Válasz opponensi véleményre - doktori disszertáció

Véleményező: Dr. Gyulai Dávid
Cím: Ipar 4.0 és 5.0 megoldások fejlesztése ontológiák alapján - modellezés és optimalizálás - doktori dolgozat
Szerző: Nagy László
Témavezető: Dr. Abonyi János és Dr. Ruppert Tamás

Tisztelt Dr. Gyulai Dávid!

Nagyon szépen köszönöm konstruktív és rendkívül hasznos kommentjeit a disszertáció kapcsán. Alaposan átnéztem az összes megjegyzést és ezekre részletesen válaszolok jelen levélben.

Köszönettel:
Nagy László

2023. Május 31., Veszprém

A fejezettel kapcsolatos korábbi hiányérzetemet, miszerint nem került ismertetésre „vajon miért nem elterjedt még mindig az ipari gyakorlatban (általános, mindennapi alkalmazás), az egyébként technológiai szempontból érett, és tudományosan igazolt hatékonyságú megoldások használata” új tartalmi elemek ismertetik. Ezt érdemesnek tartom a szóbeli vitán további rövid tárgyalásra.

Köszönöm a visszajelzést! A nyilvános doktori védésemre készülök extra diával, ahol visszatérhetünk ennek a témának a megvitatására.
Illetve néhány további alkalmazási példát megemlítenék, ami bizonyítja, hogy az ipari elterjedés is folyamatban van már. A Bosch-nál Data scientist projektek keretében adatmodellezésben alkalmazzások az ontológiákat, valamint kimondottam termelő rendszerek esetén a ‘LIS: A Knowledge Graph-based Line Information System’ rendszert dolgozták ki MES és ERP feladatok ellátására ontológiák alapján. A Kuka Robotisc-nál ‘Semantic Data Management’-et alkalmaznak az igen komplex és nagy számú robotkar pozíciók, mozgásminták és művelet módok modellezésére. Továbbá személyes tapasztalatom a Siemens Energy-nél is találkoztam szemantikus technológiákkal, amikor az ‘Energy Transmission’ üzletágban dolgoztam 2021-ben, komplex erősáramú hálózatok gráf alapú modellezésére és optimalizálásra alkalmazzák.
Az I.1.1 ábra szemléletesen mutatja be az ontológiák illetve tudásgráfok irodalmának fejlődését: mi lehet az oka annak, hogy előbbi mintha „stagnálna”? Az előnyök és hátrányok összehasonlítása jól ismerteti a két terület közötti különbségeket, de számomra nem egyértelmű hogy ezek alapján ekkor különbségnek kellene mutatkoznia.

Az ábra készítése során csak a Scopus adatai kerültek felhasználásra, ami okozhat torzítást, valamint feltételezhetően a publikációk címében, és kulcsszavakban az ‘ontológiát’ egyre inkább a ‘tudás gráf’ kifejezés veszi át. Továbbá ami ezen ábrának nem része a ‘semantic technology’ kifejezés is gyakran előfordul a korábbiak mellett.

Az ontológiák előnyeinél szerepel, hogy a SPARQL gyorsabb lekérdezéseket tesz lehetővé adott esetekben, ez számomra némileg ellentmond a későbbi pontban megfogalmazott számítási komplexitásoknak.

Alapvetően a SPARQL az RDF típusú adatok lekérdezésére lett tervezve, így ha az adatok RDF formátumban vannak strukturálva, akkor a SPARQL hatékony választás lehet. Azonban a lekérdezések bonyolultsága, az adatkészlet mérete, heterogenitása és a különböző technológiák által

alkalmazott optimalizálási technikák mind befolyásolhatják a teljesítményt. Tehár érdemes a különböző adat lekérdezési technológiákat tesztelni és összehasonlítani a specifikus környezetben, hogy meghatározható legyen az optimális.
Illetve ipari példaként megemlíteném az amerikai, big data elemzésére szakosodott Palantir vállalatot, melynek egyik fő szoftver szolgáltatása a 'Palantir Foundry' egy ontológia alapú vállalatirányítási rendszer ami többek között képes élő adat elemzésre, számos optimalizációs megoldást nyújt, valamint machine learning és AI eszközök integrálásának köszönhetően a skálázhatóság is megoldott komplex, és nagy mennyiségű adatok esetén is (https://www.palantir.com/platforms/foundry/).

A II.2 fejezet egy bonyolult korlátozás-rendszerrel (task-operator-equipment-skill) bíró sorkiegyenlítési feladatra ad heurisztikus megoldást. A bemutatott új tudományos eredményeket a tesztek igazolják, valamin az átolgozott változat tartalmaz egy érzékenységvizsgálati kísérletsorozatot. Az eredmények alapján a megoldás érzékeny néhány paraméter megváltozására, a kérdésem, hogy ezeket az eredményeket figyelembe véve a gyakorlatban mely megoldást javasolná megvalósításra, illetve mi alapján lehet eldönteni hogy az alternatív megoldások melyike a legmegfelelőbb ipari környezetben?

A feladat megoldható paraméter optimalizálással, vagy egyszerűen az egyes faktorok súlyozásával, amelyhez az adott ipari területet domain tudása adja az alapot. Így az alternatív megoldások közül az adott termelési kritériumok mellett, az idő-, képzettség- és eszköz alapú célok (költségek) közötti megfelelő egyensúlyt adó megoldást választanám. Továbbá, adott ipari környezettől függően változhatnak az idő alapú cél mellett az eszköz- és a képzettség célok fontossága, melyet befolyásolhatnak például a termelő folyamatok technológiai kötöttségei, erőforrások vagy megmunkáló gépek limitációja.

A saját eredmények a korábbi változathoz képest hangsúlyosabban jelennek meg a fejezetben, a megoldás skálázhatóságáról továbbra is kevés szó esik. Van erre vonatkozóan valamilyen teszteredmény, esetleg tapasztalat?

A skálázhatóságát kifejezetten nem vizsgáltam, ugyanakkor a kidolgozott és továbbfejlesztett szimulált hűtéses optimalizáció hatékonyságának köszönhetően nagyobb ipari problémákon is gond nélkül alkalmazhatónak kell lennie. Az algoritmus komplexitása alapvetően nem változik, mivel a kombinált célfüggvény az eredeti célok lineáris kombinációja (a részcélok egymásnak részben ellentmondanak) így a keresési tér komplexitása sem változik. Ugyanakkor olyan eset előfordulhat, hogy gyorsabb lesz az optimalizációs algoritmus eredményének konvergenciája, ha csak egy cél kerül figyelembe vételre.

A II.3.1.2 ismerteti az algoritmus inicializálását, amely jellemzően valamilyen véletlen állapotból indul. A doktorjelölt megemlíti, hogy célszerű lehet többször futtatni az algoritmust különböző, véletlenszerűen generált kezdeti állapotokból. Ilyenkor, adott esetben különböző megoldáshoz jutva, hogyan lehet aggregálni az eredményeket, és javaslatot tenni a végső megoldásra?

A többszöri futtatás csak az algoritmus hangolására vonatkozik, melyet minden hálózat esetén az első alkalmazásnál kell megtenni, ekkor az eredmények csakis külön értelmezhetőek és a legkedvezőbbek alapján választható ki a paraméter halmaz, melynek megoldása természetesen kiváltható akár paraméter optimalizálással is. A kidolgozott algoritmus előnye, hogy az adott hálózaton a kezdeti felparaméterezést követően már nagyságrenddel gyorsabban képes megtalálni a közösségeket a hálózatban, ezáltal hatékonyan elemezhető az adott rendszer akár periodikusan, akár online módon.

Az új táblázatos eredményeknél nem világos (II.3.2, II.3.3 táblázatok), hogy mit jelent a C (talán ez a közösségek száma?), és miért van említve m a táblázat szövegében, ugyanis nincs használva. Talán gépelési hiba?

Ehhez kapcsolódó kérdés, hogy amennyiban C –> m¸hogyan lehetséges hogy pl. a második teszt esetén az érték (19) kívül esik a minimum és maximum által megadott (20-50) tartományon?

A ‘II.3.1.1 Cost function - Modularity’ fejezetben említésre kerül, hogy a ‘C’ az adjacencia mátrix partícióinak száma. A II.3.2, II.3.3 táblázatok leírásának esetében sajnos gépelési hibát ejtettem, és helyesen valóban ‘community number (C)’. Továbbá az ‘m(min)’ és ‘m(max)’ értékek node számban kifejezve adják meg, hogy minimum, illetve maximum hány darab csomópont kerülhet egy közösségbe a mesterséges generálás (LFR) során. Tehát például II.3.2-es táblázat esetén 1000 csomóponttal rendelkező hálózatban az adott csoportokban minimum 20, maximum 100 node kerülhet (ezen tartományban véletlenszerűen), így 19, illetve 24 értékű a ‘C’ közösségek száma a hálózatokban.

Jól értem, hogy a saját megoldás kevésbé érzékeny a paraméterek változása, a vizsgált egyéb módszerekhez képest? Kérem tisztázni, hogy miért az adott módszereket vizsgálta, illeve valóban ezek-e jelenleg a terület leghatékonyabb algoritmusai.

Köszönhetően annak, hogy a megfelelő hangolás mellett (gamma és resolution értékek) a bemutatott algoritmus képes detektálni a megfelelő számú közösséget a hálózatban (II.3.5. ábra), az NMI és ARI mutatószámok is magasabbak a többi módszeréhez képest. Az algoritmus implementálása Matlab környezetben történt, így az összehasonlítás során toolbox-okban elérhető és hasonlóan modularitás alapú közösség detektáló algoritmusokkal történt az összehasonlítás. Továbbá a jelenleg is folyamatban lévő publikálás során a teljes algoritmust, a széleskörűbb tesztelés érdekében implementáltam Python-ban, ahol az alábbi algoritmusokkal is összehasonlítom: ‘Greedy, Louvain, Label propagation, Leiden modularity’. Ezen tesztek során is az eredmények hasonló tartományban vannak, mint ahogy a disszertációban bemutatásra került.