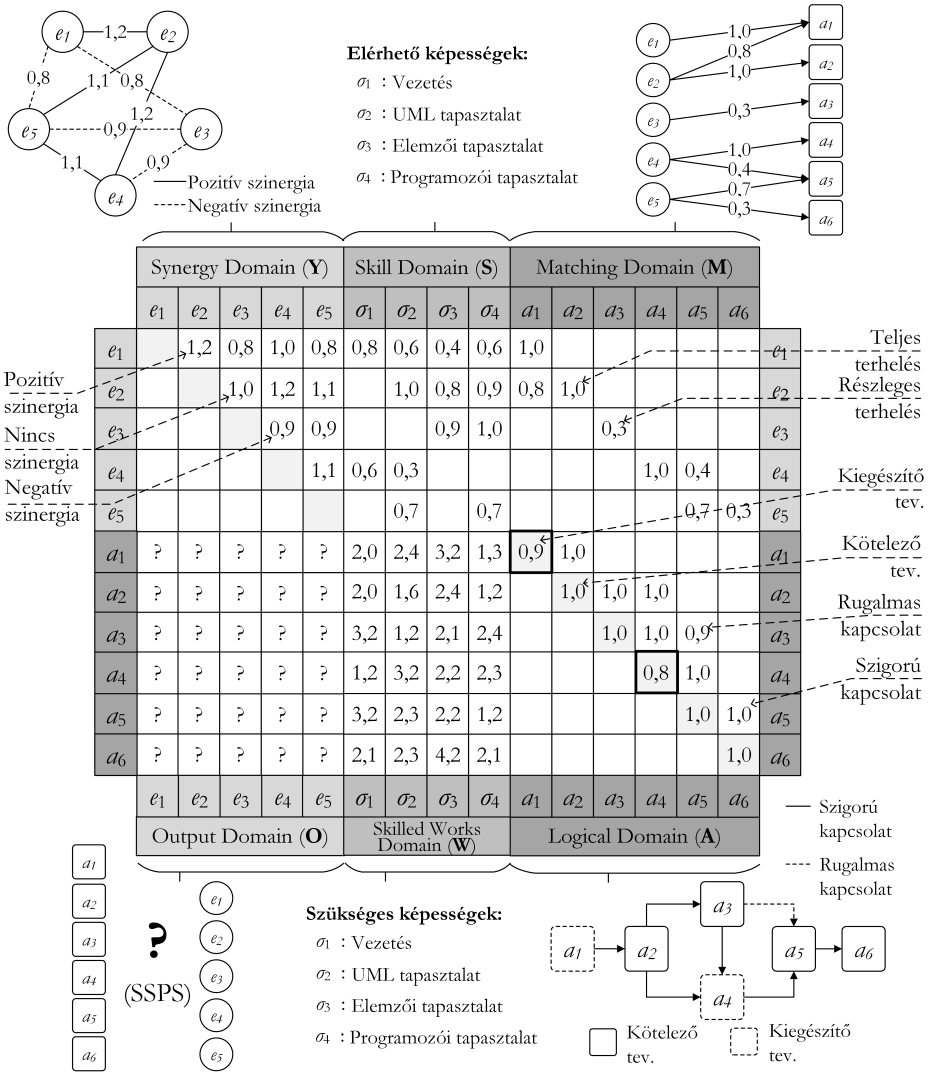
Az 1. ábrán a fenti mátrixok (adatok), mint a feladat részeit (domain) és azok kapcsolatait szemléltetjük.

[6]Persze $\sum_{j=1}^{n} [M]_{i,j} \leq 1$ *nincs* megkövetelve.

[7]$a_i \prec a_j$ esetén $a_i^{end}(\mathbf{O}) \leq a_j^{start}(\mathbf{O})$.

Mátrix alapú modellünk csak a meglévő és szükséges képességekkel foglalkozik. Fő célunk a munkavállalók és a feladatok egymáshoz rendelése (allocation).

A munkavállalót úgy kell a feladatokhoz rendelnünk, hogy azok együttes mennyisége ne haladja meg a munkavállaló munkaidejét, a (30)–(33) célfüggvények optimalizálásával.



1. ábra. SMM mátrix (Az Y részmátrixban csak a felső háromszöget jelöltük. A többi részmátrixban az üres cellák értéke alapértelmezés szerint 0.)

2.2. A projekt időtartama

Tegyük fel most, hogy az algoritmus már döntött az összes A^{-}-beli kiegészítő feladatot és összes $a_i \bowtie a_j$ rugalmas kapcsolatot illetően[8] valamint az e_i munkavállalók a_j feladatokhoz rendeléséről ($\mathbf{O}_{j,i}$). A továbbiakban egy a_j tevékenységet akkor hívunk elvégzendőnek, az a_i és a_j közötti kapcsolatot pedig szigorúnak ($a_i \prec a_j$), ha az algoritmus ezeket végül így határozta meg.

[2] feltételezése szerint a munkavállalók feladathoz rendelését a projektben rögzítettnek tekintjük (az algoritmus döntése után).

A képességek és szintjeik nem keverhetők össze, ezért minden σ_k képességre külön-külön ki kell számolnunk az ε_j csapat által, az a_j feladatban elvégzhető munkát[9] (szinergiák nélkül):

$$A_j^w(k) := \sum_{i=1}^{m} \left([\mathbf{S}]_{i,k} \cdot [\mathbf{O}]_{j,i}\right) = \sum_{i \in \varepsilon_j} \ell\left(e_i, \sigma_k\right) \cdot [\mathbf{O}]_{j,i}, \tag{5}$$

és szinergiákkal:

$$A_j^{w,adj}(k) := \overline{Y}_{\varepsilon_j} \cdot A_j^w(k) = \overline{Y}_{\varepsilon_j} \cdot \sum_{i=1}^{m} \left([\mathbf{S}]_{i,k} \cdot [\mathbf{O}]_{j,i}\right). \tag{6}$$

Mivel az a_j feladathoz a σ_k képességből $L\left(a_j, \sigma_k\right) = [\mathbf{W}]_{j,k}$ „mennyiség” kell, az ε_j csapatnak a_j elvégzéséhez szükséges idő (szinergiák nélkül):

$$a_{j,k}^{dur}(\mathbf{O}) = \frac{L\left(a_j, \sigma_k\right)}{A_j^w(k)} = \frac{[\mathbf{W}]_{j,k}}{\sum\limits_{i=1}^{m} \left([\mathbf{S}]_{i,k} \cdot [\mathbf{O}]_{j,i}\right)}, \tag{7}$$

míg a szinergiákkal módosított idő:

$$a_{j,k}^{dur,adj}(\mathbf{O}) = \frac{L\left(a_j, \sigma_k\right)}{A_j^{w,adj}(k)} = \frac{[\mathbf{W}]_{j,k}}{\overline{Y}_{\varepsilon_j} \cdot \sum\limits_{i=1}^{m} \left([\mathbf{S}]_{i,k} \cdot [\mathbf{O}]_{j,i}\right)}. \tag{8}$$

Feltételezzük, hogy mindegyik e_i a képességeit egyszerre használja[10], ezért a_j elvégzéséhez szükséges összidő (szinergiák nélkül):

$$a_j^{dur}(\mathbf{O}) = \max_{k \in S(a_j)} \left\{a_{j,k}^{dur}(\mathbf{O})\right\}, \tag{9}$$

[8] az input az $\mathbf{A}$ mátrixban, az algoritmus döntése az $\mathbf{A}(\mathbf{O})$ mátrixban található.

[9] az összegezést nyugodtan írhatjuk az összes i -re, hiszen $i \notin \varepsilon_j$ esetén $[\mathbf{O}]_{j,i} = 0$.

[10] és $S(a_j) \subseteqq \bigcup\limits_{i \in \varepsilon_j} S(e_i)$, mert $k \notin \bigcup\limits_{i \in \varepsilon_j} S(e_i)$ esetén (7) és (8) nevezői nullák. Ez (21) a C_2 feltételben.

illetve (szinergiákkal):

$$\overline{a}_j^{dur}(\mathbf{O}) := a_j^{dur,adj}(\mathbf{O}) = \max_{k \in S(a_j)} \left\{ a_{j,k}^{dur,adj}(\mathbf{O}) \right\}. \tag{10}$$

Tehát $a_j^{end}(\mathbf{O}) = a_j^{start}(\mathbf{O}) + \overline{a}_j^{dur}(\mathbf{O})$, ahol:

$$a_j^{start}(\mathbf{O}) \geq a_j^{start}(\mathbf{O})_{\min} := \begin{cases} 0 & \text{ha } \nexists\, a_i \in A,\, a_i \prec a_j \\ \max\{a_i^{end}(\mathbf{O}) : a_i \prec a_j\} & \text{máskor} \end{cases}. \tag{11}$$

A fenti képletekben feltételezzük a munkavállalók egyidejű (párhuzamos) munkavégzését az a_j feladatokban, és ne feledjük, hogy $\mathbf{S}$ és $\mathbf{O}$ az összes képességet és munkavállalót figyelembe veszi.

A *projekt* kezdő időpontja 0, (össz-) *időtartama*:

$$\mathrm{TPT} := p_{dur} = \max\{a_j^{end}(\mathbf{O}) : j = 1, \ldots, n\}. \tag{12}$$

Kiemeljük, hogy a (11) és (12) képletekkel meghatározott $a_j^{start}(\mathbf{O})$ és TPT értékek a legkorábbi kezdésekre ($a_j^{start}(\mathbf{O})_{\min}$) érvényesek, ugyanakkor az erőforrás-allokáció során fontos lehet, hogy az átfutási időt lehetőleg nem túllépve, a tevékenységek tényleges kezdését ($a_j^{start}(\mathbf{O})_{ALG}$)) eltoljuk, például azon okból, hogy az adott erőforrás-korlátot ne lépjük túl:

$$a_j^{start}(\mathbf{O})_{ALG} \geq a_j^{start}(\mathbf{O})_{\min}, \tag{13}$$

ahol $a_j^{start}(\mathbf{O})_{ALG}$ az a_j feladat *tényleges* kezdési ideje. Nyilván:

$$\overline{a}_j^{dur}(\mathbf{O})_{ALG} = \overline{a}_j^{dur}(\mathbf{O})_{\min}, \tag{14}$$

$$a_j^{end}(\mathbf{O})_{ALG} = a_j^{start}(\mathbf{O})_{ALG} + \overline{a}_j^{dur}(\mathbf{O})_{ALG}, \tag{15}$$

és

$$a_j^{start}(\mathbf{O})_{ALG} \geq \begin{cases} 0 & \text{ha } \nexists\, a_i \in A,\, a_i \prec a_j \\ \max\{a_i^{end}(\mathbf{O})_{ALG} : a_i \prec a_j\} & \text{máskor} \end{cases}. \tag{16}$$

A kezdési időpontok (valós számok egy) sorozatát:

$$\left(a_1^{start}(\mathbf{O})_{ALG}\ ,\ \ldots\ ,\ a_n^{start}(\mathbf{O})_{ALG}\right) \tag{17}$$

ütemezett kezdésidő sorozatnak (scheduled start time sequence, **SST**) nevezzük. A továbbiakban nem írjuk ki a *min* és *ALG* indexeket, de mindig az *ALG* értékekől beszélünk, ha mást nem jelzünk.

2.3. A projekt költsége

A projekt összköltsége (TPC, p_{cost}) a munkavállalók összesen, időarányosan kifizetett bére. Mivel pozitív szinergiák csökkentik, negatív szinergiák növelik az a_j^{dur} és $\overline{a}_j^{dur}$ időtartamokat, a költséget kiszámolhatjuk szinergiák nélkül és azokkal is:

$$\mathrm{TPC}_{syn} = \mathrm{TPC} := p_{cost} = \sum_{i=1}^{m}\sum_{j=1}^{n}(e_i^{salary} \times [\mathbf{O}]_{j,i} \times \overline{a}_j^{dur}(\mathbf{O}), \tag{18}$$

$$\mathrm{TPC}_{nosyn} := \sum_{i=1}^{m}\sum_{j=1}^{n}(e_i^{salary} \times [\mathbf{O}]_{j,i} \times a_j^{dur}(\mathbf{O})). \tag{19}$$

2.4. Feltételek

Az SSPSP projekt megtervezésekor több korlátozó feltételt is figyelembe *kell* vennünk.

$\mathbf{C}_1$: Mindegyik feladathoz legalább egy munkavállalót rendelnünk kell:

$$\varepsilon_j \neq \emptyset, \qquad ha \quad a_j \in A^{c(O)}. \tag{20}$$

$\mathbf{C}_2$: A munkavállalók képességei felhasználásával el kell végezni a feladatokban foglalt munkatartalmakat, azaz minden $a_j \in A^{c(O)}$ esetén:

$$S(a_j) \subseteq \bigcup_{\{i:\ 0<[\mathbf{O}]_{j,i}\}} S(e_i). \tag{21}$$

$\mathbf{C}_3$: A munkavállalók feladatokhoz való hozzárendelhetősége időben korlátos:

$$e_i^w(\tau) := \sum_{\{j\ \mid\ a_j^{start}\leq\tau\leq a_j^{end},\ a_j\in A^{c(O)}\}} \mathbf{O}_{j,i} \leq e_i^{maxw} \tag{22}$$

minden $0 \leq \tau \leq p_{dur}$ időpontban.

A $\mathbf{C}_4 - \mathbf{C}_6$ feltételekhez meg kell adnunk (megfelelő) C_s, C_p, C_c, C_t és K^w korlátokat (valós számokat), valamint további definíciókra és jelölésekre van szükségünk.[11] és $S(a_j) \subseteqq \bigcup_{i\in\varepsilon_j} S(e_i)$, mert $k \notin \bigcup_{i\in\varepsilon_j} S(e_i)$ esetén (7) és (8) nevezői nullák. Ez (21) a C_2 feltételben.

[11]Ne feledjük, hogy a bemeneti adatokat A^c és $\mathbf{A}$ tartalmazza, az algoritmus végeredménye pedig $A^{c(O)}$ és $\mathbf{A}(\mathbf{O})$-ban található. Természetesen $A^c \subseteq A^{c(O)} \subseteq A$.

– Az $a_i \in A^{c(O)}$ megvalósított feladat *pontértéke* (score value) $\mathbb{S}_i := \mathbf{A}_{i,i}$, a kihagyott $a_i \in A \backslash A^{c(O)}$ feladaté pedig $\mathbb{S}_i := 1 - \mathbf{A}_{i,i}$ $(i = 1, 2, .., n)$.

– Az $a_i \bowtie a_j$ (input) kapcsolat $(a_i, a_j \in A^{c(O)})$ *pontértéke* $p_{i,j} := \mathbf{A}_{i,j}$ abban az esetben, ha az algoritmus ezt a kapcsolatot $a_i \prec a_j$ vagy $a_j \prec a_i$-re változtatja, és $p_{i,j} := 1 - \mathbf{A}_{i,j}$ akkor, ha $a_i \sim a_j$-re változtatja.

$\mathbf{C}_4$: A projekt *össz pontértéke* (total project score) legalább C_s (lásd még (32)):

$$\mathrm{TPS} := \sqrt[n]{\prod_{i=1}^{n} \mathbb{S}_i} \;\geq\; C_s. \tag{23}$$

$\mathbf{C}_5$: A *projektstruktúra pontértéke* legalább C_p:

$$\sum_{a_i, a_j \in A^{c(O)},\ i \neq j} p_{i,j} \geq C_p. \tag{24}$$

$\mathbf{C}_6$: Az *összesített* túlmunka nem lépheti át az adott K^w felső korlátot, egy adott $\epsilon^K > 0$ tűréshatárt megengedve. Legyen $0 \leq \tau \leq p_{dur}$ esetén:

$$overwork(\tau) := \begin{cases} \sum\limits_{i=1}^{m} e_i^w(\tau) - K^w & \text{ha } \sum\limits_{i=1}^{m} e_i^w(\tau) > K^w \\ 0 & \text{máskor} \end{cases}, \tag{25}$$

és összesítve:

$$p_{over} := \int_{\tau=0}^{\tau=p_{dur}} overwork(\tau)\ d\tau. \tag{26}$$

Tehát C_6 : az összesített túlóra ϵ^K -nál kisebb:

$$p_{over} < \epsilon^K. \tag{27}$$

$\mathbf{C}_7$: A projekt *összköltsége* (total project cost, TPC) legfeljebb C_c lehet:

$$\mathrm{TPC} := p_{cost} \leq C_c. \tag{28}$$

$\mathbf{C}_8$: A projekt *összideje* (total project time, TPT) legfeljebb C_t lehet:

$$\mathrm{TPT} := p_{dur} \leq C_t. \tag{29}$$

A $\mathrm{TPT}_{\min}$, $\mathrm{TPC}_{\min}$ és $\mathrm{TPS}_{\max}$ mennyiségek a következők. $\mathrm{TPT}_{\min}$-t úgy érhetjük el, hogy az összes kiegészítő feladatot elhagyjuk (azaz $A^{c(O)} := A^c$), a rugalmas kapcsolatokat ($\bowtie$) párhuzamosítjuk ($\sim$), és a lehető legtöbb munkavállalót a projekthez rendelünk ($\mathbf{O}_{j,i} = \mathbf{M}_{i,j}$). Hasonlóan $\mathrm{TPC}_{\min}$ eléréséhez $A^{c(O)} = A^c$ javasolt, míg $\mathrm{TPS}_{\max}$ pl. az $A^{c(O)} = A$ feltétel esetén teljesül.

Modellünk célfüggvénye összetett. Célunk (leegyszerűsítve) a *legnagyobb pontértékkel, lehető legkisebb átfutási idővel és projektköltséggel rendelkező projektterv* megtalálása:

$$\mathrm{TPT} \to \min, \tag{30}$$

$$\mathrm{TPC} \to \min, \tag{31}$$

$$\mathrm{TPS} \to \max. \tag{32}$$

A fenti optimalizálási problémákat az alábbi célfüggvényben egyesítjük:

$$z := 1 - \sqrt[3]{\left(\frac{C_t - \mathrm{TPT}}{C_t - \mathrm{TPT}_{\min}}\right) * \left(\frac{C_c - \mathrm{TPC}}{C_c - \mathrm{TPC}_{\min}}\right) * \left(\frac{\mathrm{TPS} - C_s}{\mathrm{TPS}_{\max} - C_s}\right)} \to \min \tag{33}$$

a $\mathbf{C}_1 - \mathbf{C}_8$ feltételek mellett, ahol C_s, C_p, C_c és C_t adott megfelelő konstansok.

Egyszerűsítés végett, az SPSP irodalomhoz hasonlóan, az emberi képességek (időbeli) állandóságát tételezzük fel. [5] munkában a képességek fejlődésével is számolnak, jelen modellünk szintén kiterjeszthető ilyen irányba is.

3. Az alkalmazott genetikus algoritmus bemutatása

Ahogyan már a szinergiákat, rugalmas kapcsolatokat és tevékenység-előfordulásokat figyelembe- nem vevő eredeti SPSP feladat is NP-nehéz feladat, a kiterjesztett SSPSP feladat is NP-nehéz. Amely nem jelenti azt, hogy ne lenne egzakt megoldása az eredeti problémának (lásd pl. [30]), ugyanakkor a feladat már 10-es nagyságrendű tevékenységszám esetén sem oldható meg belátható időn belül. Éppen ezért a kutatók figyelme a metaheurisztikus, ezen belül is a genetikus [2] és a hangyakolónia [32] algoritmusok irányába fordult. Jelen tanulmányunk a feladat összetettsége miatt a genetikus algoritmusok módszerét alkalmazta, ahol az ütemtervet egy Nelder–Mead algoritmussal finomítottuk tovább.

3.1. A genetikus algoritmus operátorai és hiperparaméterei

Célfüggvény: A genetikus algoritmus jósági függvénye (33). A GA eredményeként kapott $\mathbf{O}$ mátrixot felhasználva az erőforrás-korlátot nem túllépő, lehető legkorábbi kezdést keressük Nelder–Mead algoritmussal. A megvalósításra a MATLAB programcsomag Global Optimization Toolboxát alkalmaztuk. Itt felhasználtuk, hogy a genetikus algoritmus valamennyi előre megadott hiperparamétere és

operátora beállítható. A részletes leírás azt a célt szolgálja, hogy ezekkel a beállításokkal az algoritmus reprodukálható legyen. A hiperparaméterek beállítását megelőzte egy validációs szakasz, ahol megvizsgáltuk a konvergencia sebességét, valamint az eredményt (szinergiák és rugalmas kapcsolatok nélkül) összehasonlítottuk az egzakt megoldást adó [30] algoritmusának eredményével (lásd: 3. fejezetet).

Egyedek kromoszómáinak felépítése: A kromoszómák három részből állnak. Az első részben, ahol arról döntünk, hogy a rugalmas kapcsolatok és tevékenységelőfordulások közül mely tevékenységeket valósítjuk meg, és mely kapcsolatokat írjuk elő, ott a lehetséges értékek 0 és 1 bináris értékek lehetnek. A következő szakasz az **O** mátrix elemei, amelyek 0 és 1 között bármilyen értéket felvehetnek. Az utolsó szakasz a tevékenységek tényleges kezdési idejét adja meg, amely bármely valós szám lehet. E három szakasz elkülönítése azért fontos, mert más operátorok szükségesek a bináris és a folytonos szakasz kezelésére.

Populáció: A populációban lévő egyedek kezdetben véletlenszerűek. Ugyanakkor eltároljuk egy elit halmazba a megengedett megoldásokat, amelyek célfüggvényértéke a legjobb 5%-ban van. Ezeket abban az esetben használjuk fel, amennyiben a keresés során a populáció többi része nem tartalmaz megengedett megoldást. A populációk 1000-1000 egyedet tartalmaztak.

Kiválasztás: Kiválasztás elsősorban a megengedett megoldásokból körmérkőzés alapján történik. A kiválasztásba a megengedett megoldások 10-szer nagyobb súllyal kerülhetnek be. Összesen egy ilyen körmérkőzésben 10-10 egyed került összehasonlításra. Az előkelőbb helyen végző egyed a célfüggvényértékei alapján számítva, nagyobb valószínűséggel került be a kiválasztott egyedek közé.

Keresztezés: A keresztezés során az egyes szakaszokra más módszert alkalmaztunk. A bináris szakasznál az egyenletes, a folytonos szakasznál az aritmetikai keresztezést alkalmaztuk. A keresztezésnél is a megengedett megoldások 10-szer nagyobb valószínűséggel vettek részt a kereszteződésben.

Mutáció: Itt is szükség volt a bináris és a folytonos szakasz kezelésének szétválasztására. Először annak a szakasznak a kiválasztása történt meg véletlenszerűen, amely mutálódhat. E szakasz hossza 1% volt a hiperparaméterek optimálása után. A kiválasztott bináris szakaszon ez a mutáció $0 \to 1$, $1 \to 0$ váltást jelentett, míg folytonos szakaszon egy véletlenszerű $\pm$ érték hozzáadását.

Mutáció-rekombináció arány: Fontos hiperparaméter a mutáció-rekombináció arány, mely esetünkben 1:5 arányt jelentett, vagyis a futások során 5-ször több rekombináció (kereszteződés) történt, mint mutáció.

Következő generáció, megállási feltétel: A mutáció, keresztezés és a szelekció során kapott új egyedek, valamint a legjobb 100 egyed is bekerült a következő populációba. A leállást két esemény idézhette elő. Vagy elértük a maximális iterációs számot, amely jelen esetben 100 volt, vagy a célfüggvényérték változása az egyes populációkban 10^{-8} alá került.

3.2. Hiperparaméterek beállítása

Hiperparaméterek közül be kellett állítani az elit egyedek arányát, a mutáció-rekombináció arányát, a megengedett megoldások dominanciáját, a populáció nagyságát, valamint az iterációk maximális számát. A paraméterek beállításánál a konvergencia gyorsasága, valamint a célfüggvényérték nagysága játszott szerepet.

3.3. Ütemezés finomhangolása

A genetikus algoritmus (33) célfüggvényre nézve egy jó megengedett (ún. good feasible) megoldást szolgáltat, ugyanakkor a tevékenységek többféle kezdése esetén is teljesülhet a célfüggvényérték maximuma. Így a NM algoritmust arra használtuk, hogy találja meg a tevékenységek lehető legkorábbi kezdését, ahol az erőforráskorlátok nem sérülnek.

3.4. Genetikus algoritmus tesztelése

A feladat komplexitása miatt nagyon nehéz meghatározni az optimális megoldást. Szinergiák figyelembevétele nélkül 5-10 tevékenységre vonatkozóan ugyanakkor léteznek már egzakt megoldók. Ilyen pl. [30] módszere. Ugyanakkor ezek a megoldók nem alkalmazhatók nagyobb számú tevékenység esetén. Az 1. táblázat bemutatja a különböző méretű feladatok esetén a futási időket. Meg kell jegyezni, hogy 10, illetve 30 tevékenység esetén az általunk javasolt módszer is megtalálta az optimális megoldást. Ugyanakkor további tevékenységek esetén csak a konvergenciát tudtuk tesztelni, mely a 3. fejezetben bemutatott hiperparaméterek esetén adta a legjobb konvergenciát.

	Alkalmazott módszer (futásidő, ms)		
Tevékenység	*Egzakt (ns)*	*GA+NN (ns)*	*GA+NN (s)*
10	51 260	1 053	1 077
30	854 992	6 385	6 411
50	–	18 055	18 153
100	–	33 062	33 174

1. táblázat. Futási idők összehasonlítása, (ns): szinergiák nélkül, (s) szinergiákkal való futtatás (5. generációs i5 Intel processzoros számítógépen tesztelve).

4. Tesztelési eredmények

A tesztelés során a vonatkozó szakirodalomban tárgyalt szinergia-struktúrák projektköltségekre gyakorolt hatását vizsgáltuk. Az ehhez szükséges adatokat a [23] iMOPSE, szabadon elérhető projekt generáló szoftverének segítségével állítottuk elő. Ez a szoftver legenerálja a projektstruktúrát, a képességeket, valamint az elvárt munkatartalmat, illetve azt, hogy a munkavállaló maximálisan mekkora mértékben vonható be egy feladat végrehajtásába. Lényegében tehát, bár nem mátrixos formában, de megkapjuk az SMM mátrixból az **A**, **M**, **S** és **W** mátrixokat. Ugyanakkor e generáló nem kezel rugalmas tevékenység-végrehajtásokat, illetve rugalmas kapcsolatokat.

4.1. Vizsgált szimulációs adatbázis bemutatása

A generált adatbázis nem tartalmazta sem a szinergikus hatásokat leíró hálózatokat, sem pedig a rugalmas kapcsolatokat, így azokat nekünk kellett beállítani. A rugalmas kapcsolatok/tevékenység-előfordulások arányát $ff = \{0,1; 0,2; 0,3\}$ mértékben határoztuk meg. A javasolt mátrix-modell két mátrixszal bővíti az eredeti modellt. Az **O** mátrixot az algoritmus keresi, ugyanakkor az **Y** mátrix is hiányzott az eredeti modellből, amely a munkavállalók közötti szinergiákat határozzák meg. Mi a szakirodalomban fellelhető és vizsgált 20 struktúrára vonatkozóan számoltunk mátrixot. A struktúrák szomszédsági mátrixai képezték szinergia mátrixot úgy, hogy ahol nem volt kapcsolat a vizsgált struktúra elemei között, ott a cellák értéke 1 volt, míg a többi helyen, negatív szinergia esetén 1-nél kisebb pozitív szám, míg pozitív szinergia esetén 1-nél nagyobb számot generáltunk. Az 1-től való maximális eltérés mind pozitív, mind negatív irányban maximum 0,3 volt úgy, hogy azok negatív, vagy pozitív hatásai az időigényre vonatkozóan átlagban kiejtsék egymást. Így lehetőség volt a pozitív és a negatív hatások összehasonlítására is.

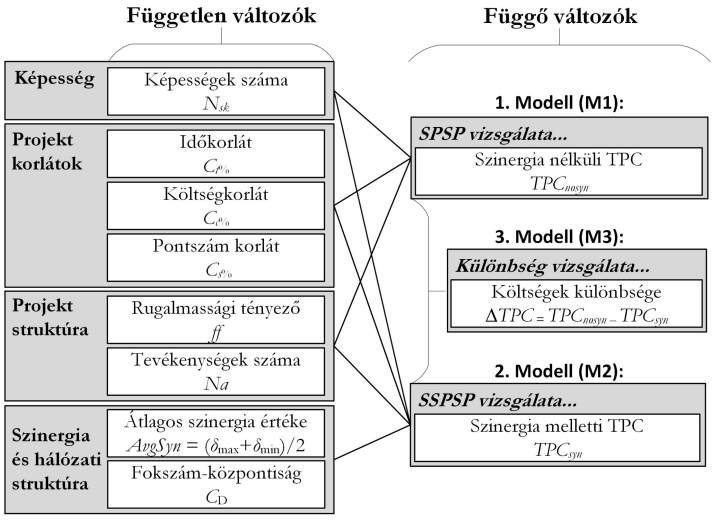
A projektstruktúra meghatározásánál egyrészt [29] kutatási eredményét, másrészt a rugalmas projektek tervezésére vonatkozó módszertanokat [31] vettük alapul. [29] valós projekteket tartalmazó adatbázison kimutatta, hogy a szoftverprojektek esetén tevékenységek végrehajtása inkább párhuzamosan zajlik, ugyanakkor a rugalmas projekttervezési módszerek maximálják az egy időben végrehajtható tevékenységek számát 3-5 tevékenységre. Másrészt a rugalmas szoftverek projektszakaszai (ún. sprintek) általában lezajlanak 2-5 hét alatt [12], ez pedig megköti a tevékenységek számát, amely jelen esetben 30-50 tevékenység volt a szimuláció során. A munkavállalók számát a szinergia-hálózatok csúcspontjai határozták meg. A korlátokat úgy határoztuk meg, hogy a szinergikus hatást, valamint a rugalmas kapcsolatokat nem figyelembe véve mekkorák lennének a projekt (idő-, költség-, erőforrás-igény) paraméterei [23] által számított módszerrel, majd e minimális értékek 1, 1,25 és 1,5-szeresét felhasználva határoztuk meg az aktuális korlátokat, így biztosítva, hogy a feladatnak legyen megoldása. Tesztelés céljából az időre, illetve

költségre optimálva, a rugalmas kapcsolatokat nem figyelembe véve összehasonlítottuk [23] eredményeit az általunk javasolt algoritmussal, melyek célfüggvényértékbeli eltérése 1% alatt volt. Ugyanakkor a későbbi vizsgálatokban mi az általunk javasolt (33) összetett célfüggvényértéket javasoltuk, így az összehasonlításban a szinergikus hatásokra, koncentráltunk.

A beállításokkal összesen 69 984 problémát kaptunk. Valamennyi problémát a genetikus algoritmussal 10-szer megoldottuk.

4.2. Eredmények ismertetése

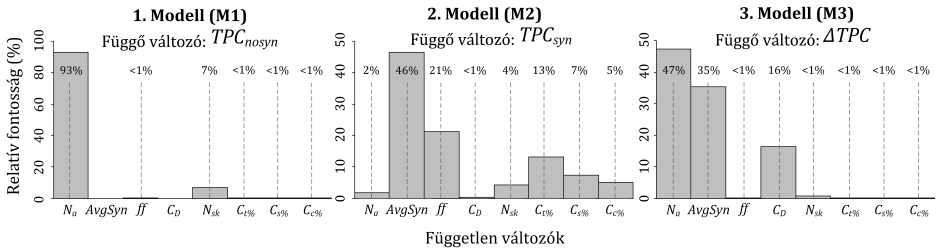
Az optimális projekttervek meghatározását követően megvizsgáltuk, hogy a projektek mely jellemzői és milyen mértékben hatottak a projektek teljes költségére (TPC). Három esetet különböztettünk meg. Míg az első kettő (M1 és M2) abban tért el egymástól, hogy számítások során figyelembe vettük-e a munkavállalók szinergiahálózatát, a harmadik esetben (M3) az első két eset különbségét, vagyis a szinergiahálózat hatását vizsgáltuk (lásd 2. ábra).



2. ábra. Eredmények vizsgálatának modelljei

A függő változók relatív fontosságának meghatározására a Matlab Regression Learner App [20] regression tree ensemble algoritmusát használtuk.[12] Az egyes modellek esetén meghatározott relatív fontossági értékeket a 3. ábra szemlélteti.

[12] A számítás során 10-szeres keresztvalidációt használtunk, a hiperparaméterek meghatározása pedig bayesi optimalizálás segítségével történt. A változók relatív fontosságát a Matlab *predictorImportance* függvényével jellemeztük [21].

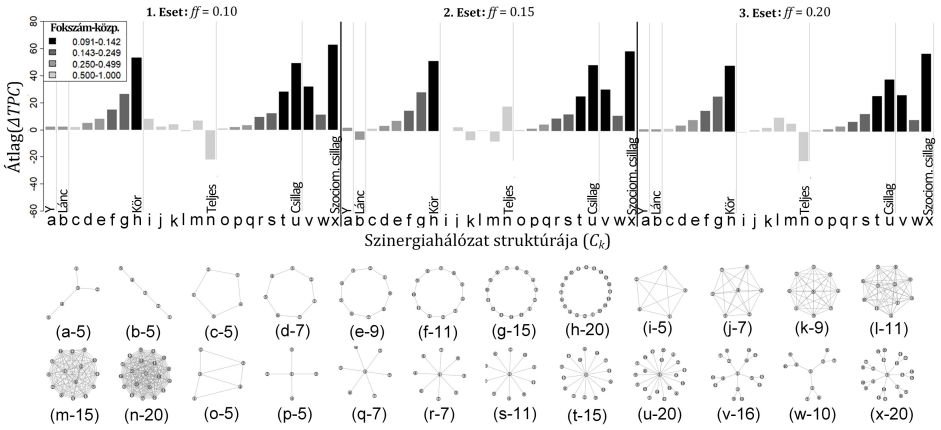


3. ábra. Függő változók relatív fontossága

Míg az eredeti SPSP esetében (lásd 3. ábra – M1) a projekt mérete (N_a) és a munkavállalók képességeinek száma (N_{sk}) gyakorolta a legnagyobb hatást a projektek költségeire, a munkavállalók közti szinergiák figyelembevétele mellett (lásd 3. ábra – M2) a legfontosabb változónk maga az átlagos szinergiaérték ($AvgSyn$) lett. Az utóbbi esetben továbbá mind a rugalmassági tényező (ff), mind pedig az egyes korlátok ($C_{t\%}, C_{s\%}, C_{c\%}$) fontosabbnak bizonyultak, mint a projektméret (N_a) és a képességek száma (N_{sk}). E két modell költségeinek különbségét (lásd 3. ábra – M3) legerősebben a projekt mérete (N_a – 47%), az átlagos szinergia értéke ($AvgSyn$ – 35%), valamint a szinergiahálózat fokszám-központisága (C_D – 16%) befolyásolta.

A fenti eredmények a szinergia-alapú megközelítés fontosságát hangsúlyozzák, hiszen amellett, hogy a szinergiával kapcsolatos paraméterek jelentős hatást gyakorolnak a projektek költségeire, a strukturális paraméter (C_D) esetén mért fontosság összhangban áll a vonatkozó szakirodalommal (lásd pl. [1]). Annak érdekében, hogy mélyebb ismereteket szerezzünk a szinergiahálózat költségekre gyakorolt hatásáról, megvizsgáltuk továbbá azok struktúrája és a projektköltség között fennálló kapcsolatot. A 4. ábra a szinergiahálózat struktúrájának (C_k) és központisági értékének (C_D, lásd színezés), valamint a projekt rugalmasságának (ff) ΔTPC-re (lásd 3. ábra – M3) gyakorolt hatását szemlélteti.

A 4. Ábra alapján az alacsony fokszám-központiság általában a költségek nagyobb csökkenéséhez vezet, azonban ez jelentősen függ a szinergiahálózat topológiájától (C_k) is. Bár a relatív fontosságok vizsgálata során (lásd 3. ábra) megállapítottuk, hogy a projekt rugalmassága (ff) elhanyagolható mértékben befolyásolja a ΔTPC-t (lásd 3. ábra), a 4. ábra alapján a lánc- és a teljes gráf hálózatok még e jelentéktelen változásokra is érzékenyen reagálnak. Néhány topológia esetében megfigyelhetjük, hogy TPC_{syn} nagyobb, mint TPC_{nosyn}, ami negatív ΔTPC-t eredményez. Ez ellentétes Sparrowe és mtsai. [28] megállapításával, hiszen a modellben a decentralizált hálózatok (pl. kör és teljes) nem képesek olyan mértékben csökkenteni a költségeket, mint a centralizált hálózatok (pl. a csillag és a szociometrikus csillag). Továbbá a véletlenszerűen kedvező és kedvezőtlen szinergiákat tartalmazó hálózatok esetében a leginkább decentralizált topológia (teljes hálózat)



4. ábra. Szinergiahálózat struktúrájának hatása a projektköltségre

vezet a legrosszabb eredményre, hiszen ez a struktúra különösen érzékeny a negatív szinergiákra.

5. Összefoglalás

A javasolt algoritmus két ponton egészíti ki a meglévő módszertant. Lehetőséget teremt arra, hogy figyelembe vegyük egyrészt a munkavállalók közötti pozitív, vagy negatív szinergiát, másrészt, a szoftverprojektekben egyre inkább elterjedt rugalmas tervezési megközelítéseket. Az eredmények nem csak a kölcsönös szinergia projektköltségekre gyakorolt jelentős – kedvező és kedvezőtlen – hatására mutatnak rá, de a szinergia-hálózat struktúrájának szerepét is hangsúlyozzák.

Az eltérő hálózatok különböző módon erősíthetik, vagy gyengíthetik a szinergikus hatásokat, amely a projektcsapatok összeállítása során fontos információt szolgáltathat a projektmenedzserek számára.

A különböző szoftverfejlesztési módszertanok (pl. spirál model, vízesés modell, V modell) esetén eltérőek lehetnek a tevékenység számok, valamint a projektben résztvevő személyek száma is. Ilyen vizsgálatokat jelen tanulmányunk nem tartalmaz. Jelen tanulmányunk a rugalmas, agilis környezetet vizsgálja, ahol a tevékenységkapcsolatok is lehetnek rugalmasak. Ugyanakkor láthattuk, hogy a flexibilitás szerepe a többi tényezőhöz képest mérsékelt, így felvethető, érdemes-e ezt a megközelítést más szoftverfejlesztési környezetre is kiterjeszteni. Ezt a felvetést egy következő tanulmányban vizsgáljuk meg.

Köszönetnyílvánítás

A közlemény megjelenését a TKP2020-NKA-10 sz. projekt keretében a Nemzeti Kutatási, Fejlesztési és Innovációs Alap 2020-4.1.1-TKP2020 sz. Tématerületi Kiválóság Programja finanszírozta, valamint az Innovációs és Technológiai Minisztérium ÚNKP-20-3 kódszámú Új Nemzeti Kiválóság Programjának a Nemzeti Kutatási, Fejlesztési és Innovációs Alapból finanszírozott szakmai támogatásával készült. Külön köszönjük Dr. Dósa György Professzor hasznos szóbeli megjegyzéseit.

Hivatkozások

[1] M.K. Ahuja, D.F. Galletta and K.M. Carley: *Individual centrality and performance in virtual R&D groups: An empirical study*, Management Science, Vol. **49**, No. **1**, pp. 21–38 (2003). DOI: 0.1287/mnsc.49.1.21.12756

[2] E. Alba and J.F. Chicano: *Software project management with GAs*, Information Sciences, Vol. **177**, No. **11**, pp. 2380–2401 (2007). DOI: 10.1016/j.ins.2006.12.020

[3] S. Bouajaja and N. Dridi: *A survey on human resource allocation problem and its applications*, Operational Research, Vol. **17**, No. **2**, pp. 339–369 (2017). DOI: 10.1007/s12351-016-0247-8

[4] T.R. Browning: *Managing complex project process models with a process architecture framework*, International Journal of Project Management, Vol. **32**, No. **2**, pp. 229–241 (2017). DOI: 10.1016/j.ijproman.2013.05.008

[5] C.K. Chang, H. Jiang, Y. Di, D. Zhu and Y. Ge: *Time-line based model for software project scheduling with genetic algorithms*, Information and Software Technology, Vol. **50**, No. **11**, pp. 1142–1154 (2008). DOI: 10.1016/j.infsof.2008.03.002

[6] F. Chicano, F. Luna, A.J. Nebro and E. Alba: *Using multi-objective metaheuristics to solve the software project scheduling problem*, Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, Vol. **50**, pp. 1142–1154 (2008). DOI: 10.1145/2001576.2001833

[7] C.A. Coello, G.B. Lamont and D.A. Veldhuizen: *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*, Springer-Verlag New York (2006).

[8] B. Crawford, R. Soto, F. Johnson, E. Monfroy and F. Paredes: *A max-min ant system algorithm to solve the software project scheduling problem*, Expert Systems with Applications, Vol. **41**, No. **15**, pp. 6634–6645 (2014). DOI: 10.1016/j.eswa.2014.05.003

[9] M. Danilovic and T.R. Browning: *Managing complex product development projects with design structure matrices and domain mapping matrices*, International Journal of Project Management, Vol. **25**, No. **3**, pp. 300-314 (2007). DOI: 10.1016/j.ijproman.2006.11.003

[10] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan: *A fast and elitist multiobjective genetic algorithm: NSGA-II*, IEEE Transactions on Evolutionary Computation, Vol. **6**, pp. 182–197 (2002). DOI: 10.1109/4235.996017

[11] M. Di Penta, M. Harman and G. Antoniol: *The use of search-based optimization techniques to schedule and staff software projects: an approach and an empirical study*, Software: Practice and Experience, Vol. **41**, No. **5**, pp. 495–519 (2011). DOI: 10.1002/spe.1001

[12] T. Dingsøyr, S. Nerur, V. Balijepally and N.B. Moe: *A decade of agile methodologies: Towards explaining agile software development*, Journal of Systems and Software, Vol. **85**, No. **6**, pp. 495–519 (2012). DOI: 10.1016/j.jss.2012.02.033

[13] M. Hapke, A. Jaszkiewicz and R. Slowinski: *Fuzzy project scheduling system for software development*, Fuzzy Sets and Systems, Vol. **67**, No. **1**, pp. 101–117 (1994). DOI: 10.1016/0165-0114(94)90211-9

[14] S. Hartmann and D. Briskorn: *A survey of variants and extensions of the resource-constrained project scheduling problem*, European Journal of Operational Research, Vol. **207**, No. **1**, pp. 1–14 (2010). DOI: 10.1016/j.ejor.2009.11.005

[15] J.D. Knowles and D.W. Corne: *Approximating the nondominated front using the pareto archived evolution strategy*, Evolutionary Computation, Vol. **8**, No. **2**, pp. 149–172 (2000). DOI: 10.1162/106365600568167

[16] Kosztyán, Zs.T.: *Exact algorithm for matrix-based project planning problem*, Expert Systems with Applications, Vol. **42**, No. **9**, pp. 4460–4473 (2015). DOI: 10.1016/j.eswa.2015.01.066

[17] Kosztyán, Zs.T., Pribojszki-Németh, A. and Szalkai, I.: *Hybrid multimode resource-constrained maintenance project scheduling problem*, Operations Research Perspectives, Vol. **6**, pp. 100–129 (2019). DOI: 10.1016/j.orp.2019.100129

[18] Kosztyán, Zs.T. and Szalkai, I.: *Multimode resource-constrained project scheduling in flexible projects*, Journal of Global Optimization, Vol. **76**, No. **1**, pp. 211–241 (2020). DOI: 10.1007/s10898-019-00832-8

[19] F. Luna, D.L. González-Álvarez, F. Chicano and M.A. Vega-Rodríguez: *The software project scheduling problem: A scalability analysis of multi-objective metaheuristics*, Applied Soft Computing, Vol. **15**, pp. 136–148 (2014). DOI: 10.1016/j.asoc.2013.10.015

[20] MathWorks: *Regression Learner App* (2019).
URL: https://www.mathworks.com/help/stats/regress ion-learner-app.html [Letöltve: 2020. augusztus 25.]

[21] MathWorks: *Predictor importance* (2019).
URL: https://www.mathworks.com/help/stats/compact regressionensemble.predictorimportance.html
[Letöltve: 2020. augusztus 25.]

[22] J.L. Moreno: *The Sciometry Reader*, EGlencoe, Illinois: The Free Press (1960).

[23] P.B. Myszkowski, M. Laszczyk, I. Nikulin and M. Skowroński: *IMOPSE: a library for bicriteria optimization in multi-skill resource-constrained project scheduling problem*, Soft Computing, Vol. **23**, No. **10**, pp. 3397–3410 (2019). DOI: 10.1007/s00500-017-2997-5

[24] A.J. Nebro, J.J. Durillo, F. Luna, B. Dorronsoro and E. Alba: *Design issues in a multiobjective cellular genetic algorithm*, International Conference on Evolutionary Multi-Criterion Optimization, Springer, pp. 126–140 (2007).

[25] L.D. Otero, G. Centeno, A.J. Ruiz-Torres and C.E. Otero: *A systematic approach for resource allocation in software projects*, Computers & Industrial Engineering, Vol. **56**, No. **4**, pp. 1333–1339 (2009). DOI: 10.1016/j.cie.2008.08.002

[26] X. Shen, L.L. Minku, N. Marturi, Y. Guo and Y. Han: *A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling*, Information Sciences, Vol. **428**, pp. 1–29 (2018). DOI: 10.1016/j.ins.2017.10.041

[27] R.S. James: *Task demands, group interaction and group performance*, Sociometry, Vol. **34**, No. **4**, pp. 483–495 (2018). DOI: 10.2307/2786194

[28] R.T. Sparrowe, R.C. Liden, S.J. Wayne and M.L. Kraimer: *Social networks and the performance of individuals and groups*, Academy of Management Journal, Vol. **44**, No. **2**, pp. 316–325 (2001). DOI: a 10.2307/3069458

[29] M. Vanhoucke: *Measuring the efficiency of project control using fictitious and empirical project data*, International Journal of Project Management, Vol. **30**, No. **2**, pp. 252–263 (2012). DOI: 10.1016/j.ijproman.2011.05.006

[30] M.Á. Vega-Velázquez, A. García-Nájera and H. Cervantes: *Measuring the efficiency of project control using fictitious and empirical project data*, International Journal of Production Economics, Vol. **202**, pp. 145–161 (2018). DOI: 10.1016/j.ijpe.2018.04.020

[31] R.K. Wysocki: *Effective Project Management: Traditional, Agile, Extreme*, John Wiley & Sons (2011).

[32] J. Xiao, X. Ao and Y. Tang: *Solving software project scheduling problems with ant colony optimization*, Computers & Operations Research, Vol. **40**, No. **1**, pp. 33–46 (2013). DOI: 10.1016/j.cor.2012.05.007

[33] E. Zitzler, M. Laumanns and L. Thiele: *SPEA2: Improving the strength Pareto evolutionary algorithm*, TIK-report, Vol. **103** (2001). DOI: 10.3929/ethz-a-004284029

Prof. Kosztyán Zsolt Tibor az MTA-PE Budapest Rangsor Kutatócsoport tudományos főmunkatársa, valamint a Pannon Egyetem Kvantitatív Módszerek Intézeti Tanszék tanszékvezető egyetemi tanára. Kutatási területe: komplex rendszerek menedzsmentje, hálózatos modellezése, különös tekintettel a projekt-, termelés- és karbantartásmenedzsment problémák hálózatalapú modellezésére. Kutatási területe ötvözi az alkalmazott informatika, a hálózattudományok és a menedzsment tudományok területeit. MTA-VEAB év kutatója 2013. Magyary Zoltán, Bolyai János és Új Nemzeti Kiválóság Programok posztdoktori ösztöndíjasa. A Neumann János Számítógéptudományi Társaság, Gazdaságinformatikai Oktatási és Kutatási Fórum (http://gikof.njszt.hu), a Gazdaságmodellezési Társaság, valamint a PMUni International Network for Professional Education and Research in Process and Project Management szervezetek elnökségi tagja.

PROF. KOSZTYÁN ZSOLT TIBOR

Pannon Egyetem
8200 Veszprém, Egyetem utca 10.
kosztyan.zsolt@gtk.uni-pannon.hu

MTA-PE Budapest Rangsor Kutatócsoport
8200 Veszprém, Egyetem utca 10.
kosztyan.zsolt@gtk.uni-pannon.hu

Dr. Szalkai István 1984-ben végzett az ELTE TTK matematikus szakán, 1993-ban egyetemi doktori (ELTE, halmazalmélet), 2013-ban PhD (Reakciómechanizmusok, PE) fokozatot szerzett. A Pannon Egyetem (Veszprém) Matematika Tanszékén oktat 1987 óta, egyetemi docens 2014 óta. Fő kutatási területei absztrakt algebra, diszkrét matematika és a matematika népszerűsítése, oktatási kérdések. A Bolyai Társulat Veszprém Megyei Tagozatának elnöke, a Mathematical Association of America "Lester R. Ford Award" díjának tulajdonosa.

DR. SZALKAI ISTVÁN

Pannon Egyetem
8200 Veszprém, Egyetem utca 10.
szalkai@almos.uni-pannon.hu

Kurbucz Marcell Tamás jelenleg a PE GSDI doktorjelöltje, valamint a Wigner FK tudományos segédmunkatársa. Doktori tanulmányait megelőzően elvégezte az ELTE TáTK közgazdasági elemző mesterképzését, valamint a PE GTK műszaki menedzser és emberi erőforrások alapképzéseit. Fő kutatási területe a komplex hálózatok elemzése. Tudományos munkájáért 2015-ben Pro Scientia Aranyéremmel tüntették ki.

KURBUCZ MARCELL TAMÁS
Pannon Egyetem
8200 Veszprém, Egyetem utca 10.
kurbucz.marcell@gtk.uni-pannon.hu
Wigner Fizikai Kutatóközpont
1121 Budapest, Konkoly-Thege Miklós út 29-33.
kurbucz.marcell@wigner.hu

SYNERGIES IN SOFTWARE PROJECT SCHEDULING

Zsolt Tibor Kosztyán, István Szalkai, Marcell Tamás Kurbucz

In the literature on software development, the common issue of resource allocation and task scheduling is referred to as the software project scheduling problem (SPSP). While most of the software projects are managed in an agile framework, the SPSP ignores the two main features of this approach: the flexibility of planning and the complexity of teamwork. This paper focuses on these two possible approaches to extend the traditional SPSP. First, a general form of the SPSP assumes fixed logic plans; however, applying flexible dependencies and using task priorities instead of fixed occurrences will result in more flexible project plans consistent with the agile approach. Second, while software development projects and particularly software development projects using the agile approach place a greater emphasis on teamwork than the traditional methods, in the SPSP, employees are regarded as independent resources. To understand the impact of synergies on project scheduling, the solutions of the traditional and extended SPSPs are analyzed and compared on a simulated project database.

Keywords: Software Project Scheduling; Synergy Network; Genetic Algorithm