Študija učinkovitosti algoritma za razporejanje terenskega dela

A Study of the Performance of a Fieldwork Scheduling Algorithm

Tea Tušar
Institut "Jožef Stefan" in
Mednarodna podiplomska šola
Jožefa Stefana
Jamova cesta 39
Ljubljana, Slovenija
tea.tusar@ijs.si

Nace Sever
Univerza v Ljubljani
Fakulteta za računalništvo in
informatiko
Večna pot 113
Ljubljana, Slovenija
nace.sever@gmail.com

Aljoša Vodopija
Bogdan Filipič
Institut "Jožef Stefan" in
Mednarodna podiplomska šola
Jožefa Stefana
Jamova cesta 39
Ljubljana, Slovenija
aljosa.vodopija@ijs.si
bogdan.filipic@ijs.si

POVZETEK

Razporejanje terenskega dela je zahteven optimizacijski problem. Za njegovo reševanje smo razvili trinivojski algoritem. Na prvem nivoju evolucijski algoritem razporedi naloge po delavcih, na drugem nivoju hevristika za vsakega delavca razporedi naloge po dnevih, na tretjem nivoju pa algoritem razveji in omeji rešuje problem mešanega celoštevilskega linearnega programiranja (angl. Mixed-Integer Linear Programming, MILP), kjer nalogam za vsakega delavca in vsak dan posebej dodeli čas njihovega začetka. V tem prispevku se posvečamo študiji učinkovitosti algoritma na tretjem nivoju. Izkaže se, da ta nalog ne more razporediti dovolj hitro za praktično uporabo, zato za povečanje njegove učinkovitosti MILP poenostavimo. Rezultati poskusov kažejo, da poenostavitev izboljša učinkovitost algoritma na tretjem nivoju, medtem ko je učinek na celoten algoritem načeloma ugoden, a odvisen od problema.

KLJUČNE BESEDE

problem razporejanja, evolucijski algoritem, hevristika, algoritem razveji in omeji, mešano celoštevilsko linearno programiranje, učinkovitost

ABSTRACT

Fieldwork scheduling is a demanding optimization problem. To solve it, we developed a three-level algorithm. At the first level, an evolutionary algorithm distributes tasks to workers, at the second level, a heuristic algorithm distributes tasks of each worker over days, and at the third level, a branch-and-bound algorithm solves the problem in the form of mixed-integer linear programming (MILP), where the starting times of tasks need to be scheduled for each worker and each day separately. In this paper, we study the efficiency of the algorithm at the third level. Because it cannot schedule the tasks fast enough for practical use, we try to increase its efficiency by simplifying the MILP. Experimental results show that the simplification improves the performance of the algorithm at the third level, while the effect on the overall algorithm is in principle favorable, but depends on the problem.

Information Society 2022, 10–14 October 2022, Ljubljana, Slovenia

KEYWORDS

scheduling problem, evolutionary algoirthm, heuristic algorithm, branch-and-bound algorithm, mixed-integer linear programming, efficiency

1 UVOD

Razporejanje terenskega dela je optimizacijski problem, ki zahteva dodelitev delavca in časa začetka opravljanja vsaki terenski nalogi tako, da je zadoščeno vsem omejitvam razporejanja in je cena celotnega urnika čim nižja. Obstajajo številne različice tega problema, ki se razlikujejo tako po omejitvah kot po načinu izračuna cene razporeda. Posledično obstajajo tudi različni pristopi za njegovo reševanje [6]. Študija [3] primerja dve formulaciji problema, in sicer v obliki problema usmerjanja vozil (angl. Vehicle Routing Problem, VRP) in v obliki problema mešanega celoštevilskega linearnega programiranja (angl. Mixed-Integer Linear Programming, MILP). Rezultati poskusov študije nakazujejo, da je oblika MILP za zapis razporejanja nalog terenskega dela ustreznejša od oblike VRP, zato tudi naš pristop uporablja obliko MILP.

Vendar pa je učinkovitost reševanja takšnih kombinatoričnih problemov zelo odvisna od njihove velikosti. Že pri relativno majhnih problemih se namreč pogosto zgodi, da jih ni moč rešiti v doglednem času. Zato se v našem pristopu zgledujemo po podobnih prijemih iz sorodnega dela (glej npr. [1]) in problem razdelimo na manjše, lažje obvladljive podprobleme. Problem razporejanja terenskega dela tako rešujemo s trinivojskim optimizacijskim algoritmom, pri katerem na prvem nivoju evolucijski algoritem razporedi naloge po delavcih, na drugem nivoju hevristika za vsakega delavca razporedi naloge po dnevih, na tretjem nivoju pa algoritem razveji in omeji rešuje problem v obliki MILP, tj. nalogam za vsakega delavca in vsak dan posebej dodeli čas njihovega začetka.

Tak trinivojski algoritem je sposoben v uri zadovoljivo rešiti tudi nekoliko večje probleme (npr. z 20 delavci, 20 dnevi in več sto nalogami), a je ta čas za praktično uporabo predolg. Zato želimo algoritem pohitriti. Ozko grlo predstavlja reševanje problema MILP, saj sta evolucijski algoritem in hevristika zelo hitra, tako da lahko največjo pohitritev celotnega algoritma dosežemo s pohitritvijo na tretjem nivoju.

V nadaljevanju prispevka v 2. razdelku najprej predstavimo našo različico problema razporejanja terenskega dela, nato pa v 3. razdelku na kratko opišemo trinivojski algoritem za njeno reševanje. V 4. razdelku analiziramo učinkovitost algoritma na tretjem nivoju, v 5. razdelku pa predlagamo poenostavitev problema MILP in preverimo njen učinek najprej na algoritem na

tretjemu nivoju in končno na celoten trinivojski algoritem. Prispevek sklenemo z zaključki v 6. razdelku.

2 PROBLEM RAZPOREJANJA TERENSKEGA DELA

Problem razporejanja terenskega dela opišemo s scenarijem razporejanja, spremenljivkami problema, omejitvami in optimizacijskim kriterijem (podrobne formalne definicije tu ne moremo zapisati zaradi pomanjkanja prostora). Obravnavamo najbolj splošno različico problema, v kateri želimo razporediti večino nalog, saj ta pokriva tudi posebni primer, ko je zaradi spremembe v zadnjem trenutku treba prerazporediti samo nekaj nalog.

2.1 Scenarij razporejanja

Časovno obdobje razporejanja je razdeljeno na dneve, znotraj njih je čas obravnavan zvezno. Za vsak dan poznamo začetek in konec rednega delovnika ter trajanje morebitnih nadur (bodisi na začetku bodisi na koncu dneva). Dane imamo tudi množico *lokacij*, časovne oddaljenosti za vsak par lokacij ter množico *kompetenc*, ki so skupne nalogam in delavcem. Scenarij razporejanja vsebuje tudi podatke o *delavcih*, in sicer za vsakega kompetence, dovoljeno število nadur ter začetno in končno lokacijo. Podatki o *nalogah* pa za vsako obsegajo njeno trajanje, želeno in obvezno časovno okno, prioriteto, zahtevane kompetence in morebitne želene delavce. *Malice* so posebne naloge, za katere lokacija ni definirana (malica se vedno izvaja na isti lokaciji kot predhodna naloga in se ne more prekrivati z drugimi nalogami).

Dodatno lahko scenarij razporejanja vsebuje že vnaprej pripravljene razporede posameznih nalog, ki so dveh tipov. *Obveznih razporedov* se ne sme spreminjati, a jih je treba vseeno upoštevati, saj postavljajo omejitve k razporejanju ostalih nalog. Po drugi strani pa se *želene razporede* lahko spreminja, a to vpliva na ceno končnega urnika.

2.2 Spremenljivke

Spremenljivke optimizacijskega problema v celoti določijo urnik, saj za vsako nalogo povedo ali je razporejena ali ni (ni namreč treba razporediti vseh nalog) in če je, kateri delavec jo bo opravil ter kdaj se bo začela izvajati.

2.3 Omejitve

Urnik, ki predstavlja rešitev problema, je dopusten samo, če izpolnjuje vse naslednje omejitve:

- Delavec lahko izvaja samo eno nalogo hkrati (v časovnem razporejanju nalog je treba poskrbeti tudi za upoštevanje trajanja potovanja med lokacijami).
- Delavec lahko izvaja naloge le znotraj delovnega časa in ima omejeno število nadur.
- Delavec mora imeti zahtevane kompetence za opravljanje naloge.
- Naloge morajo biti razporejene znotraj svojih obveznih časovnih oken.
- Nalog z obveznim razporedom se ne sme prerazporejati.

2.4 Optimizacijski kriterij

Optimizacijski kriterij oz. cena urnika, ki jo želimo minimizirati, je definirana kot utežena vsota naslednjih delnih kriterijev (prve tri postavke so si v nasprotju, zato je smiselno upoštevati samo eno od njih naenkrat):

- Vsi delavci naj bodo čim bolj enakomerno obremenjeni.
- Dnevno aktivnih delavcev naj bo čim manj.
- Aktivni delavci naj bodo čim bolj enakomerno obremenjeni.
- Skupno trajanje potovanj med lokacijami naj bo čim krajše.
- Izvede naj se čim več nalog.
- Naloge naj se izvedejo čim prej.
- Delavci naj imajo čim manj neaktivnega časa.
- Nadur naj bo čim manj.
- Naloge z višjo prioriteto naj se začnejo izvajati pred nalogami z nižjo prioriteto.
- Naloge, ki zapadejo prej, naj se začnejo izvajati pred nalogami, ki zapadejo kasneje.
- Naloge naj se izvedejo čim bliže želenemu časovnemu oknu.
- Nalogo, ki ima želene delavce, naj opravi eden izmed želenih delavcev.
- Nalogo z želenim razporedom naj se izvede čim bliže temu razporedu.

Uteži posameznih delnih kriterijev so zelo pomembne, saj določajo njihova medsebojna razmerja in drastično vplivajo na dobljene rešitve. Nastavili smo jih s pomočjo ekspertnega znanja in poskusov na številnih različnih scenarijih.

3 TRINIVOJSKI OPTIMIZACIJSKI ALGORITEM

V nadaljevanju na kratko predstavimo vse tri nivoje optimizacijskega algoritma.

3.1 Prvi nivo: razporejanje nalog po delavcih

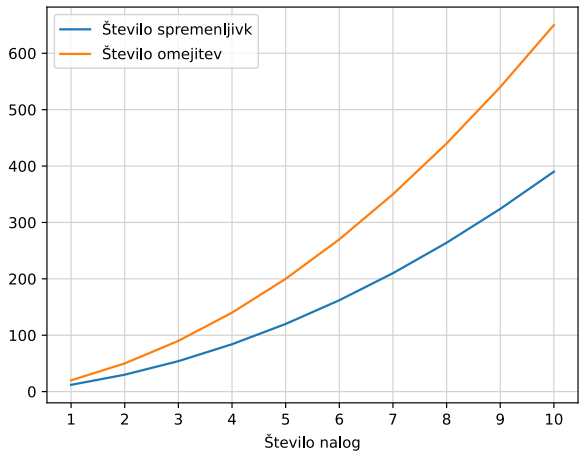
Na tem nivoju z evolucijskim algoritmom [5] vsaki nalogi dodelimo delavca, ki jo bo izvedel. Evolucijski algoritem začetno populacijo N_p rešitev ustvari naključno, vendar tako, da vse rešitve ustrezajo omejitvam za delavce (prve tri omejitve v razdelku 2.3). Potem algoritem izvaja naslednje korake največ N_g generacij. V vsaki generaciji algoritem najprej izbere N_p staršev s turnirsko selekcijo. Nato pare staršev križa in mutira (pri mutaciji uporabimo različne strategije zasnovane po meri delnih kriterijev (glej razdelek 2.4), ki jih izbiramo tako, da se pogostost uporabe sklada z njihovimi utežmi). Tako dobljeno populacijo evolucijski algoritem ovrednoti tako, da za vsako rešitev izvede drugi in tretji nivo algoritma. Nato staro populacijo prepiše z novo (najboljšo staro rešitev ohrani) in nadaljuje z enakimi koraki.

3.2 Drugi nivo: razporejanje nalog delavca po dnevih

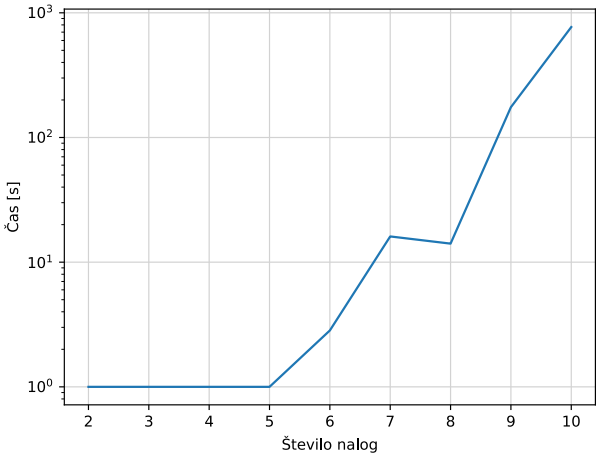
Hevristika na drugem nivoju naloge vsakega delavca razporedi po dnevih. Po vrsti vsem nalogam, urejenim naraščajoče po številu dni, v katerih se lahko izvedejo, dodelimo zanje najugodnejši dan. Ugodnost dne določimo z glasovanjem, ki poteka tako, da različni delni kriteriji (glej razdelek 2.4) glasujejo za dneve, ki so zanje najugodnejši. Glasovi so uteženi z utežmi delnih kriterijev, nalogi pa dodelimo dan z največ glasovi.

3.3 Tretji nivo: določitev časa začetka nalog za en dan enega delavca

Na tretjem nivoju z algoritmom razveji in omeji nalogam za en dan enega delavca dodelimo začetni interval. Problem torej zapišemo v obliki MILP tako, da upoštevamo samo tiste omejitve in delne kriterije, ki so na tem nivoju še smiselni (npr. na tem



Slika 1: Odvisnost števila spremenljivk in omejitev v formulaciji problema MILP na tretjem nivoju od števila nalog.



Slika 2: Povprečen čas, potreben za optimalno rešitev problema glede na njegovo velikost.

nivoju se ne ukvarjamo več s kompetencami, enakomerno obremenjenostjo delavcev in podobnimi delnimi kriteriji, saj je zanje poskrbljeno na prvih dveh nivojih).

Podobno kot pri predstavitvi problema tudi tu zaradi omejenega prostora ne moremo navesti celotne formulacije problema MILP. Za razumevanje nadaljevanja je najpomembneje vedeti, da imamo pri takšni formulaciji za problem z n nalogami $3n^2 + O(n)$ spremenljivk in $5n^2 + O(n)$ omejitev, kot prikazuje slika 1.

4 PREIZKUS UČINKOVITOSTI

Kot je razvidno iz slike 1, je število spremenljivk problema MILP zelo veliko že za probleme z majhnim številom nalog, kar otežuje nalogo optimizacijskemu algoritmu. Čas, ki ga potrebuje za najdbo optimalne rešitve, preverimo s poskusom na množici testnih problemov, ki imajo lastnosti podobne problemom iz prakse.

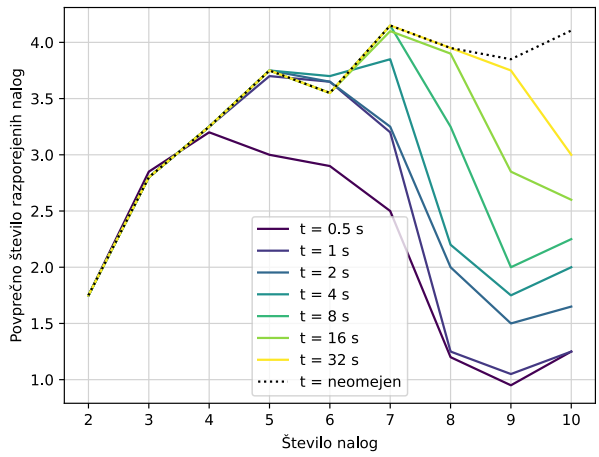
Ta množica vsebuje 180 testnih problemov (20 za vsako velikost problema od dveh do desetih nalog), pri katerih je treba določiti čas izvajanja nalog za enega delavca v enem dnevu. Nekateri problemi imajo samo navadne naloge, lahko pa imajo tudi malico, eno nalogo z obveznim razporedom ali pa oboje. Trajanje malice je vedno pol ure, trajanje ostalih nalog pa je izbrano naključno iz porazdelitve, ki skuša posnemati probleme iz prakse. Tako je večina nalog krajših od 90 minut, nekaj pa jih ima dolžino do štirih ur. Prioriteta vsake naloge je izbrana naključno med 1 in 9. Prav tako so lokacije izvajanja nalog in začetna lokacija delavca izbrane naključno izmed lokacij nekaterih večjih slovenskih mest. Večina nalog ima neomejeno časovno okno, pri nekaterih pa je okno skrajšano na začetku ali koncu dneva. Trajanje delovnega časa je izbrano naključno med 6 in 10 ur, lahko pa delavec vedno opravlja do dve naduri.

Vse testne probleme rešujemo z algoritmom razveji in omeji iz reševalnika SCIP [2] preko knjižnice OR-Tools [4]. Pri tem beležimo čas, ki ga algoritem potrebuje, da najde optimalno rešitev. Reševanje poteka na osebnem računalniku s 16 GB pomnilnika in frekvenco procesorja 3,60 GHz.

Rezultati poskusa so prikazani na sliki 2. Vidimo, da algoritem praviloma potrebuje eksponentno več časa z dodajanjem vsake naloge. Če želimo, da je celoten trinivojski algoritem koristen v praksi, si lahko za reševanje problema MILP privoščimo le nekaj sekund, kar pomeni, da je za naše potrebe algoritem neučinkovit že za probleme s sedmimi ali več nalogami.

Preverimo še, kako dobro deluje algoritem, če mu omejimo čas, ki ga ima na voljo za iskanje rešitev. Poskuse izvedemo z naslednjimi časovnimi omejitvami: 0.5 s, 1 s, 2 s, 4 s, 8 s, 16 s in 32 s. Pri tem opazujemo, koliko nalog je algoritem razporedil, in to število primerjamo z optimalnim številom razporejenih nalog (dobljenim v prejšnjem poskusu, ko algoritem ni bil časovno omejen). Čeprav cilj algoritma ni samo razporediti čim več nalog, je število razporejenih nalog dober pokazatelj kakovosti delovanja algoritma.

Na sliki 3 vidimo, da ob prekratkem času na problemih z veliko nalogami algoritem odpove (večino nalog zavrne, čeprav bi jih lahko razporedil). Na primer, ko ima algoritem na voljo le 0,5 s, primerno deluje le za probleme z do štirimi nalogami, za večje probleme pa njegova uspešnost pade in ko je nalog osem ali več, v povprečju razporedi le eno nalogo. Delovanje algoritma je nekoliko boljše, če ima na voljo daljši čas, a šele pri 32 s se



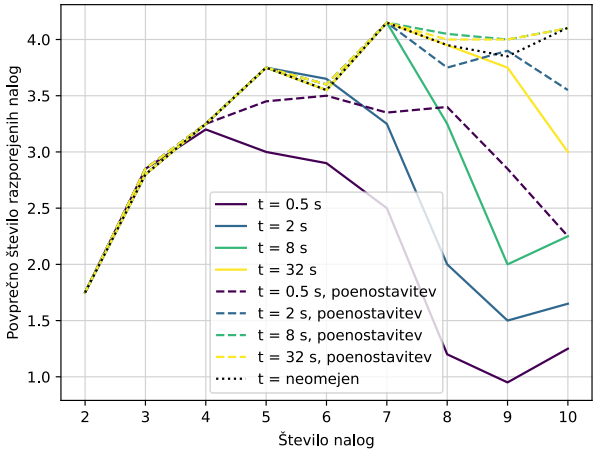
Slika 3: Povprečno število razporejenih nalog pri različnih časovnih omejitvah in velikostih problemov (s črtkano črno črto je prikazano optimalno število razporejenih nalog).

število razporejenih nalog na problemih z devetimi in desetimi nalogami približa optimalnemu številu razporejenih nalog.

5 POENOSTAVITEV PROBLEMA

Ker z delovanjem algoritma nismo zadovoljni, poskusimo problem poenostaviti. Za zapis delnih kriterijev za prioriteto in čas zapadlosti potrebujemo $2n^2 + O(n)$ spremenljivk ter $4n^2 + O(n)$ omejitev, kar je zelo veliko, sploh ker ta dva delna kriterija nista zelo pomembna. Zato preizkusimo, kako algoritem deluje, če ju izpustimo (zanju lahko do neke mere poskrbimo na zgornjih dveh nivojih optimizacijskega algoritma). Število spremenljivk in omejitev se še vedno povečuje kvadratično s številom nalog, vendar pa smo koeficienta pred kvadratnim členom s 3 oziroma 5 zmanjšali na 1.

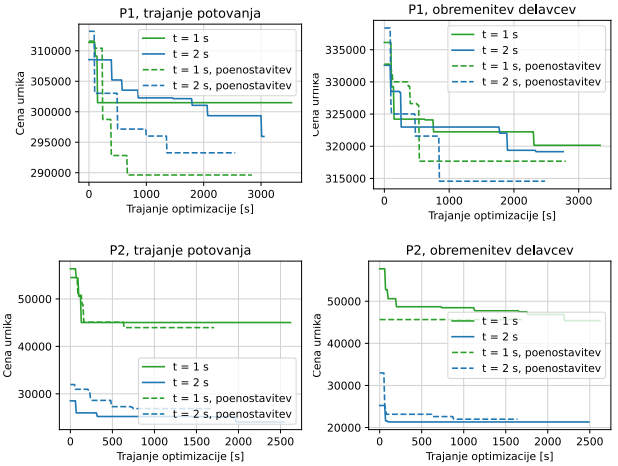
Za poenostavljeni problem izvedemo podoben test kot v razdelku 4, le da testiramo samo pri časovnih omejitvah 0.5 s, 2 s, 8 s in 32 s. Na sliki 4 primerjamo število razporejenih nalog pri osnovnemu ter poenostavljenemu problemu. Vidimo, da na večjih poenostavljenih problemih algoritem deluje mnogo bolje.



Slika 4: Primerjava delovanja algoritma na izvirni (polne črte) in poenostavljeni formulaciji problema (črtkane črte) pri različnih časovnih omejitvah in velikostih problemov (s črno črtkano črto je prikazano optimalno število razporejenih nalog izvirne formulacije problema).

S poenostavitvijo torej dosežemo, da algoritem na tretjem nivoju deluje zadovoljivo tudi za praktične potrebe. Vendar pa to še ne pomeni nujno, da poenostavitev izboljša delovanje celotnega trinivojskega algoritma. To preverimo s poskusom na dveh testnih problemih, P1 in P2, pri katerih damo enkrat večjo utež delnemu kriteriju trajanja potovanja, drugič pa enakomerni obremenitvi delavcev. Na ta način dobimo štiri različne testne probleme. P1 obsega 220 nalog, deset delavcev in sedem dni, P2 pa 114 nalog (vsebuje tudi malice), pet delavcev in tri dni. Za vsak testni problem poženemo štiri različice trinivojskega algoritma, ki se razlikujejo samo na tretjem nivoju – ta uporablja bodisi izvirni bodisi poenostavljeni problem, izvajanje algoritma pa je omejeno bodisi na 1 s bodisi na 2 s.

Slika 5 kaže rezultate teh poskusov. Na problemu P1 (zgornja dva grafa na sliki) lahko jasno vidimo, da poenostavitev problema na tretjem nivoju koristi učinkovitosti celotnega algoritma. Tega ne moremo trditi za problem P2, na katerem je delovanje izvirne in ponostavljene različice zelo podobno, vidimo pa veliko boljše delovanje v primeru omejitve izvajanja na 2 s.



Slika 5: Rezultati optimizacije za štiri različice algoritma na dveh problemih (P1 zgoraj in P2 spodaj) z dvema različnima delnima kriterijema (trajanje potovanja levo in enakomerna obremenitev delavcev desno). Manjše vrednosti so boljše.

6 ZAKLJUČKI

V prispevku smo analizirali učinkovitost algoritma za razporejanje terenskega dela. Posvetili smo se le časovno najzahtevnejšemu delu trinivojskega algoritma – reševanju problema MILP na tretjem nivoju. Z dvema poskusoma smo pokazali, da algoritem razveji in omeji ni dovolj učinkovit za reševanje praktičnih problemov, zato smo problem MILP poenostavili. To ne spremeni kriterijev celotnega problema, algoritmu na tretjem nivoju pa omogoči, da učinkovito reši tudi probleme z desetimi nalogami (več jih v praksi ne pričakujemo). Primerjali smo tudi, kako poenostavitev vpliva na delovanje celotnega algoritma, in ugotovili, da čeprav obstajajo problemi, za katere poenostavitev ni koristna, v splošnem daje dobre rezultate in se je bomo posluževali tudi v praksi.

ZAHVALA

To delo je nastalo v okviru projekta Inteligentno in okolju prijazno razporejanje terenskega dela – MF-Scheduler, katerega naročnik je Comland d.o.o., sofinancerja pa Ministrstvo za gospodarski razvoj in tehnologijo Republike Slovenije in Evropski sklad za regionalni razvoj Evropske unije, ter raziskovalnega programa P2-0209, ki ga financira Javna agencija za raziskovalno dejavnost Republike Slovenije iz državnega proračuna.

LITERATURA

[1] S. Bertels in T. Fahle. 2006. A hybrid setup for a hybrid scenario: Combining heuristics for the home health care problem. *Computers & Operations Research*, 33, 10, 2866–2890. DOI: 10.1016/j.cor.2005.01.015.

[2] K. Bestuzheva in sod. 2021. The SCIP Optimization Suite 8.0. Technical Report. Optimization Online, (dec. 2021). http://www.optimization-online.org/DB_HTML/2021/12/8728.html.

[3] J. A. Castillo-Salazar, D. Landa-Silva in R. Qu. 2016. Workforce scheduling and routing problems: Literature survey and computational study. *Annals of Operations Research*, 239, 1, 39–67. DOI: 10.1007/s10479-014-1687-2.

[4] Google Developers. 2021. About OR-Tools. Retrieved 19. avg. 2022 from https://developers.google.com/optimization/introduction/overview.

[5] A. E. Eiben in J. E. Smith. 2015. *Introduction to Evolutionary Computing*. (2. izd.). Springer. DOI: 10.1007/978-3-662-44874-8.

[6] D. C. Paraskevopoulos, G. Laporte, P. P. Repoussis in C. D. Tarantilis. 2017. Resource constrained routing and scheduling: Review and research prospects. *European Journal of Operational Research*, 263, 3, 737–754. DOI: 10.1016/j.ejor.2017.05.035.