Válasz Dr. Kovács László Professzor Úr

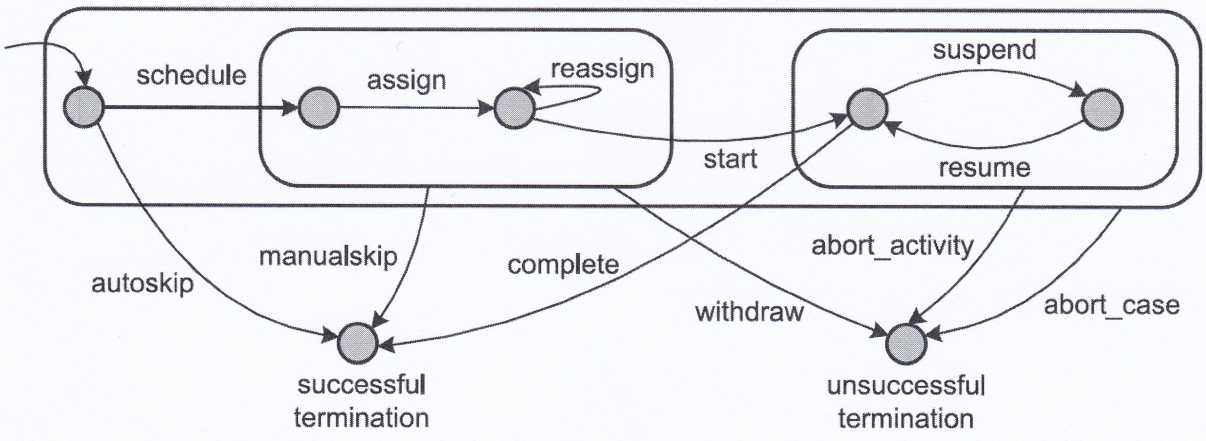
„Üzleti folyamatok online nyomon követése és javítása folyamatbányászati és mesterséges intelligencia algoritmusok alkalmazásával”

című doktori (PhD) disszertációmhoz megküldött bírálatára

1. kérdés/megjegyzés: Az alapfogalmaknál (16. oldal) az olvasható: „az esemény tevékenységeket foglal magába. Egy tevékenység végrehajtása több eseményből is állhat”. Kérdésem, hogyan biztosítható a modellben a felvázolt rekurzív szerkezet hatékony kezelése. Létezik-e egy maximális beágyazási mélység? Hogyan illeszkedik ez a rekurzív szerkezet az 1. Definícióhoz?

1. kérdésre/megjegyzésre adott válasz: Egy esemény csak akkor foglalhat magába több tevékenységet, ha összetett tevékenységről van szó. Ebben az esetben az eseményfolyamban csak az összetett tevékenység figyelhető meg, a résztevékenységek nem, ezért a folyamatmodellben is csak az összetett tevékenység szerepel. A modellben a rekurzív szerkezet kezelése az eseményfolyam megfigyelhetőségén alapul: ha a résztevékenységek nem jelennek meg külön eseményként, akkor a modell nem követeli meg ezek beágyazott feldolgozását. Ennek megfelelően a maximális beágyazási mélység az eseményszintű reprezentációban gyakorlatilag 1, mivel a mélyebb rétegek nem válnak láthatóvá az eseményfolyamban.

Egyszerű eseményfolyam esetén azonban egy tevékenység végrehajtása több eseményből is állhat. A standard tranzakciós életciklus-modell (lásd az 1. ábrán) alapján egy tevékenységhez tetszőleges számú esemény tartozhat. Például a „reassign” esemény bármennyiszer végrehajtódhat, és a „suspend” és „resume” események is többször ismétlődhetnek egymás után. Az elfogadott tranzakciós életciklus állapotok ugyanakkor folyamat- és tevékenységfüggők lehetnek.



1. ábra: Standard tranzakciós életciklus-modell [1]

A folyamatmodellben általában minden tevékenységhez csak egy átmenet tartozik, jellemzően a „complete" eseményhez. Ebben az esetben csak az ilyen tranzakciótípust tartalmazó eseményeket vesszük figyelembe a megfelelőség-ellenőrzés során. Amennyiben más tranzakciótípusokat is be kívánunk vonni, a folyamatmodell átmeneteinek elnevezésében szerepeltetni kell a tevékenység nevét és a tranzakciótípust is (pl. „review-start" és „review-complete").

Az 1. Definíció alapján egyszerű eseményfolyam esetén a tevékenységhez tartozó események külön-külön megfigyelési egységként, azaz eseményként jelennek meg az eseményfolyamban. Ezekben az eseményekben az eset- és tevékenységazonosító megegyezik, míg a tranzakciótípus eltér, és a további attribútumok között található. Így az 1. Definíció szerinti eseményreprezentáció természetes módon kizárja az összetett tevékenységek részletes bontását, és a rekurzív szerkezet az eseményfolyamban nem jelenik meg explicit módon.

2. kérdés/megjegyzés: A 11. Definíció alapján az igazítási mozgás megengedi a (>>, t) lépéseket. Kérdésem, hogy egy ciklust tartalmazó folyamatmodellben ez eredményezhet-e egy végtelen (vagy nagyon hosszú véges) ciklust. Mivel lehet ezt az esetet elkerülni?

2. kérdésre/megjegyzésre adott válasz: Igen, egy ciklust tartalmazó folyamatmodell eredményezhet végtelen (vagy nagyon hosszú véges) ciklust az igazítás előállítása során, de csak abban az esetben, ha az alkalmazott tevékenységköltség-függvény a ciklust okozó átmenetet tartalmazó modell mozgáshoz 0 költséget rendel, míg a többi lehetséges igazítási mozgásokhoz pozitív költséget. Emellett, ha az adott átmenethez változóírási műveletek is tartoznak, akkor az alkalmazott változóköltség-függvény minden eltéréshez 0 költséget rendel. Ebben az esetben a módosított inkrementális A* algoritmus a végtelen ciklust részesítheti előnyben, mivel az olcsóbbnak tűnik, mint a többi igazítási mozgás.

Az ilyen esetek elkerülhetők, ha megfelelő tevékenységköltség-függvényt használunk. Például az alapértelmezett költségfüggvény, amely minden modell és eseménynapló mozgáshoz 1-es költséget rendel, megelőzi a végtelen ciklusok kialakulását, mivel a kereső algoritmus a legkisebb költségű (és így legrövidebb) igazítást preferálja. Így a ciklusos viselkedés csak akkor jelenik meg az igazításban, ha azt a megfigyelt események indokolják.

3. kérdés/megjegyzés: Az Algoritmus 1.-ben egy végtelen ciklus szerepel: *while true: ...* nincs kilépési utasítás. Minek hatására fog itt a ciklus leállni?

3. kérdésre/megjegyzésre adott válasz: Matematikailag az eseményfolyamot események végtelen sorozataként definiáljuk, ezért az algoritmusban végtelen ciklus szerepel. A valóságban azonban az eseményfolyam rendszerint véges ideig tart, viszont előre nem tudható, hogy mikor ér véget, azaz mikor kell leállítani a megfigyelést. A leállási feltétel megoldásfüggő lehet. Ideális esetben az eseményfolyam leállásáról egyértelmű jelzést kapunk, és a ciklusból való kilépés ehhez a jelzéshez köthető. A leállási feltétel lehet időalapú is: ha egy adott időintervallumon belül nem figyelhető meg új esemény, akkor a megfigyelés

megszakítható. Ez utóbbi esetben azonban fennáll annak a veszélye, hogy a megfigyelés idő előtt leáll, vagy épp ellenkezőleg, a folyamat feleslegesen fut tovább, miközben az eseményfolyam már régen leállt.

4. kérdés/megjegyzés: A (3.9) képletnél szerepel, hogy az egyes útvonaltervek által kielégítendő teljes igény nem haladja meg a hozzá rendelt jármű teljes kapacitását. A képletben a szumma aggregátor szerepel. Ha jól értelmezem, a feladatok szállítások a fejcsúcs és a végcsúcs között, azaz a jármű leadja az árut a végcsúcsnál. Mivel minden feladatot egyetlen műveletben végeznek el, a jármű üres lesz a következő feladat megkezdése előtt. Ekkor viszont nem tűnik logikusnak a feladatokra vonatkozó szummázás a képletben. Mi miatt érvényes mégis a szumma aggregálás?

4. kérdésre/megjegyzésre adott válasz: Egy CARP példa begyűjtési szolgáltatásra (pl. városi hulladékgyűjtés) vagy szétosztási szolgáltatásra (pl. az utcák sózása) vonatkozhat. A 3.9 képletben szereplő szumma aggregátor azt fejezi ki, hogy az egy adott útvonaltervhez rendelt összes feladat végrehajtása során érintett teljes mennyiség (pl. begyűjtendő vagy szétosztandó anyag) nem haladhatja meg a jármű kapacitását. Ez a megkötés alapvető része a CARP modellnek. A kérdésben felvetett értelmezés, miszerint a jármű minden feladat elvégzése után kiürül, és így minden feladat önállóan értelmezendő, nem illeszkedik a CARP szemléletéhez. A CARP-ban ugyanis a jármű egyetlen útvonal során több feladatot is elvégezhet, miközben a kapacitása fokozatosan csökken (szétosztás esetén) vagy nő (begyűjtés esetén). A kapacitás tehát kumuláltan értendő az útvonal mentén, nem pedig minden feladat után lenullázva. A 3.9 képlet tehát a teljes útvonaltervhez rendelt feladatok összesített begyűjtendő/szétosztandó anyagmennyiségére vonatkozik.

Begyűjtési szolgáltatás esetén a jármű üresen (0 mennyiséggel) indul a telephelyről, és minden t feladat elvégzése során $dem(t)$ mennyiséget gyűjt be. Az útvonalterv végrehajtása során begyűjtött teljes mennyiség (vagyis a feladatokhoz tartozó igények összege) nem haladhatja meg a jármű q kapacitását. Ha a jármű eléri a kapacitáshatárt, akkor a jármű több feladatot már nem tud elvégezni, ezért visszatér a telephelyre.
Szétosztási szolgáltatás esetén a jármű feltöltve (q mennyiséggel) indul, és minden t feladat elvégzése során $dem(t)$ mennyiséget oszt szét. Ebben az esetben is a jármű kapacitását nem haladhatja meg a teljes szétosztott mennyiség az útvonalterv végrehajtása során, azaz az elvégzett feladatok igényeinek összege nem lehet nagyobb, mint q. Ha a szétosztott összmennyiség eléri q-t, a jármű újratöltés céljából visszatér a telephelyre.

5. kérdés/megjegyzés: A CARP-ABC algoritmusnál szerepel a felfedezés-feltárás alapú módszerek megkülönböztetése. Itt az elnevezéseknél érzek igényt a pontosításra, mivel az irodalomban eltérő értelmezések is megjelennek. A feltárást szokás a dolgozó méh munkamódszeréhez kapcsolni. A dolgozatban a leírtak alapján a feltárás az irányított, heurisztika vezérelt felfedezéshez áll inkább közel. Látható, hogy nem könnyű röviden, egy szóval jellemezni a javasolt módszert.

5. kérdésre/megjegyzésre adott válasz: A dolgozatban az „exploration" és a „discovery" elnevezések magyar megfelelőiként használtam a „felfedezés" és a „feltárás" kifejezéseket. Elismerem, hogy a két kifejezés magyarra fordítása nem egyértelmű, mivel mindkét szó jelentése részben átfed, és az irodalomban sem mindig következetes az elkülönítésük. A dolgozatban a „felfedezés" a keresési tér minél szélesebb körű bejárását jelöli, amelyet elsősorban véletlenszerű döntések irányítanak, míg a „feltárás" az ígéretesnek tűnő megoldások környezetében végzett, céltudatosabb keresést jelenti.

6. kérdés/megjegyzés: Milyen megfontolásokkal indokolható az $sqrt(l)$ méretkorlát a részútvonaltervekre (Algoritmus 6)?

6. kérdésre/megjegyzésre adott válasz: A részútvonalterv művelet az utazó ügynök problémához kifejlesztett mohó résztúra mutációs (*Greedy Sub Tour Mutation*, röviden *GSTM*) műveleten alapul [2], ezért a paraméter-beállításokat annak optimalizált változatából vettem át. A GSTM esetén a $\sqrt{l_k}$ maximális részútvonal méretkorlát bizonyult a legjobbnak a paraméteroptimalizáció során. Ez a hossz kellően hosszú ahhoz, hogy érdemi módosításokat hajtson végre az l_k hosszúságú útvonalterven, de kellően rövid ahhoz, hogy finom, fokozatos változásokat tegyen lehetővé. Ez elősegíti a keresési tér hatékony, lokálisan érzékeny bejárását, elkerülve a túl nagy ugrásokat, amelyek ronthatják a megoldás minőségét.

7. kérdés/megjegyzés: A 3.3 táblázatnál a bevezetett mérőszámok jól mutatják az egyes módszerek relatív hatékonyságát. Emellett viszont még helyet kaphatott volna egy oszlop, amely azt mutatja, hogy az esetek hány százalékában volt egyáltalán javulás?

7. kérdésre/megjegyzésre adott válasz: A 3.3 táblázat valóban csak az egyes módszerek relatív hatékonyságát mutatja. A kapcsolódó információkat az alábbi táblázatokban (1. és 2. táblázat) tűntettem fel. Az 1. táblázat példánként mutatja az iterációk eredményeit futásokra átlagolva. A második oszlop az iterációk teljes számát mutatja. A harmadik oszlop (Nincs javulás) azoknak az iterációknak a számát és százalékos eloszlását tartalmazza, ahol nem sikerült jobb megoldást találni. A negyedik oszlop (Van javulás) pedig azoknak az iterációknak a számát és százalékos arányát mutatja, ahol történt javulás. A 2. táblázat keresésenként, azaz a végrehajtott dolgozó méh fázisonként átlagolva mutatja az iterációk eredményeit.

1. táblázat: Futásonkénti iterációk száma átlagolva

Példa	Iterációk száma	Nincs javulás	Van javulás
egl-e1-A	43 673	40 554 (92,86%)	3 119 (7,14%)
egl-s1-A	34 883	31 641 (90,71%)	3 242 (9,29%)
egl-g1-A	16 312	11 165 (68,44%)	5 147 (31,56%)

2. táblázat: Keresésenkénti iterációk száma átlagolva

Példa	Iterációk száma	Nincs javulás	Van javulás
egl-e1-A	303,62	281,95 (92,86%)	21,67 (7,14%)
egl-s1-A	344,13	312,12 (90,70%)	32,01 (9,30%)
egl-g1-A	4 189,78	2 867,77 (68,45%)	1 322,00 (31,55%)

8. kérdés/megjegyzés: A 2.1.2-ben pontosítást javasolnék az „adat-orientált" elnevezésben, mivel az „adat" szokásos értelmezése, hogy minden ami mérhető az adat, így az erőforrás leírások, kapcsolatok adatai is. Például az ide tartozó „Erőforrás-tudatos igazítások" fejezetben explicite szerepel az az alkalmazott CRUD mátrixnál az „adatobjektumok" elnevezés.

8. kérdésre/megjegyzésre adott válasz: A szakirodalomban az „adattudatos igazítások" (data-aware alignments, pl. [3]) kifejezés terjedt el, ezért a dolgozat első változatában ezt a megnevezést használtam. Egy korábbi bírálat alapján azonban az „adat-orientált" kifejezés használatát javasolta, így a dolgozatban ennek megfelelően módosítottam a terminológiát.

Fontos megjegyezni, hogy az adattudatos (vagy adat-orientált) igazítások célja annak ellenőrzése, hogy az eseménynaplóban rögzített adatértékek helyesek-e és összhangban vannak-e az üzleti szabályokkal. Ide elvileg az erőforrásokra vonatkozó adatok is beletartozhatnak, ha azok az eseményekben szerepelnek mint strukturált adatok (például szerepkör vagy név attribútumként). Azonban az erőforrás-tudatos igazítások külön perspektívaként jelennek meg a szakirodalomban, mivel ezek nem pusztán az értékek helyességét vizsgálják, hanem a tevékenységek és a végrehajtók közötti jogosultsági és szervezeti megfelelést is ellenőrzik (például a „négy szem elve" betartását).

Az adatobjektum (data object) kifejezés sem mindig egyértelmű. Használata utalhat egyszerű név-érték párokra, de összetettebb adatstruktúrákra is. Az adattudatos igazítások technikái jellemzően olyan adatokat tudnak kezelni, amelyek eseményszinten, strukturált név-érték párokként jelennek meg, és nem igénylik a kapcsolatok vagy komplex viszonyok elemzését, mint az erőforrás-tudatos megközelítések.

9. kérdés/megjegyzés: A matematika oldaláról nézve, túlzónak tűnt a „képes kezelni az összes lehetséges eseményt" kifejezés. Természetesen a gyakorlatban oda képzeljük a megfelelő kontextust, de talán szebb, ha explicite mi adunk egy keretet. Például leszűkítve a statisztikailag elképzelhető esetekre.

9. kérdésre/megjegyzésre adott válasz: Valóban, a „képes kezelni az összes lehetséges eseményt" kifejezés pontosítása indokolt lehet. A megfogalmazás alatt azokat az eseményeket értem, amelyek az adott probléma kontextusán belül statisztikailag elképzelhetőek és a feladat szempontjából relevánsak.

10. kérdés/megjegyzés: Egy kicsit szokatlan volt olvasni az „éretlen, még nem eléggé kiforrott folyamatok" jelölést. Esetleg tudna javasolni szinonimát az „éretlen" szó helyett?

10. kérdésre/megjegyzésre adott válasz: Az „éretlen" kifejezés helyett célszerűbb lehet például a „kezdetleges", „formálódó", „alacsony érettségi szintű" vagy „korai szakaszban lévő" kifejezések, amelyek jobban tükrözik a folyamatok fejlesztés alatt álló állapotát.

11. kérdés/megjegyzés: A dolgozatban csak elvétve található elírás, ezek valahogy átmentek az ellenőrzésen. Ilyen például a MOCC definícióban szereplő „Confromance" vagy „BPMN folyamatmodel".

11. kérdésre/megjegyzésre adott válasz: Valóban néhány elírás (pl. „Confromance" a MOCC definícióban, illetve „BPMN folyamatmodel") a többszöri átolvasás ellenére is benne maradt a szövegben.

12. kérdés/megjegyzés: Érdemes-e a rövid rész-szakaszokat preferálni a mohó újrakapcsolási algoritmusoknál?

12. kérdésre/megjegyzésre adott válasz: A mohó újrakapcsolási algoritmusoknál nem feltétlenül érdemes mindig a rövid rész-szakaszokat preferálni. Az eredmény minőségét elsősorban az befolyásolja, hogy a kivágott részútvonalterv első és utolsó útszakasza, illetve a beillesztés helye körüli útszakaszok mennyire illeszkednek egymáshoz, azaz mekkora a távolság a kapcsolódási pontok között. A részútvonalterv hossza akkor válhat lényegessé, ha az eredeti útvonalterv már viszonylag jól optimalizált. Ilyenkor kisebb módosítások (rövidebb részútvonaltervek áthelyezése) elegendőek lehetnek, míg egy kevésbé jó útvonaltervnél akár hosszabb részútvonaltervek mozgatása is jelentős javulást eredményezhet.

13. kérdés/megjegyzés: Miben nyilvánul meg az ABC eljárás előnye a vizsgált feladatban a szintén elterjedt PSO optimalizációs eljárással összevetve?

13. kérdésre/megjegyzésre adott válasz: A vizsgált feladatra konkrét PSO-alapú megoldást nem találtam, ezért közvetlen összehasonlítást nem tudok végezni. Általánosságban azonban elmondható, hogy az ABC algoritmus jól teljesít összetett problémák globális optimalizálásában, mivel képes a keresési tér széles körű feltérképezésére és hatékonyan elkerüli a lokális optimumokat a felderítő mechanizmus (angolul scout mechanism) révén. Az ABC egyik további előnye, hogy kevesebb érzékeny paraméterrel rendelkezik (pl. a méhek száma), így a paraméterhangolás egyszerűbb, mint a PSO esetében.

Ezzel szemben a PSO algoritmus jellemzően gyorsabban konvergál, de nagy dimenziójú keresési terekben hajlamos a korai konvergenciára, amely gyengébb megoldásokhoz vezethet [4, 5]. Ráadásul kombinatorikus optimalizálási problémák esetén (mint amilyen a CARP is), a PSO bináris változatát (BPSO) kell alkalmazni. Ez nehezebb, mivel a diszkrét keresési tér miatt a részecskék mozgása csak valószínűségi alapon értelmezhető, és a paraméterhangolás is bonyolultabb.

Az ABC és a PSO algoritmusok strukturált összehasonlítását az alábbi táblázat (3. táblázat) foglalja össze. Mivel a vizsgált probléma egy összetett, kombinatorikus keresési térrel rendelkező optimalizálási feladat, több lokális optimum jelenlétével, az ABC algoritmus alkalmazása előnyösebbnek tekinthető.

3. táblázat: Az ABC és a PSO eljárás összehasonlítása

Jellemző	**ABC előnyei**	**PSO korlátai**
Diverzitás megőrzése	Jó (felderítő mechanizmus révén)	Gyenge (a konvergencia dominál)
Diszkrét problémák kezelése	Természetesen kezelhető	Nehézkes (BPSO alkalmazása szükséges)
Paraméterérzékenység	Alacsony	Magas
Lokális minimum elkerülése	Hatékony	Gyakran beleesik
Skálázhatóság	Jó kisebb populációval is	Nagy populációval stabilabb

Szeretném megköszönni Dr. Kovács László professzor úrnak, hogy elvállalta és elkészítette a bírálatot. A bírálatban megfogalmazott kérdések és észrevételek nagyon hasznosak voltak számomra.

Veszprém, 2025. június 12.

.....Nagy Zsuzsanna.....

Nagy Zsuzsanna
doktorandusz
Pannon Egyetem
Informatikai Tudományok Doktori Iskola

Hivatkozások

[1] B. F. Hompes, J. C. Buijs és W. M. van der Aalst, „A generic framework for context-aware process performance analysis", *On the Move to Meaningful Internet Systems: OTM 2016 Conferences: Confederated International Conferences: CoopIS, C&TC, and ODBASE 2016, Rhodes, Greece, October 24-28, 2016, Proceedings*, C. Debruyne és tsai., szerk., Lecture Notes in Computer Science sor., Springer, Cham, 10033. köt., 2016, 300–317. old. doi: https://doi.org/10.1007/978-3-319-48472-3_17.

[2] M. Albayrak és N. Allahverdi, „Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms", *Expert Systems with Applications*, 38. évf., 3. sz., 1313–1320. old., 2011. doi: https://doi.org/10.1016/j.eswa.2010.07.006.

[3] J. Carmona, B. van Dongen és M. Weidlich, „Conformance checking: foundations, milestones and challenges", *Process Mining Handbook*, Springer, Cham, 2022, 155–190. old. doi: https://doi.org/10.1007/978-3-031-08848-3_5.

[4] V. R. Kulkarni és V. Desai, „ABC and PSO: A comparative analysis", *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, India, December 15-17, 2016, Proceedings*, IEEE, New York, NY, USA, 2017, 1–7. old. doi: https://doi.org/10.1109/ICCIC.2016.7919625.

[5] M. Butt, „Optimization with Swarm Intelligence: A Comparative Study of Artificial Bee Colony and Particle Swarm Optimization Algorithms", diplomadolgozat, Itä-Suomen yliopisto (University of Eastern Finland), 2023. cím: https://erepo.uef.fi/bitstream/handle/123456789/30887/urn_nbn_fi_uef-20231405.pdf.